Minimum Energy Consumption for Rate Monotonic Tasks

Sanjeev Baskiyar · Chiao Ching Huang · Tin-Yau Tam

Abstract Limited battery power is a typical constraint in stand-alone embedded systems. One way to extend the battery lifetime is by reducing CPU power consumption. Because of the quadratic relationship between power consumption in CMOS circuits and CPU voltage, power reduction can be obtained by scaling down supply voltage, or Dynamic Voltage Scaling. However, reducing supply voltage slows down CPU speed since supply voltage has a proportional relationship with CPU frequency. On the other hand, in any real-time embedded environment (especially hard real-time), timing constraints are critical.

In this paper, we focus on dynamic energy reduction of tasks scheduled by Rate Monotonic (RM) algorithm in a hard real-time embedded environment. The RM algorithm preemptively schedules any set of periodic tasks by assigning higher priorities to frequent tasks. For any periodic task set that satisfies the CPU utilization bound, we determine the provably optimal scaling of the worst-case execution time of each task that consumes minimum dynamic energy while satisfying the utilization bound. As RM algorithm is widely used, we expect this work can lead to better energy reduction management and expectations.

Keywords CPU dynamic power consumption · Lagrange multiplier · Periodic tasks · DVFS

1 Introduction

Maturity of wireless technology and mobile markets have dramatically changed lifestyles. More and more consumers depend on wireless or portable devices, such as cellular phone, personal digital assistant (PDA), digital camera or global positioning system (GPS). With the greatly growing popularity of handheld devices, energy consumption has always been one of the major determinants in the purchase decision. For decades, researchers have sought to find

baskisa@auburn.edu; huangcc@gmail.com; tamtiny@auburn.edu

the optimal solution to save power in embedded systems and produce more energy efficient power supplies.

As far as Real-Time Embedded Systems are concerned, power consumption becomes an even bigger issue. One of the well-known characteristics of a Real-Time Operating System is its time constraint. In a Soft Real-Time System, missed deadlines may not impact the system's functionality. However, a missed deadline in a Hard Real-Time System could lead to unrecoverable system failure. Reducing power dissipation often means sacrificing system performance; tasks may complete later than their deadlines which cannot be tolerated in a Hard Real-Time system [14]. Therefore, energy reduction must be balanced with deadline guarantees.

There are three major parts of CPU power dissipation: dynamic power, static power and the leakage power. The dynamic power is related to logic switching during computation. During switching many transistors may momentarily conduct current simultaneously which can provide a short-circuit path from source to ground. The related power is called the static power. Very small currents are always present between differently doped parts of the transistor. These currents give rise to leakage power and are higher with decreasing transistor size and increasing temperatures. All these powers are related to the supply voltage. However, the dynamic and static power are also related to the frequency of operation. In this paper, we focus mainly on the dynamic power consumption.

This study focuses on Rate Monotonic (RM) schedulable tasks for Hard Real-Time Embedded Systems. The task model is preemptive and static priority driven. The approach is to expand computation time of each task using Dynamic Voltage Scaling (DVS) until the aggregate CPU utilization of all tasks approximately reaches the upper bound of Rate Monotonic schedulability test (RM test) [10]; thus, lowering CPU speed and supplied voltage, yet guaranteeing no task misses its deadline. With the method of Lagrange Multiplier, the approach deduces a specific formula and iteration yielding an exact scaling factor for each task (the scaling factor is usually different for different tasks). Then, the computation time of each task is increased by applying the corresponding scaling factor. Consequently, the minimum power consumption can be achieved and all tasks will be processed within their time constraints.

The remainder of this study is organized as follows. Section 2 reviews recent studies. Specifically, different kinds of DVS approaches and their challenges have been introduced and analyzed. Section 3 provides a detailed explanation for the formulation and deduction of the problem. In Section 4, we provide a complete proof and present the task scaling algorithm inferred from this study. Section 5 includes some experiments and observations during this research and discussion of the result and Section 6 concludes with suggestions for future work.

2 Related Work

Substantial research has been conducted in the last decade towards energy reduction in computation. In this section we introduce DVS techniques and present some of its strategies. Before we start, we explain a few terms related to scheduling Real-Time tasks. *Worst Case Execution Time* (WCET) is the estimated maximum possible run-time of a task. In scheduling Hard Real-Time tasks, to guarantee the timing constraints, WCET is used [15].

