Expected System Energy Consumption Minimization in Leakage-Aware DVS Systems*

Jian-Jia Chen
Computer Engineering and Networks Lab. (TIK)
Swiss Federal Institute of Technology Zurich
(ETH Zurich), Switzerland
jchen@tik.ee.ethz.ch

Lothar Thiele
Computer Engineering and Networks Lab. (TIK)
Swiss Federal Institute of Technology Zurich
(ETH Zurich), Switzerland
thiele@tik.ee.ethz.ch

ABSTRACT

The pursuit of energy efficiency is becoming more and more important in hardware and software designs. This research explores energy-efficient scheduling for a periodic real-time task with uncertain execution time in dynamic voltage scaling (DVS) systems with non-negligible leakage/static power consumption. Distinct from the assumption of non-reducible static power consumption in the literature, this paper considers the possibility to reduce it by turning a processor to a dormant mode. We propose an algorithm to derive an optimal frequency assignment to minimize the expected energy consumption without procrastination, while another extended algorithm is developed to apply procrastination scheduling for further energy reduction. Experimental results show that the proposed algorithms can effectively minimize the expected energy consumption.

Categories and Subject Descriptors: D.4.1 SoftwareOPERATING SYSTEMS – Scheduling

General Terms: Algorithms, Design, Management

Keywords: Dynamic Voltage Scaling (DVS), Expected Energy Consumption Minimization, Leakage-Aware Scheduling, Probability

1. INTRODUCTION

Power management has become an important system design issue for embedded systems and server systems to prolong the operation duration or reduce power bills. Dynamic energy/power consumption due to switching activities and static energy/power consumption due to leakage current are two major sources of energy consumption of a processor [9]. The dynamic voltage scaling (DVS) technique was introduced to balance the dynamic energy consumption and the performance of a system, in which different supply voltages lead to different execution frequencies. The dynamic power consumption is usually a convex and increasing function of frequency, which motivates to execute at as low frequency as possible. However, executing at a lower frequency stretches the execution time with more leakage/static energy consumption.

*Support by grants from ROC National Science Council NSC-096-2917-I-564-121 and the European Community's Seventh Framework Programme FP7/2007-2013 project Predator.

ISLPED'08, August 11–13, 2008, Bangalore, India..

For real-time systems, energy-efficient scheduling is to minimize the energy consumption without making any task instance miss its deadline, provided that each task instance executes as its worst-case estimation. Once a task instance completes earlier than the worst-case estimation, the unused time (slack) can be reclaimed to reduce the energy consumption [1, 12]. In addition to worst-case estimations, we can also get the probability distribution of the workload of a task by profiling, and the scheduler can adjust the execution frequencies based on the distribution to provide a more energy-efficient schedule than applying slack reclamation. To minimize the *expected* dynamic energy consumption provided that the leakage/static power consumption is negligible or constant, some previous studies [5, 6, 11, 16–18] provide accelerating strategies. In particular, Gruian [5] considered the scheduling of multiple tasks and allocated execution time to tasks based on their worst-case execution cycles. Yuan and Nahrstedt [17] used the accelerating scheduling strategy for soft real-time multimedia tasks. Xu et al. [16] and Zhang et al. [18] explored inter-task scheduling for frame-based real-time tasks. Gruian and Kuchcinski [6] developed heuristics for task ordering to reduce the expected energy consumption. Xu et al. [15, 16] extended the accelerating scheduling for systems with discrete frequencies.

In nano-meter manufacturing, leakage current in CMOS circuits has significant contribution of the total power consumption, in which the static power consumption is comparable to the dynamic power dissipation [9]. To reduce the energy consumption resulting from leakage current, a processor might enter a dormant mode, in which the power consumption of the processor in the mode can be treated as negligible. However, turning the processor to the active mode requires time and energy overheads. For example, the Transmeta processor with the 70nm technology has $483\mu J$ energy overhead and less than 2 millisecond (ms) timing overhead [9]. Leakage-aware scheduling has been recently explored on DVS platforms, such as [3, 9, 10]. In particular, Chen and Kuo [4], Jejurikar et al. [9], and Lee et al. [10] developed procrastination scheduling strategies to decide when to turn the processor to the dormant mode.

However, the minimization of expected energy consumption in the literature assumes constant static power consumption and does not consider the possibility to turn the processor to a dormant mode. Their derived solutions minimize the dynamic energy consumption, but might consume too much static energy. This paper presents new approaches to minimize the expected system energy consumption for a real-time task with a discrete probability density function of the workload in leakage-aware DVS systems. To our best knowledge, this is the first paper for the exploration of the minimization of expected system energy consumption for tasks with uncertain execution time on platforms with the possibility to turn the processor off (to the dormant mode). As shown in the literature, e.g., [3, 9], there is a *critical frequency* in the available processor frequencies with the minimum energy consumption for execution. We show

that the simple extension of existing accelerating algorithms by taking the critical frequency as the minimum execution frequency is not optimal. An optimal schedule for leakage-aware DVS scheduling would execute with decelerating frequencies at the beginning to create more time slack to turn the processor off, and then execute with accelerating frequencies at the end to minimize the dynamic energy consumption. This paper presents an algorithm to derive an optimal frequency assignment to minimize the expected system energy consumption under the timing constraint. Then, based on the procrastination concepts in the literature [3,4,7,9], we also propose an algorithm to apply procrastination scheduling for further energy reduction if the processor has been in the dormant mode at the beginning of a task instance. Experimental results show that our algorithms can effectively minimize the energy consumption.

The rest of this paper is organized as follows: Section 2 provides system models and a motivational example. Section 3 presents our algorithms. Experimental results are presented in Section 4. Section 5 concludes this paper.

2. SYSTEM MODELS

2.1 Processor Models

The (system) power consumption function $P(f)$ of frequency f on a processor has two parts: $P_\delta(f)$ and P_{ind}, where $P_\delta(f)$ (P_{ind}, respectively) is dependent (independent, respectively) on the frequency [4]. The frequency-dependent power consumption $P_\delta(f)$ comes from the short-circuit power consumption and the dynamic power consumption due to the charging and discharging of gates on a CMOS DVS processor, while the leakage power consumption mainly contributes to P_{ind}. We explore the scheduling on processors with a continuous spectrum of the available frequencies between the upper-bounded frequency $f_{\max}$ and the lower-bounded frequency $f_{\min}$. It is easy to apply the algorithms in [2,8] to revise the derived solutions to systems with discrete frequencies only.

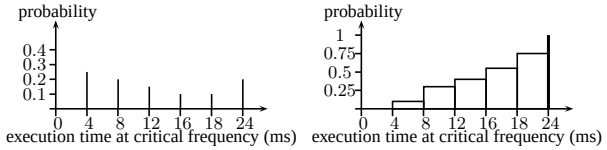
We assume that $P(f)$ is a strictly convex and increasing function, while $P(f)/f$ is merely a convex function [7]. For example, the most adopted frequency-dependent power consumption function $P_\delta(f) = \alpha f^\gamma$ satisfies the assumption for any positive α and any $\gamma > 1$. The *critical frequency* f^*, defined as the available frequency that minimizes the energy consumption for execution [3,4,7,9], is the frequency f with the minimum $\frac{P(f)}{f}$. For example, when $P_\delta(f) = \alpha f^\gamma$, f^* is $\max\left\{\min\{\sqrt[\gamma]{\frac{P_{ind}}{\alpha(\gamma-1)}}, f_{\max}\}, f_{\min}\right\}$.

A processor has two modes: *dormant mode* and *active mode*. When the processor is in the dormant mode, the power consumption of the processor is 0 by scaling the static power consumption [4, 7]. The processor has to be in the active mode for task execution. However, switching between the two modes takes time and consumes energy. Since periodic tasks are considered, the procedure to turn the processor to the dormant mode can be assumed instantaneously with negligible energy overhead by treating the overhead as a part of the overhead to turn on the processor. We denote E_{sw} (t_{sw}, respectively) as the energy (time, respectively) of the *switching overhead* from the dormant mode to the active mode.

When the processor is idle in the active mode, the processor executes NOP instructions at frequency $f_{\min}$ for energy minimization. When the processor is idle and the idle interval is longer than the *break-even time* $\frac{E_{sw}}{P(f_{\min})}$, turning it to the dormant mode is worthwhile. Let t_θ be the break-even time, i.e., $t_\theta = \frac{E_{sw}}{P(f_{\min})}$. Without loss of generality, we assume that t_{sw} is no more than t_θ.

2.2 Task Models

A periodic task is an infinite sequence of task instances, where each task instance of a task comes in a regular period. We explore the scheduling of a periodic task τ in a leakage-aware DVS system, in which the period of the task is denoted by p. The relative deadline d of the task is equal to the period p, where extensions for the case that $d < p$ can be achieved easily. The amount of the worst-case execution cycles of the task is c. Executing at frequency f for t time units is assumed to complete $f \times t$ cycles with energy consumption $P(f)t$. Without loss of generality, we only focus on the case that $\frac{c}{f_{\max}} \leq p$, in which the task can complete before its period by executing at the highest frequency.



(a) Discrete PDF (b) Discrete CDF

Figure 1: An example for the probability functions of workload information of a task.

As shown in the literature [11,14,16], the probability information of computation requirement of task τ can be profiled as a discrete probability density function (PDF) of execution cycles. Suppose that the probability ψ_i that a task instance of the task completes its execution right with Y_i cycles is obtained by profiling, where $Y_1 < Y_2 < \cdots < Y_K = c$ and K is the number of Y_is with probability $\psi_i > 0$. For brevity, let Y_0 be 0. Hence, the range $(0, c]$ is divided into K bins with different sizes, where the j-th bin B_j is with $X_j = Y_j - Y_{j-1}$ amount of cycles. The discrete cumulative density function (CDF) for task τ to have cycles no more than Y_k is $\Psi_j = \sum_{k=1}^{j} \psi_k$. By definition, $\Psi_K = 1$. Figure 1 is an example of a task τ with $K = 6$, where $X_1 = X_2 = \cdots = X_K = 0.004f^*$ with $\psi_1 = 0.25$, $\psi_2 = 0.2$, $\psi_3 = 0.15$, $\psi_4 = 0.1$, $\psi_5 = 0.1$, and $\psi_4 = 0.2$. Figure 1(a) is the PDF, while Figure 1(b) is the CDF.

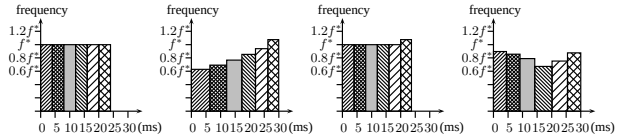
The probability that the schedule has to execute bin B_j is $1 - \Psi_{j-1}$, denoted by Ψ_j^*, where Ψ_0 is defined as 0 for boundary condition. Hence, for the example in Figure 1, $\Psi_1^* = 1$, $\Psi_2^* = 0.75$, $\Psi_3^* = 0.55$, $\Psi_4^* = 0.4$, $\Psi_5^* = 0.3$, and $\Psi_6^* = 0.2$.

2.3 Problem Definition and an Example

The problem studied in this paper is to *minimize the expected energy consumption* for the execution of a periodic real-time task on a DVS system with non-negligible leakage power consumption, where the task must be completed in time in the worst cases. As suggested in the literature [7, 9], we could first minimize the dynamic energy consumption by treating the critical frequency as the minimum available frequency, and then turn the processor to the dormant mode once the available idle interval to the beginning of the next task instance is longer than the break-even time.