For periodic tasks, *slack time* is defined as the time between the completion of the execution of a task and its next invocation. DVS achieves energy savings by stretching execution of tasks by CPU frequency reduction (and supply voltage reduction) during the execution of the task to partially or completely fill the slack.

Slack times can be classified into static slack times and dynamic slack times [8]. The static slack time is obtained by considering the WCET of the task whereas the dynamic slack time is obtained by considering the actual execution time of the task. Off-line (or static) approaches, e.g. static voltage scaling, path based method, stochastic method and maximum constant speed utilize known slack times in advance by reducing CPU frequency and consequently reducing supply voltage [15].

The actual task execution time is unpredictable because of conditionals, I/O wait times or processor related unpredictables such as pipelining, superscalar execution or wait states. Since tasks could finish earlier than their WCETs [5], on-line approaches have a greater opportunity for energy reduction, however such approaches themselves incur overhead. On-line approaches such as stretching to next-task-arrival (NTA), priority-based slack stealing and utilization updating, exploit this characteristic by reclaiming dynamic slack times and recompute CPU frequency at task release time to lower power consumption.

This paper addresses the static scheduling approach. The approach in this paper provides a *proven* minimal dynamic energy consumption when using the static approach and dynamic approaches can be used over and above our static approach.

2.1 Power-Down Modes

Power-down strategy is a traditional way to conserve power supply. Conventional shutdown approaches go to reduced power mode after the system has been idle for a few seconds/minutes. Predictive shutdown approaches calculates the upcoming idle time, which is static slack time, before tasks run and power off when the idle interval justifies the cost [17]. Usually, power-down mode runs tasks at its full speed and turns off the power supply while CPU is in the idle state.

2.2 Dynamic Voltage Scaling (DVS)

Dynamic Voltage Scaling (DVS) is based on two concepts and has been a mainstream in the area of power reduction. One concept is the quadratic relationship between power dissipation and supply voltage ($p \propto V^2$). Another concept for using DVS is that even in Hard Real-Time environments, as long as tasks can complete on time, a lower system throughput may be sufficient.

2.3 CMOS Circuit Design

Chandrakasan et al. [2] came up with an idea of redesigning system architecture by delaying CMOS circuit behaviors as much as possible while using the lowest possible supply voltage. This architecture-driven voltage scaling technique is capable of achieving the goal of power savings without regard to its complexity.

2.4 DVS Algorithm with DVS Processor

With the help of CMOS circuit technique improvement, more and more variable-voltage microprocessors based on CMOS logic have been brought to the market. Researchers then exploited this trend to develop various DVS algorithms so as to make use of these available DVS processors.

DVS algorithms can be classified into intra-task DVS (IntraDVS) and inter-task DVS (InterDVS) algorithms according to how they use slack times. IntraDVS passes on slack times from the current task to itself in the next cycle [4][15]. Usually, IntraDVS determines supply voltage off-line. InterDVS distributes the slack times from the current task to the following tasks. Most current approaches belong to InterDVS [16][14][5][3][9]. Besides these two, researchers further incorporate the strengths of IntraDVS and InterDVS to comprise a hybrid DVS (HybridDVS) algorithms [8][3][9].

The first DVS algorithm was proposed by M. Weiser et al. [20]. This approach is an average throughput-based DVS algorithm and only considers Soft Real-Time environments. It does not guarantee that Hard Real-Time deadlines will be met. Shin and Choi [16] considered delay overheads from power-down mode and voltage switching, and developed Low Power Fixed Priority Scheduling in which they combined off-line and on-line approaches. Their solution still needs a trade-off analysis between increased task execution time and power consumption.

Similar to our approach, the scheme proposed by Manzak and Chakrabarti [12] also tries to find the operating voltage for the processor while it executes each task. They formulate their equation to minimize total energy consumption subject to a constant total computation time. With the method of Lagrange Multiplier, Manzak and Chakrabarti obtain a relationship among task voltages with the constraint of minimum energy consumption. They further develop an

iterative algorithm to adjust task slack times and accordingly adjust voltage assignments. For RM scheduling, the iterative voltage assignments start until combined CPU utilization reaches upper bound of RM test. Still, there is a trade off between energy/power savings and the complexity of this algorithm.

Stochastic IntraDVS described by F. Gruian [4] not only scaled voltages to fill static slack times but also addressed dynamic slack times caused by run-time task executions and combined on-line slack distribution method to achieve the goal (but not optimal) of minimizing energy consumption while meeting all deadlines.

Swaminathan and Chakrabarty [18] took into consideration the effect of voltage switching times on the energy consumption of the task set and presented a novel mixed-integer linear programming model, called extended-low-energy earliest-deadline-first for the NP-complete scheduling algorithm. Their approach is specified for non-preemptive Earliest-Deadline-First (EDF) task set rather than preemptive RM task set.

Pillai and Shin [14] used a scaling factor obtained from lowest CPU frequencies in a given task set for all tasks. Their results show statically scaled RM tasks cannot reduce energy consumption aggressively. Y. H. Tsai et al. [9] further exploit Pillai and Shin's static voltage scaling approach by incorporating it with their deferred-workload-based inter-task DVS.

Liu and Mok [11] define available cycle function and required cycle function to limit CPU speed so that there is no idle cycle when the CPU executes and no job misses its deadline. To solve energy minimization problem, they propose an off-line algorithm and a low complexity on-line dynamic algorithm to reclaim dynamic slack cycles.

Shin and Choi [16] combine not only off-line, on-line approaches but also bring the processor to a power-down mode. Hakan et al. [5] include an off-line solution to compute the optimal processor speed and present on-line speed adjustment mechanism to reclaim dynamic slack times. Similarly, Gruian [4] addresses off-line task stretching to obtain scaling factors and incorporate on-line slack distribution to further utilize dynamic slack times.

2.5 Motivation

As "power-down only" strategy is limited to the length of idle intervals, it is not fully efficient to reduce power consumption [7]. Furthermore, changing power mode incurs interrupt latency and delay overhead, which is usually a trade-off between power dissipation and performance for a Real-Time system [16]. Overall, power-down approach is better in event-driven applications.

As for DVS, off-line algorithms have lower complexity and are easy to implement. On-line algorithms can result in more energy savings, but it is often difficult to predict run-time workload. Furthermore, on-line approaches may suffer from higher complexity and higher chances to miss tasks deadlines. To cope with such difficulties in on-line scheduling, researchers often speed up tasks to catch up with their deadlines. Yet, accelerating tasks by increasing

CPU frequency also presents increasing supply voltage. Therefore, it rarely guarantees minimum power consumption.

Although some of the above studies are for Hard Real-Time environments and the task set is scheduled with the RM algorithm, a mathematical guarantee of the combined CPU utilization of the scaled tasks passing the RM test bound remains to be provided.

3 Formalization of the problem

Since this research is based on RM scheduling algorithm, the system model has the same assumption as Liu and Layland's [10]. Given a task set of n periodic tasks t_i, $i = 1, 2, \ldots, n$, we assume that the deadline of t_i is equal to its period T_i. All tasks are independent with computation times C_i.

The combined CPU utilization u for multiprogramming in a Hard Real-Time Embedded System is:

$$u = \sum_{i=1}^{n} \frac{C_i}{T_i}. \tag{1}$$

Liu and Layland [10] proved that the task set will be schedulable if $u \leq n(2^{\frac{1}{n}} - 1)$ which is called the Rate Monotonic schedulability test or RM test. RM test is sufficient but not necessary [1]. In other words, if a task set passes RM test, it will be schedulable; if it fails, it may or may not be schedulable.

3.1 Energy Consumption

The dynamic power consumed per CPU cycle is:

$$p = afNV^2 \tag{2}$$

where p is the power dissipation, a is the average activity factor, f is the clock frequency of processor or the CPU frequency, N is the effective switching capacitance of wires and transistor gates and V is the supply voltage. Since switching capacitance N is proportional to the number of transistors on the chip [19] and all the tasks use the same processor, N is constant. We only consider reducing the CPU frequency f or the supply voltage $V = V_{dd}$ here to achieve power reduction. The frequency at which a device can operate and its supply voltage are interdependent. This dependence has been modeled as follows [22]:

$$f = k\frac{(V_{dd} - V_t)^\alpha}{V_t} \tag{3}$$

where V_t is the threshold voltage and α is a process dependent parameter that models velocity saturation. The energy consumption of a program shows convex energy behavior, i.e. there exists an optimal CPU frequency at which energy consumption is minimum [24,25], thus finding such frequencies is relevant. It has been empirically observed that for velocity saturation devices