With the ignoring of the static power consumption, we can adopt the accelerating-frequency (denoted by AF) strategy in [11, 16, 17]. Then, if the derived solution executes some bins at frequencies less than the critical frequency, the frequencies of these bins are promoted to the critical frequency, in which this approach is denoted by *Accelerating-Frequency with Critical Frequency* (AFCF). Since frequency promotion would create some slack time, we can then apply the above procedure repeatedly until all the frequencies in the solution are no less than the critical frequency, in which this approach is denoted as *Repeatedly Accelerating-Frequency with Critical Frequency* (RAFCF). Another approach, denoted as *Constant-Frequency with Critical Frequency* (CFCF), runs at a constant frequency $\max\{\frac{c}{p}, f^*\}$.

The above approaches sound reasonable, but cannot guarantee to derive an optimal frequency assignment. Consider to schedule a



(a) CFCF and RAFCF (b) AF (c) AFCF (d) Optimal

Figure 2: A motivational example.

task with period (relative deadline) equal to 30 milliseconds on an Intel XScale processor with power consumption function $P(f) = 1520f^3 + 80$ mWatt [16], minimum frequency $f_{\min} = 150$MHz, and energy switching overhead 1 mJoule, where f is with the unit of GHz and the critical frequency f^* is, hence, about 297MHz. The probability density function and the size of the bins of the task are illustrated in Figure 1(a). Suppose that the scheduler would turn the processor to the dormant mode once the deadline subtract the completion time is more than the break-even time (11.75 milliseconds) and the processor is turned to the active mode for operation right at the beginning of the next period. Executing the task by using CFCF would lead to a solution to execute every bin at the critical frequency f^*, as shown in Figure 2(a), with expected system energy consumption $(\sum_{\ell=1}^{4} \Psi_\ell^* P(f^*) \times 0.004 + \psi_\ell E_{sw}) + (\Psi_5^* P(f^*) \times 0.004 + \psi_5 P(f_{\min}) \times 0.01) + (\Psi_6^* P(f^*) \times 0.004 + \psi_6 P(f_{\min}) \times 0.006)$, which is 2.423mJ. Applying AF would lead to a solution, as shown in Figure 2(b), which executes the bins at frequencies $0.630f^*, 0.693f^*, 0.768f^*, 0.854f^*, 0.940f^*$, and $1.076f^*$ where the expected system energy is 2.395mJ. Because the frequencies of the first five bins are lower than the critical frequency, AFCF assigns these five bins to execute at the critical frequency, as shown in Figure 2(c), with expected system energy consumption 2.429mJ. Applying RAFCF will execute the sixth bin at the critical frequency, in which the solution is the same as CFCF.

However, executing the six bins at frequencies $0.898f^*, 0.857f^*, 0.791f^*, 0.673f^*, 0.754f^*$, and $0.877f^*$, as shown in Figure 2(d), leads to a solution with 2.326mJ. The reason why AF is not optimal comes from the ignorance of the static power consumption. The reason why CFCF, AFCF, and RAFCF do not derive optimal solutions is due to the ignorance of energy switching overhead E_{sw} and the static power consumption during execution. To minimize the static energy consumption, we would, at the beginning, execute at higher frequencies. For example, the first four bins in Figure 2(d) are executed at decelerating frequencies, while the last three bin bins are executed with accelerating frequencies to minimize the dynamic energy consumption.

3. EXPECTED ENERGY MINIMIZATION

This section provides our algorithms for the minimization of the expected system energy consumption. We first focus on the case that the processor is in the active mode at the beginning of every task instance. Then, we will show how to use the procrastination approach to procrastinate the execution of the next task instance to lengthen the idle interval for energy saving.

3.1 Processor is Active at the Beginning

This subsection considers the case that the processor is always in the active mode at the beginning of any task instance. Therefore, an optimal schedule would execute each task instance right after it is released without turning the processor to the dormant mode before it completes. For an optimal frequency assignment of these K bins of task τ, there must a κ^* with $0 \leq \kappa^* \leq K$, in which (1) the processor is turned to the dormant mode if the task instance completes after executing any of the first κ^* bins and (2) the processor is idle in the active mode if the task instance completes after executing any of the last $K - \kappa^*$ bins. Suppose that the execution time of bin B_i in the optimal solution is t_i^*. Therefore, the expected system energy consumption for the first κ^* bins is $\left(\sum_{\ell=1}^{\kappa^*} \Psi_\ell^* P(\frac{X_\ell}{t_\ell^*}) t_\ell^* + \psi_\ell E_{sw}\right)$, where the first term is the expected energy consumption for execution and the second is that for energy switching overhead. Since the processor is going to be idle in the active mode after completing any of the last $K - \kappa^*$ bins, the expected system energy consumption for the last $K - \kappa^*$ bins is $\left(\sum_{\ell=\kappa^*+1}^{K} \Psi_\ell^* P(\frac{X_\ell}{t_\ell^*}) t_\ell^* + \psi_\ell (p - \sum_{j=1}^{\ell} t_j^*) P(f_{\min})\right)$.

Suppose that we are given a specified non-negative integer $\kappa \leq K$. If there is an algorithm that can derive an optimal solution under the constraint that the completion time of bin B_κ is no later than $p - t_\theta$ and the completion time of bin $B_{\kappa+1}$ (if exists) is later than $p - t_\theta$, we can just apply the algorithm for $\kappa = 0, 1, 2, \ldots, K$ and return the solution with the minimum expected system energy consumption as the optimal solution. Here, instead of finding the optimal frequency assignment for a specified κ, we find a lower bound by relaxing the constraints so that the completion times of bins B_κ and $B_{\kappa+1}$ are not constrained. Let t_ℓ be the execution time of bin B_ℓ and T_K be a non-negative variable for the slack before the deadline after completing bin B_K. That is, T_K is $p - \sum_{\ell=1}^{K} t_\ell$. The optimization of the lower bound of the expected system energy consumption under a specified κ can formulated as follows:

$$\text{minimize} \quad \left(\sum_{\ell=1}^{K} \Psi_\ell^* P(\tfrac{X_\ell}{t_\ell}) t_\ell\right) + \left(\sum_{\ell=1}^{\kappa} \psi_\ell E_{sw}\right) + \sum_{\ell=\kappa+1}^{K} \psi_\ell P(f_{\min})(T_K + \sum_{j=\ell+1}^{K} t_j) \tag{1a}$$

subject to

$$T_K + \sum_{\ell=1}^{K} t_\ell = p, \tag{1b}$$

$$\frac{X_\ell}{f_{\max}} \leq t_\ell \leq \frac{X_\ell}{f_{\min}}, \ \forall \ell = 1, 2, \ldots, K \text{ and} \tag{1c}$$

$$T_K \geq 0. \tag{1d}$$

LEMMA 1. *For a given κ with $f_{\min} > 0$, if the optimal solution of Equation (1) is with $T_K > 0$, all the last $K - \kappa$ bins are executed at frequency $f_{\min}$ in the optimal solution. If $f_{\min}$ is 0, T_K is 0 for any $\kappa < K$.*

PROOF. This lemma can be proved by contradiction. If $T_K > 0$ and there is a bin B_ℓ with $\ell > \kappa$ executed at a higher frequency f_ℓ than $f_{\min}$, executing bin B_ℓ at frequency $f_{\min}$ or $\frac{X_\ell}{T_K + \frac{X_\ell}{f_\ell}}$ has a smaller value of the objective function of Equation (1). □

For a specified κ, let $(t_1^{*,\kappa}, t_2^{*,\kappa}, \ldots, t_K^{*,\kappa})$ be the optimal solution of Equation (1). The following lemmas indicate that finding the optimal solution for Equation (1) does not lose any precision to derive an optimal frequency assignment to minimize the expected system energy consumption of task τ.

LEMMA 2. *Suppose that $\sum_{\ell=1}^{\kappa^\flat} t_\ell^{*,\kappa^\flat} > p - t_\theta$ for some $\kappa^\flat \geq 1$, the optimal solution of Equation (1) by setting κ as $\kappa^\flat - 1$ is less than the optimal solution when κ is $\kappa^\flat$.*

LEMMA 3. *Suppose that $\sum_{\ell=1}^{\kappa^\flat+1} t_\ell^{*,\kappa^\flat} < p - t_\theta$ for some $\kappa^\flat < K$, the optimal solution of Equation (1) by setting κ as $\kappa^\flat + 1$ is less the optimal solution when κ is $\kappa^\flat$.*

PROOF. The above lemmas can be proved by showing that the optimal solution of Equation (1) by setting κ as $\kappa^\flat$ is more in

the objective function of Equation (1) than setting κ as $\kappa^\flat - 1$ in Lemma 2 or setting κ as $\kappa^\flat + 1$ in Lemma 3. □

LEMMA 4. *If the optimal solution of Equation (1) is the minimum by setting κ as $\kappa^\dagger$, then $\sum_{\ell=1}^{\kappa^\dagger} t_\ell^{*,\kappa^\flat} \leq p - t_\theta$ for any $\kappa^\dagger$ and $\sum_{\ell=1}^{\kappa^\dagger+1} t_\ell^{*,\kappa^\flat} \geq p - t_\theta$ for any $\kappa^\dagger < K$.*

PROOF. Applying Lemmas 2 and 3 can prove this lemma. □

Therefore, if there is an algorithm which can derive an optimal solution for Equation (1), we can just apply the algorithm for $\kappa = 0, 1, 2, \ldots, K$ and return the solution with the minimum expected system energy consumption as the optimal solution. Consider the case that κ is K. If executing all the bins at the critical frequency can make the task instance complete before its deadline, i.e., $\sum_{\ell=1}^{K} \frac{X_\ell}{f^*} \leq p$, executing all the bins at the critical frequency is optimal for Equation (1). Otherwise, κ^* must be less than K.

For any $\kappa < K$, we will first derive solutions for unconstrained frequencies, and then, revise to satisfy the frequency constraints.

3.1.1 Unconstrained Frequencies

We begin to solve the convex programming in Equation (1) with relaxation on the frequency constraints by setting $f_{\min}$ as 0 and $f_{\max}$ as ∞. By Lemma 1, for any non-negative $\kappa < K$, T_K must be 0 in optimal solutions. Since $\sum_{\ell=1}^{\kappa} \psi_\ell E_{sw}$ is a constant regardless of the variable t_ℓs, we can ignore the constant during optimization. The relaxation of Equation (1) for $0 \leq \kappa < K$ is as follows:

$$\text{minimize } \left(\sum_{\ell=1}^{K} \Psi_\ell^* P(\frac{X_\ell}{t_\ell}) t_\ell \right) + \sum_{\ell=\kappa+1}^{K} \psi_\ell P(f_{\min})(\sum_{j=\ell+1}^{K} t_j) \quad (2a)$$

$$\text{subject to } \sum_{\ell=1}^{K} t_\ell = p, \text{ and} \quad (2b)$$

$$t_\ell \geq 0, \ \forall \ell = 1, 2, \ldots, K. \quad (2c)$$

The convex programming in Equation (2) can be solved by applying the Lagrange Multiplier Method. For notational brevity, let

$$F_{\kappa,\ell}(t_\ell) = \begin{cases} \Psi_\ell^* P(\frac{X_\ell}{t_\ell}) t_\ell & \ell \leq \kappa \text{ and} \\ \Psi_\ell^* P(\frac{X_\ell}{t_\ell}) t_\ell + P(f_{\min}) t_\ell \sum_{j=\kappa+1}^{\ell-1} \psi_j & \text{otherwise.} \end{cases}$$