(most modern CPUs), [26] frequency scales almost linearly with V_{dd}. Since V is approximately proportional to CPU frequency f, reducing f also reduces V proportionally [21]. We can say the power dissipation p is approximately proportional to the cube of its CPU frequency f:

$$p \propto f^3 \tag{4}$$

Reducing the CPU frequency f by a factor $\frac{1}{X_i}$, where X_i is a scaling factor and is always greater than 1, results in increasing the computation time C_i to $X_i C_i$. Thus, we have the new combined CPU utilization equation u' as follows:

$$u' = \sum_{i=1}^{n} \frac{X_i C_i}{T_i} = \frac{X_1 C_1}{T_1} + \frac{X_2 C_2}{T_2} + \cdots + \frac{X_n C_n}{T_n} \leq n(2^{\frac{1}{n}} - 1). \tag{5}$$

The energy consumption E_i in executing task t_i is:

$$E_i \propto C_i \times p_i \tag{6}$$

where C_i is the computation time of task t_i, p_i is the power dissipation of the processor due to t_i .

According to (4), we could modify energy consumption equation E_i as the following form

$$E_i \propto C_i \times {f_i}^3 \tag{7}$$

where f_i is the CPU frequency.

Since f is reduced to $f' = \frac{f}{X_i}$, along with the equivalent scaling $X_i C_i$, the scaled energy consumption E'_i becomes

$$E'_i \propto X_i C_i \times (\frac{f}{X_i})^3 = \frac{C_i \times f^3}{X_i^2} \tag{8}$$

Let, for all tasks, when no frequency scaling is used, the net energy consumed be represented by E and when frequency scaling is used by E'. Since f is common in both E and E' for comparative purposes we have:

$$E \propto \sum_{i=1}^{n} C_i \tag{9}$$

$$E' \propto \min \sum_{i=1}^{n} \frac{C_i}{X_i^2} \tag{10}$$

3.2 Problem Statement

Based on the above derivations, we can restate our problem in the following form:

Given $u = \sum_{i=1}^{n} \frac{C_i}{T_i} < n(2^{\frac{1}{n}} - 1)$, our goal is to find $\min \sum_{i=1}^{n} \frac{C_i}{X_i^2}$ subject to the constraints $\sum_{i=1}^{n} \frac{X_i C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$ and $1 \leq X_i \leq \frac{T_i}{C_i}$. We assume that the switching capacitance involved for every task is about the same. Associated with our problem are the following parameters:

- u: combined CPU utilization
- n: number of tasks in a task set
- C_i: computation time of task t_i
- T_i: request period of task t_i
- X_i: scaling factor of task t_i

4 Solution to the optimization problem

Based on the above derivations, we restate the problem in the following mathematical form:

Problem: Find $\hat{X} := (\hat{X}_1, \ldots, \hat{X}_n)$ which yields

$$\min \sum_{i=1}^{n} \frac{C_i}{X_i^2}, \tag{11}$$

where $0 < C_i$ for all i, subject to the constraints

1. $\sum_{i=1}^{n} \frac{X_i C_i}{T_i} \leq n(2^{1/n} - 1)$ with $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$,
2. $1 \leq X_i \leq \frac{T_i}{C_i}$, $i = 1, \ldots, n$.

Let

$$f(X) := \sum_{i=1}^{n} \frac{C_i}{X_i^2}.$$

When $n = 1$ the minimization problem is trivial: $f(X) = \frac{C_1}{X_1^2}$ and its minimum is attained at $X = \frac{T_1}{C_1}$. So from now on we only consider $n \geq 2$.

Remark 1 Set

$$S := \{x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{X_i C_i}{T_i} \leq n(2^{1/n} - 1), 1 \leq X_i \leq \frac{T_i}{C_i}\}. \tag{12}$$

The minimization problem can be written as: find $\hat{X} \in S$ that gives $\min_{X \in S} f(X)$. The condition $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$ ensures that the point $(1, \ldots, 1)$ is in the region. So the constraint set S (the domain of the minimization problem) is nonempty. Notice that S is a closed and bounded set in $\mathbb{R}^n$, that is, S is a compact set. Moreover the function $f(X)$ is continuous on the compact set S and f is always positive over S. By Weierstrass's theorem [6, p.541], there is a minimum of f in S. So the minimization problem (11) is solvable. However Weierstrass's theorem does not tell the minimal elements $\hat{X} \in S$, i.e, $f(\hat{X}) = \min_{X \in S} f(\hat{X})$.

The forthcoming discussion is to find $\hat{X}$ and indeed it is unique. We start with a lemma.

Lemma 1 The sequence $h(n) := n(2^{1/n} - 1)$ is strictly monotonic decreasing and $\lim_{n\to\infty} h(n) = \ln 2$. Moreover $h(n) < 1$ for all positive integers n except $n = 1$; $h(1) = 1$.

Proof Clearly $h(1) = 1$. Let $a := 2^{\frac{1}{n(n+1)}} > 1$. By using the identity $a^n - 1 = (a-1)(a^{n-1} + a^{n-2} + \cdots + a + 1)$, we obtain

$$\begin{aligned} h(n) - h(n+1) &= n(2^{\frac{1}{n}} - 1) - (n+1)(2^{\frac{1}{n+1}} - 1) \\ &= n\left[\left(2^{\frac{1}{n(n+1)}}\right)^{n+1} - 1\right] - (n+1)\left[\left(2^{\frac{1}{n(n+1)}}\right)^{n} - 1\right] \\ &= (a-1)\left[n(a^n + a^{n-1}\cdots + 1) - (n+1)(a^{n-1} + \cdots + 1)\right] \\ &= (a-1)[na^n - (a^{n-1} + \cdots + 1)] \geq 0 \end{aligned}$$

since $a^n \geq a^i$ for $i = 1, \ldots, n-1$ as $a > 1$. So $h(n)$ is strictly monotonic decreasing. By l'Hospital's rule,

$$\lim_{n\to\infty} h(n) = \lim_{n\to\infty} \frac{2^{1/n} - 1}{\frac{1}{n}} = \lim_{n\to\infty} \frac{-\frac{1}{n^2} 2^{1/n} \ln 2}{-\frac{1}{n^2}} = \lim_{n\to\infty} 2^{1/n} \ln 2 = \ln 2.$$

■

Remark 2 When $n \geq 2$, one can rewrite the set S as

$$S = \{x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{X_i C_i}{T_i} \leq n(2^{1/n} - 1), 1 \leq X_i\},$$

i.e., one can ignore the condition $X_i \leq \frac{T_i}{C_i}$ in (12) since it follows from $\sum_{i=1}^{n} \frac{X_i C_i}{T_i} \leq n(2^{1/n} - 1) < 1$ by Lemma 1 and $1 \leq X_i$. But we will keep using (12) as the description of S.

Each $X \in S$ must satisfy

$$X_i < \frac{T_i}{C_i}, \quad i = 1, \ldots, n, \tag{13}$$

otherwise $X_j = \frac{T_j}{C_j}$ for some j and we would have

$$\sum_{i=1}^{n} \frac{X_i C_i}{T_i} \geq \frac{X_j C_j}{T_j} = 1 > n(2^{1/n} - 1),$$

by Lemma 1. This contradicts (12).

The idea of the following lemma is to narrow down our search of the minimal elements in S.

Lemma 2 The minimization problem (11) only has solution(s) in the compact set R, where

$$R := \{x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{X_i C_i}{T_i} = n(2^{1/n} - 1), 1 \leq X_i \leq \frac{T_i}{C_i}\} \subset S. \tag{14}$$

Set

$$X_\epsilon := X + \epsilon(1, \ldots, 1) = (X_1 + \epsilon, \ldots, X_n + \epsilon).$$

If the minimum of $f(X)$ occurred at X over the region defined by

$$\sum_{i=1}^{n} \frac{X_i C_i}{T_i} < n(2^{1/n} - 1),$$

then by (13) for sufficiently small ϵ

$$1 \leq (X_\epsilon)_i \leq \frac{T_i}{C_i} \qquad i = 1, \ldots, n$$

and

$$\sum_{i=1}^{n} \frac{(X_\epsilon)_i C_i}{T_i} = \sum_{i=1}^{n} \frac{(X_i + \epsilon) C_i}{T_i} = \sum_{i=1}^{n} \frac{X_i C_i}{T_i} + \epsilon \sum_{i=1}^{n} \frac{C_i}{T_i} < n(2^{1/n} - 1).$$

In other words, $X_\epsilon \in S$. Clearly $f(X_\epsilon) < f(X)$, a contradiction.

■

From now on we set

$$K := n(2^{1/n} - 1).$$

By Lemma 2 the minimization problem (11) is reduced to find $\hat{X} \in R$ that gives

$$\min_{X \in R} f(X). \tag{15}$$

The following result reveals a special entry order of the minimal element $\hat{X}$ and this order is completely determined by the ordering of T's.

Theorem 1 Suppose that $T_{\sigma(1)} \geq \cdots \geq T_{\sigma(n)}$ with some permutation $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ and the minimization problem attains its minimum at $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_n) \in R$. Then $\hat{X}_{\sigma(1)} \geq \cdots \geq \hat{X}_{\sigma(n)}$ (≥ 1). In particular, if $T_1 \geq \cdots \geq T_n$, then $\hat{X}_1 \geq \cdots \geq \hat{X}_n$.