The Lagrange Multiplier Method is to find a constant λ so that $\sum_{\ell=1}^{K} t_\ell = p$ and the first derivative $F'_{\kappa,\ell}(t_\ell)$ of function $F_{\kappa,\ell}(t_\ell)$ is λ for $\ell = 1, 2, \ldots, K$. Take $P(f) = \alpha f^3 + \beta$ with $f_{\min} = 0$ for example. Our objective is to solve the following equation:

$$\Psi_\ell^*(-2\alpha \frac{X_\ell^3}{t_\ell^3} + \beta) = \lambda, \qquad \forall \ell \leq \kappa, \quad (3a)$$

$$\Psi_\ell^*(-2\alpha \frac{X_\ell^3}{t_\ell^3} + \beta(1 + \sum_{j=\kappa+1}^{\ell-1} \psi_j)) = \lambda, \quad \forall \ell > \kappa, \text{ and} \quad (3b)$$

$$\sum_{\ell=1}^{K} t_\ell = p. \quad (3c)$$

Finding a constant λ satisfying the above constraints can be solved efficiently by applying Newton's method or the bisection method.

3.1.2 Constrained Frequencies

With frequency constraints, there are two cases in which T_K is 0 or T_K is larger than 0. By Lemma 1, if the optimal solution is with $T_K > 0$ for a specified κ, all the last $K - \kappa$ bins are executed at frequency $f_{\min}$ and every bin B_ℓ with $\ell \leq \kappa$ is with execution time t_ℓ so that $F'_{\kappa,\ell}(t_\ell)$ is $\sum_{i=\kappa+1}^{K} \psi_i P(f_{\min})$, where $F'_{\kappa,\ell}(t_\ell)$ is the derivative of $F_{\kappa,\ell}(t_\ell)$. If t_ℓ with $F'_{\kappa,\ell}(t_\ell) = \sum_{i=\kappa+1}^{K} \psi_i P(f_{\min})$ is larger (less, respectively) than the execution time at frequency $f_{\min}$ ($f_{\max}$, respectively), the execution time of bin B_ℓ is set to $\frac{X_\ell}{f_{\min}}$ ($\frac{X_\ell}{f_{\max}}$, respectively). If the summation of the execution time of these K bins above is more than period p, it is clear that the optimal solution for a specified κ is with $T_K = 0$.

Now, we focus on the case that T_K is 0. For a specified $\kappa < K$, the optimal solution for Equation (1) can be derived by applying the Karush-Kuhn-Tucker optimality condition [13, §14]. First, we obtain an optimal solution by assuming that there is no frequency constraint as in Section 3.1.1. If there is no frequency violation, the solution is also an optimal solution for Equation (1). However, if the execution time t_ℓ^κ for some bin is larger than $\frac{X_\ell}{f_{\min}}$, we force some bin to execute at frequency $f_{\min}$ and form a sub-problem. Moreover, if the execution time t_ℓ^κ for some bin is less than $\frac{X_\ell}{f_{\max}}$, we fix some bin to execute at frequency $f_{\max}$ and form a sub-problem. We then solve the sub-problems repeatedly.

Let $\mathbf{B}$ be the set of the K bins. Let sets $\mathbf{B}_{\max}$ and $\mathbf{B}_{\min}$ consist of the bins B_ℓs, in which t_ℓ^κ are restricted to $\frac{X_\ell}{f_{\max}}$ and $\frac{X_\ell}{f_{\min}}$, respectively. Initially, sets $\mathbf{B}_{\max}$ and $\mathbf{B}_{\min}$ are both empty sets. After solving Equation (2) for a specified κ, if there is a bin B_ℓ in set $(\mathbf{B} \setminus \mathbf{B}_{\max} \setminus \mathbf{B}_{\min})$ with $t_\ell^\kappa < \frac{X_\ell}{f_{\max}}$, we greedily restrict the bin B_{ℓ^*} in $(\mathbf{B} \setminus \mathbf{B}_{\max} \setminus \mathbf{B}_{\min})$ with the largest $F'_{\kappa,\ell^*}(\frac{X_{\ell^*}}{f_{\max}})$ to execute at frequency $f_{\max}$ and move B_{ℓ^*} from $\mathbf{B}$ to $\mathbf{B}_{\max}$. We repeatedly solve Equation (2) until all the t_ℓs in the optimal solution use frequencies no more than $f_{\max}$. Then, if all the bins are executed with frequencies no less than $f_{\min}$, we return the solution. Otherwise, there is a bin B_ℓ in $(\mathbf{B} \setminus \mathbf{B}_{\max} \setminus \mathbf{B}_{\min})$ with $t_\ell^\kappa > \frac{X_\ell}{f_{\min}}$, and we greedily restrict the bin B_{ℓ^*} in $(\mathbf{B} \setminus \mathbf{B}_{\max} \setminus \mathbf{B}_{\min})$ with the smallest $F'_{\kappa,\ell^*}(\frac{X_{\ell^*}}{f_{\min}})$ to execute at frequency $f_{\min}$, move B_{ℓ^*} from $\mathbf{B}$ to $\mathbf{B}_{\min}$, and solve the formed sub-problem repeatedly.

Algorithm 1 illustrates the pseudo-code of the above procedures. The time complexity is $O(K^2(K + L))$, where $O(L)$ is the time complexity for deriving the optimal solution of Equation (2) with the bisection method or Newton's method. Based on the Karush-Kuhn-Tucker optimality condition [13, §14], Algorithm 1 can be proved to derive optimal solutions of Equation (1) for a given κ.