Proof Without loss of generality, we may assume that σ is the identity, i.e., $T_1 \geq \cdots \geq T_n$. Suppose on the contrary that $\hat{X}_j > \hat{X}_i$ (≥ 1) for some $1 \leq i < j \leq n$. By (13)

$$\hat{X}_i < \frac{T_i}{C_i}, \quad i = 1, \ldots, n.$$

Let

$$\hat{X}_i(\delta) = \hat{X}_i + \delta, \qquad \hat{X}_j(\delta) = \hat{X}_j - \delta \frac{C_i T_j}{C_j T_i}$$

and $\hat{X}_k(\delta) = \hat{X}_k$ for all $k \neq i, j$. So

$$\sum_{k=1}^{n} \frac{C_k \hat{X}_k(\delta)}{T_k} = \sum_{k \neq i,j} \frac{C_k \hat{X}_k(\delta)}{T_k} + \frac{C_i \hat{X}_i(\delta)}{T_i} + \frac{C_j \hat{X}_j(\delta)}{T_j} = K.$$

Since $\hat{X}_i(\delta) < \frac{T_i}{C_i}$ and $1 < \hat{X}_j(\delta)$ for sufficiently small $\delta > 0$, we have $\hat{X}(\delta) \in R$. Set

$$\begin{aligned}
\varphi(\delta) &:= f(\hat{X}(\delta)) \\
&= \frac{C_1}{\hat{X}_1^2} + \cdots + \frac{C_i}{(\hat{X}_i(\delta))^2} + \cdots + \frac{C_j}{(\hat{X}_j(\delta))^2} + \cdots + \frac{C_n}{\hat{X}_n^2} \\
&= \frac{C_1}{\hat{X}_1^2} + \cdots + \frac{C_i}{(\hat{X}_i + \delta)^2} + \cdots + \frac{C_j}{(\hat{X}_j - \delta \frac{C_i T_j}{C_j T_i})^2} + \cdots + \frac{C_n}{\hat{X}_n^2}.
\end{aligned}$$

Now

$$\begin{aligned}
\frac{d\varphi(\delta)}{d\delta} &= \frac{-2C_i}{(\hat{X}_i + \delta)^3} + \frac{2\frac{C_i T_j}{T_i}}{(\hat{X}_j - \delta \frac{C_i T_j}{C_j T_i})^3} \\
&= \frac{2C_i \left[(\hat{X}_i + \delta)^3 \frac{T_j}{T_i} - (\hat{X}_j - \delta \frac{C_i T_j}{C_j T_i})^3 \right]}{(\hat{X}_i + \delta)^3 (\hat{X}_j - \delta \frac{C_i T_j}{C_j T_i})^3}.
\end{aligned}$$

So

$$\varphi'(0) = \frac{2C_i \left[\hat{X}_i^3 \frac{T_j}{T_i} - \hat{X}_j^3 \right]}{\hat{X}_i^3 \hat{X}_j^3} < 0,$$

since $1 \leq \hat{X}_i < \hat{X}_j$ and $T_i \geq T_j > 0$. This contradicts the assumption that $f(\hat{X})$ is the minimum.

■

We will first solve a slightly different problem: find $X \in R'$ that yields

$$\min_{X \in R'} f(X) \tag{16}$$

where

$$R' := \{X \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{X_i C_i}{T_i} = K\}.$$

In other words, we remove the condition $1 \leq X_i \leq \frac{T_i}{C_i}$ from the original minimization problem (11). Note that $R \subset R'$ and the region R' is a hyperplane in $\mathbb{R}^n$ represented by the equation

$$\sum_{i=1}^{n} \frac{X_i C_i}{T_i} = K.$$

By the method of Lagrange multiplier, set

$$\nabla f = \lambda \nabla g$$

which implies

$$-\frac{2C_i}{X_i^3} = \lambda \frac{C_i}{T_i}.$$

Denote by $X^L = (X_1^L, \ldots, X_n^L)$ the solution to (16). Since $C_i \neq 0$,

$$X_i^L = \left(\frac{2T_i}{-\lambda}\right)^{1/3}. \tag{17}$$

Substitute (17) into $\sum_{i=1}^{n} \frac{X_i C_i}{T_i} = K$ to have

$$\sum_{j=1}^{n} \left(\frac{2T_j}{-\lambda}\right)^{1/3} \frac{C_j}{T_j} = K.$$

Thus we have

$$\frac{1}{(-\lambda)^{1/3}} \sum_{j=1}^{n} (2T_j)^{1/3} \frac{C_j}{T_j} = K. \tag{18}$$