THEOREM 1. *Algorithm 1 derives an optimal solution for Equation (1) for a specified $\kappa < K$.*

PROOF. This can be proved by using the Karush-Kuhn-Tucker optimality condition. See [13] for reference. □

3.1.3 Overall Algorithm

We now can use Algorithm 1 to derive an optimal solution of Equation (1) for a specified κ. The algorithm, denoted by Algorithm STATIC, to minimize the expected system energy consumption is to apply Algorithm 1 for $\kappa = 0, 1, \ldots, K$ and return the best among these derived solutions. Based on the argument of the optimality when $\kappa = K$, Lemma 4, and Theorem 1, the derived solution is optimal. The time complexity is $O(K^3(K + L))$, where $O(L)$ is for deriving optimal solutions of Equation (2). Note that Algorithm STATIC only has to be performed once for deriving the execution frequencies. Hence, the time complexity is acceptable.

3.2 Processor is Dormant at the Beginning

Section 3.1 focuses on the case that the processor is always in the active mode at the beginning of a task instance. However, if the processor is turned to the dormant mode after completing the previous task instance, we can procrastinate the execution of the current

Algorithm 1 : κ-STATIC

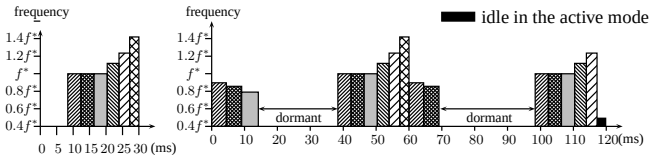
Input: $\kappa, p, K, \{B_1, B_2, \ldots, B_K\}, f_{\max}, f_{\min}, P()$;

find $f_{\min} \le f^* \le f_{\max}$ in which $\frac{P(f^*)}{f^*}$ is minimized;
//$\kappa = K$
if $\kappa = K$ and $\sum_{\ell=1}^{K} \frac{X_\ell}{f^*} \le p$ **then**
  return $(\frac{X_1}{f^*}, \frac{X_2}{f^*}, \ldots, \frac{X_K}{f^*})$;
else if $\kappa = K$ **then**
  return message "setting κ as K is not optimal";
//$T_K > 0$
for each bin B_ℓ with $\ell \le \kappa$ **do**
  find $t_\ell^\flat$ so that $F'_{\kappa,\ell}(t_\ell^\flat)$ is $\sum_{i=\kappa+1}^{K} \psi_i P(f_{\min})$ for $\ell \le \kappa$;
  $t_\ell^\flat \leftarrow \min\left\{\max\{t_\ell^\flat, \frac{X_\ell}{f_{\max}}\}, \frac{X_\ell}{f_{\min}}\right\}$;
$t_\ell^\flat \leftarrow \frac{X_\ell}{f_{\min}}$ for every $\kappa < \ell \le K$;
//$T_K = 0$
$\mathbf{B} \leftarrow \{B_\ell \mid 1 \le \ell \le K\}, \mathbf{B}_{\min} \leftarrow \emptyset, \mathbf{B}_{\max} \leftarrow \emptyset$;
while true **do**
  while true **do**
    solve Equation (2) by applying the Lagrange Multiplier Method with the restriction to execute bins in set $\mathbf{B}_{\min}$ at frequency $f_{\min}$ and bins in set $\mathbf{B}_{\max}$ at frequency $f_{\max}$;
    let $(t_1^{*,\kappa}, t_2^{*,\kappa}, \ldots, t_K^{*,\kappa})$ be the solution derived by Step 13;
    if $t_\ell^{*,\kappa} \ge \frac{X_\ell}{f_{\max}}, \forall 1 \le \ell \le K$ **then**
      break;
    else
      let bin B_{ℓ^*} be the bin B_ℓ in $\mathbf{B}$ with the largest $F'_{\kappa,\ell}(\frac{X_\ell}{f_{\max}})$;
      $\mathbf{B} \leftarrow \mathbf{B} \setminus \{B_{\ell^*}\}, \mathbf{B}_{\max} \leftarrow \mathbf{B}_{\max} \cup \{B_{\ell^*}\}$;
  if $t_\ell^{*,\kappa} \le \frac{X_\ell}{f_{\min}}, \forall 1 \le \ell \le K$ **then**
    break;
  else
    let bin B_{ℓ^*} be the bin B_ℓ in $\mathbf{B}$ with the smallest $F'_{\kappa,\ell}(\frac{X_\ell}{f_{\min}})$;
    $\mathbf{B} \leftarrow \mathbf{B} \setminus \{B_{\ell^*}\}, \mathbf{B}_{\min} \leftarrow \mathbf{B}_{\min} \cup \{B_{\ell^*}\}$;
//**Compare the solutions when** $T_K > 0$ **and** $T_K = 0$
if $\sum_{\ell=1}^{K} t_\ell^\flat > p$ **then**
  return $(t_1^{*,\kappa}, t_2^{*,\kappa}, \ldots, t_K^{*,\kappa})$;
else
  return the better solution between $(t_1^{*,\kappa}, t_2^{*,\kappa}, \ldots, t_K^{*,\kappa})$ and $(t_1^\flat, t_2^\flat, \ldots, t_K^\flat)$;

task instance to utilize the energy saving in the dormant mode. That is, if the worst-case execution time of the task is $w < p$, we would let the processor stay in the dormant mode until the current task instance has been released for $p - w$ time units so that the energy consumption in the $p - w$ time units for procrastination is 0. With a specified κ, we would like to derive a frequency assignment so that the expected energy consumption for execution and idling (in the active mode) is minimized for each bin. The expected energy consumption by setting the execution time of bin B_ℓ as t_ℓ is $F_{\kappa,\ell}(t_\ell)$. For any ℓ no more than κ, function $F_{\kappa,\ell}(t_\ell)$ is minimized when bin B_ℓ is executed at the critical frequency f^*. For any ℓ larger than κ, function $F_{\kappa,\ell}(t_\ell)$ is minimized when t_ℓ is set as $\max\{\frac{X_\ell}{f_{\max}}, t_\ell^\dagger\}$, where $t_\ell^\dagger$ is the root of $F'_{\kappa,\ell}(t_\ell^\dagger) = 0$ and $F'_{\kappa,\ell}(t_\ell)$ is the derivative of function $F_{\kappa,\ell}(t_\ell)$. For example, when $P(f) = \alpha f^3 + \beta$, $t_\ell^\dagger$ is $\max\left\{\frac{X_\ell}{f_{\max}}, X_\ell \sqrt[3]{\frac{2\alpha\Psi_\ell^*}{\beta\Psi_\ell^* + (\alpha f_{\min}^3 + \beta)\sum_{j=\kappa+1}^{\ell-1}\psi_j}}\right\}$.

Let $t_\ell^{\dagger,\kappa}$ be the value t_ℓ which minimizes function $F_{\kappa,\ell}(t_\ell)$ without violating frequency constraints. Then, we just have to find $\kappa^\dagger$ so that $\sum_{\ell=1}^{K} t_\ell^{\dagger,\kappa} \le p$ and $\left(\sum_{\ell=1}^{\kappa^\dagger} \psi_\ell E_{sw}\right) + \left(\sum_{\ell=1}^{K} F_{\kappa^\dagger,\ell}(t_{\ell,}^{\dagger,\kappa^\dagger})\right)$ is minimized. Then, once the processor completes a task instance and decides to enter the dormant mode, the next task instance will start its execution after it is released for $p - \sum_{\ell=1}^{K} t_\ell^{\dagger,\kappa}$ time units with the above frequency assignment.

Take the example in Section 2.3 for demonstration, in which $\kappa^\dagger$ is 2. The minimum expected energy consumption for procrastination is 2.208mJ by executing the first three bins at the critical frequency f^* and the last three bins at frequencies $1.119f^*$, $1.236f^*$, and $1.420f^*$ with worst-case execution time 21.631 milliseconds. Figure 3(a) illustrates the resulting frequency assignment. Suppose that the task completes right after executing the third bin for the first instance, and right after the sixth bin for the second instance, the second bin for the third instance, and the fifth bin for the fourth instance. Figure 3(b) shows the actual schedule by applying the procrastination procedure.



(a) Optimal (b) An actual schedule

Figure 3: A procrastination scheduling example.

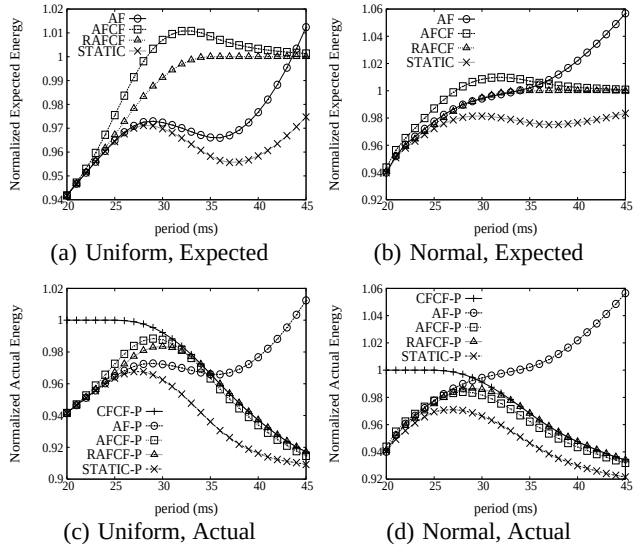
4. EXPERIMENTS

This section provides experimental results on Intel XScale, in which $P(f)$ is approximately modeled as $1520f^3 + 80$ mWatt [4, 16] with the assumption that $f_{\min} = 0$. When procrastination is needed, the parametric procrastination strategy in [4] is adopted by setting the parameter α in [4] as 0.5. Algorithms CFCF-P, AF-P, AFCF-P, and RAFCF-P are the corresponding algorithms with procrastination scheduling, while the strategy in Section 3.2 is only applied for Algorithm STATIC-P.

We perform evaluations for synthetic real-time tasks with *normal* and *uniform* discrete probability density functions as suggested in [17, 18]. The amount of worst-case execution cycles is a random variable in the range of $[0.025f^*, 0.035f^*]$. For a specified amount c of the worst-case execution cycles of a task τ, the probability density function for a normal distribution is generated by setting the mean as a random variable in the range of $[\frac{2c}{6}, \frac{4c}{6}]$ and the standard deviation as $\frac{c}{6}$. For a uniform distribution, the probability density function is $\frac{1}{c}$. The number of bins of a synthetic task is an integral random variable in the range of $[10, 20]$, in which each bin is with the same size. The probability ψ_k of bin B_k is the accumulative probability of the bin in the generated probability distribution.

The normalized expected energy consumption of an algorithm is defined as the expected energy consumption of a task instance derived from the algorithm divided by that from Algorithm CFCF. The normalized actual energy consumption of an algorithm is defined as the energy consumption derived from the algorithm divided by that from Algorithm CFCF without applying procrastination. In our experiments, each configuration is done with 256 independent settings, and their average value is reported. The actual energy consumption is derived by running 10000 task instances.