So from (17) and (18)

$$X_i^L = \frac{(2T_i)^{1/3} K}{\sum_{j=1}^{n} (2T_j)^{1/3} \frac{C_j}{T_j}} = \frac{T_i^{1/3} K}{\sum_{j=1}^{n} T_j^{1/3} \frac{C_j}{T_j}} < \frac{T_i}{C_i} \tag{19}$$

(since $K < 1$) which is the only critical point of f over R'. Moreover

$$f(X^L) = \sum_{i=1}^{n} \frac{C_i}{(X_i^L)^2} = \frac{(\sum_{j=1}^{n} T_j^{1/3} \frac{C_j}{T_j})^2}{K^2} (\sum_{i=1}^{n} \frac{C_i}{T_i^{2/3}}) \tag{20}$$

is the local minimum value of $f(X)$ over R'. Thus it is the global minimum over R'.

If $T_1 \geq \cdots \geq T_n$, then from (19)

$$X_1^L \geq \cdots \geq X_n^L.$$

However we cannot conclude that $1 \leq X_i^L$ for all $i = 1, \ldots, n$, i.e., X^L may not be in R since the second constraint of (11) is not satisfied. Nevertheless, it is clear that

$$f(X^L) = \min_{X \in R'} f(X) \leq \min_{X \in R} f(X). \tag{21}$$

Theorem 2 Let $T_1 \geq \cdots \geq T_n$. Let $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_n) \in R$ be a solution to the minimization problem (11).

1. $X_n^L > 1$ if and only if $\hat{X}_n > 1$. In this case, $\hat{X} = X^L$ is a solution to the minimization problem (11).

2. If $X_n^L \le 1$, then

$$\hat{X}_1 \ge \cdots \ge \hat{X}_{r-1} > \hat{X}_r = \cdots = \hat{X}_n = 1 \tag{22}$$

for some $r = 1, \ldots, n$. Moreover $(\hat{X}_1, \ldots, \hat{X}_{r-1})$ is a solution to the following minimization problem

$$\min \sum_{i=1}^{r-1} \frac{C_i}{X_i^2} + \sum_{i=r}^{n} C_i$$

subject to the (new) conditions
(a) $1 \le X_i \le \frac{T_i}{C_i}$, $i = 1, \ldots, r-1$,
(b) $\sum_{i=1}^{r-1} \frac{C_i X_i}{T_i} \le \hat{K}$, where $\hat{K} := K - \sum_{i=r}^{n} \frac{C_i}{T_i}$.
Namely,

$$\hat{X}_i = \frac{T_i^{1/3} \hat{K}}{\sum_{j=1}^{r-1} T_j^{1/3} \frac{C_j}{T_j}}, \qquad i = 1, \ldots, r-1. \tag{23}$$

In either case, $\hat{X}$ is the unique solution to the minimization problem (11).

Proof By Theorem 1 the solution $\hat{X} \in R$ to the minimization problem (11) satisfies $\hat{X}_1 \ge \cdots \ge \hat{X}_n$.

(1) By (19) $X_1^L \ge \cdots \ge X_n^L$, $X_i^L < \frac{T_i}{C_i}$ for all $i = 1, \ldots, n$, and $\sum_{i=1}^{n} \frac{X_i^L C_i}{T_i} = K$. So $X_n^L \ge 1$ if and only if $X^L \in \mathrm{Int}(R)$, the interior of R. In this case, it amounts to $\hat{X} = X^L$ by (21) and it is the unique solution since we only have one critical point for the minimization problem over R'.