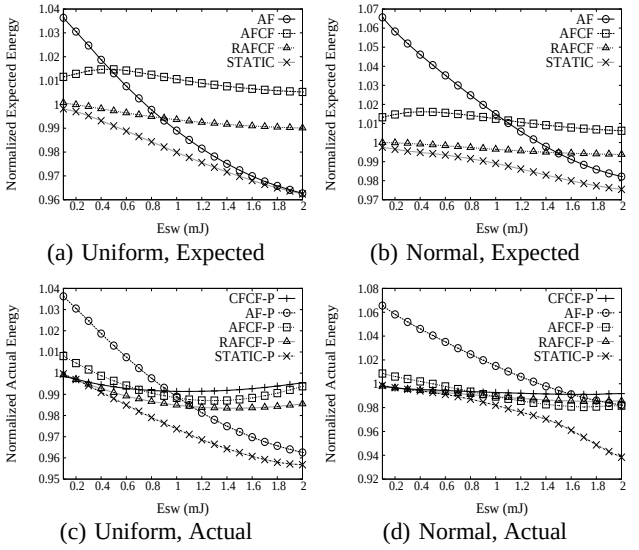
Figure 4 is the experimental results by varying the period from 20 milliseconds to 45 milliseconds when E_{sw} is 1.5mJ. Figures 4(a) and 4(b) are for the normalized expected energy consumption of the evaluated algorithms, and Figures 4(c) and 4(d) are for the normalized actual energy consumption. When the period is small, there is not too much room to turn the processor to the dormant mode. Therefore, in Figures 4(a) and 4(b), when task period is less than 25 milliseconds, the variation of accelerating strategies, including Algorithms AF, AFCF, and RAFCF, could derive solutions similar to Algorithm STATIC, in which all of them outperform Algorithm CFCF since the task is executed at the constant (critical) frequency to turn the processor off. The reason why Algorithms AFCF and RAFCF might be worse than Algorithm AF is because they increase the execution frequency to try to turn the processor off but the processor remains active in the actual schedule. For Algorithm AF, procrastination scheduling does not help at all since



(a) Uniform, Expected (b) Normal, Expected

(c) Uniform, Actual (d) Normal, Actual

Figure 4: Experimental results by varying periods of tasks with E_{sw} equal to 1.5 mJ.



(a) Uniform, Expected (b) Normal, Expected

(c) Uniform, Actual (d) Normal, Actual

Figure 5: Experimental results by varying E_{sw} from 0.1 mJ to 2 mJ with period equal to 30 milliseconds.

each task instance is with worst-case execution time equal to the period.

Figure 5 is the experimental results by varying the energy switching overhead from 0.1mJ to 2mJ when the period is 30 milliseconds. Figures 5(a) and 5(b) are for the normalized expected energy consumption of the evaluated algorithms, and Figures 5(c) and 5(d) are for the normalized actual energy consumption. Similarly, when E_{sw} is small, we have more room to turn the processor to the dormant mode, and, hence, Algorithm CFCF outperforms the variation of the accelerating strategies, while Algorithm STATIC is still the best. When E_{sw} is large enough ($E_{sw} \geq$ 1mJ in Figure 5(a) and $E_{sw} \geq 1.5$mJ in Figure 5(b)), the processor has less room to be turned to the dormant mode, and, again, Algorithm AF becomes better than Algorithms CFCF, AFCF, and RAFCF. With procrastination scheduling, Algorithm STATIC-P can minimize the energy consumption than all the other evaluated algorithms as shown in Figures 5(c) and 5(d).

5. CONCLUSION

This paper presents effective approaches to minimize the expected energy consumption for a real-time task with uncertain execution time in leakage-aware DVS systems. The simple extensions of existing algorithms by taking the critical frequency as the minimum execution frequency are shown not optimal. This paper develops an algorithm to derive an optimal frequency assignment to minimize the expected system energy consumption under the timing constraint. Then, we adopt the procrastination scheduling strategy for further energy reduction if the processor is decided to be turned to the dormant mode. Experiments show that our algorithms can effectively minimize the expected energy consumption.

It is not difficult to extend the approaches in this paper to systems with multiple power-saving modes. The proposed algorithms can also be used for multiple periodic real-time tasks by considering that these tasks are with joint workload, such as the approaches in [17]. However, whether the proposed algorithms in this paper are adaptable to the approaches in [16, 18] for inter-task scheduling is unknown since we cannot find an exact function to express the (optimal) expected system energy consumption for a given period.

References

[1] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium*, 2001.

[2] J.-J. Chen. Expected energy consumption minimization in DVS systems with discrete frequencies. In *SAC*, 2008.

[3] J.-J. Chen and T.-W. Kuo. Procrastination for leakage-aware rate-monotonic scheduling on a dynamic voltage scaling processor. In *ACM Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)*, 2006.

[4] J.-J. Chen and T.-W. Kuo. Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems. In *ICCAD*, 2007.

[5] F. Gruian. Hard real-time scheduling for low-energy using stochastic data and DVS processors. In *ISLPED*, 2001.

[6] F. Gruian and K. Kuchcinski. Uncertainty-based scheduling: energy-efficient ordering for tasks with variable execution time. In *ISLPED*, 2003.

[7] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2003.

[8] T. Ishihara and H. Yasuura. Voltage scheduling problems for dynamically variable voltage processors. In *Proceedings of the International Symposium on Low Power Electronics and Design*, 1998.

[9] R. Jejurikar, C. Pereira, and R. Gupta. Leakage aware dynamic voltage scaling for real-time embedded systems. In *Proceedings of the Design Automation Conference*, 2004.

[10] Y.-H. Lee, K. P. Reddy, and C. M. Krishna. Scheduling techniques for reducing leakage power in hard real-time systems. In *15th Euromicro Conference on Real-Time Systems (ECRTS)*, 2003.

[11] J. R. Lorch and A. J. Smith. Pace: A new approach to dynamic voltage scaling. *IEEE Trans. Computers*, 53(7):856–869, 2004.

[12] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *ACM SOSP*, 2001.

[13] R. L. Rardin. *Optimization in Operations Research*. Prentice Hall, 1998.

[14] C. Xian and Y.-H. Lu. Dynamic voltage scaling for multitasking real-time systems with uncertain execution time. In *ACM GLS-VLSI*, 2006.

[15] R. Xu, R. G. Melhem, and D. Mossé. A unified practical approach to stochastic DVS scheduling. In *EMSOFT*, 2007.

[16] R. Xu, D. Mossé, and R. G. Melhem. Minimizing expected energy in real-time embedded systems. In *EMSOFT*, 2005.

[17] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. In *SOSP*, 2003.

[18] Y. Zhang, Z. Lu, J. Lach, K. Skadron, and M. R. Stan. Optimal procrastinating voltage scheduling for hard real-time systems. In *DAC*, 2005.