(2) If $X_n^L \le 1$, then $X^L \notin \mathrm{Int}(R)$. Since X^L is the only critical point in R' and $R \subset R'$, we conclude that the minimum point(s) of f over R must be in the boundary ∂R of R:

$$\partial R := \{x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{X_i C_i}{T_i} = K, X_i = 1 \text{ for some } i \text{ or } X_j = \frac{T_j}{C_j} \text{ for some } j\}.$$

Since $X_i < \frac{T_i}{C_i}$ for all i by (13), $\hat{X}$ must occur in

$$\hat{\partial} R := \{x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{X_i C_i}{T_i} = K, X_i = 1 \text{ for some } i\}.$$

In other words, via Theorem 1 we have (22), i.e.,

$$\hat{X}_1 \ge \cdots \ge \hat{X}_{r-1} > \hat{X}_r = \cdots = \hat{X}_n = 1,$$

for some $r = 1, \ldots, n$. Because the region

$$R_r := \{(X_1, \ldots, X_{r-1}, 1, \ldots, 1) \in \mathbb{R}^n : \sum_{i=1}^{r-1} \frac{C_i X_i}{T_i} + \sum_{i=r}^{n} \frac{C_i}{T_i} = K,\ 1 \le X_i \le \frac{T_i}{C_i},\ i = 1, \ldots, r-1\}$$

is a subset of R, $(\hat{X}_1, \ldots, \hat{X}_{r-1})$ is the solution to the following minimization problem

$$\min \sum_{i=1}^{r-1} \frac{C_i}{X_i^2} + \sum_{i=r}^{n} C_i$$

subject to the (new) constraints

1. $1 \leq X_i \leq \frac{T_i}{C_i}$, $i = 1, \ldots, r-1$,
2. $\sum_{i=1}^{r-1} \frac{C_i X_i}{T_i} \leq \hat{K}$, where $\hat{K} := K - \sum_{i=r}^{n} \frac{C_i}{T_i}$.

Since $T_1 \geq \cdots \geq T_{r-1}$ and $\hat{X}_{r-1} > 1$, by Theorem 2(1) the minimum is attained at $(\hat{X}_1, \cdots, \hat{X}_{r-1})$ which is provided by the method of Lagrange multiplier, i.e.,

$$\hat{X}_i = \frac{T_i^{1/3} \hat{K}}{\sum_{j=1}^{r-1} T_j^{1/3} \frac{C_j}{T_j}} > 1, \qquad i = 1, \ldots, r-1.$$

Moreover the solution is unique once r is fixed.

To show that $\hat{X}$ is unique, it is sufficient to show that r is unique. Suppose on the contrary that $(\hat{X}_1, \ldots, \hat{X}_{r-1}, 1, \ldots, 1) \in \mathbb{R}^n$ and $(\tilde{X}_1, \ldots, \tilde{X}_{r'-1}, 1, \ldots, 1) \in \mathbb{R}^n$ both yield the minimum, i.e.,

$$f((\hat{X}_1, \ldots, \hat{X}_{r-1}, 1, \ldots, 1)) = f((\tilde{X}_1, \ldots, \tilde{X}_{r'-1}, 1, \ldots, 1)) = \min_{X \in R} f(X)$$

but $r \neq r'$. In other words,

$$\tilde{X}_i = \frac{T_i^{1/3} \tilde{K}}{\sum_{j=1}^{r-1} T_j^{1/3} \frac{C_j}{T_j}} > 1, \qquad i = 1, \ldots, r'-1.$$

For definiteness, assume $r < r'$. Clearly $R_r \subset R_{r'}$ and $(\tilde{X}_1, \ldots, \tilde{X}_{r'-1})$ is the unique solution to the following minimization problem

$$\min \sum_{i=1}^{r'-1} \frac{C_i}{X_i^2} + \sum_{i=r'}^{n} C_i$$

subject to the constraints

1. $1 \leq X_i \leq \frac{T_i}{C_i}$, $i = 1, \ldots, r'-1$,
2. $\sum_{i=1}^{r'-1} \frac{C_i X_i}{T_i} \leq \tilde{K}$, where $\tilde{K} := K - \sum_{i=r'}^{n} \frac{C_i}{T_i}$.

We would then have $f(\hat{X}_1, \ldots, \hat{X}_{r-1}, 1, \ldots, 1) > f(\tilde{X}_1, \ldots, \tilde{X}_{r'-1}, 1, \ldots, 1)$, a contradiction. So r is unique. Hence $(\hat{X}_1, \ldots, \hat{X}_{r-1})$ and $\hat{X}$ are unique.

■

When $X_n^L \leq 1$, to solve Problem 11, it suffices to determine $r-1$ which is the largest index i such that $\hat{X}_i > 1$ and apply (23). The following is very useful with respect to the determination of $r-1$.

Proposition 1 Let $T_1 \geq \cdots \geq T_n$. Let $r-1$ be the largest integer i such that $\hat{X}_i > 1$ ($r-1=0$ means that all $\hat{X}$'s are 1). Then

$$r - 1 \leq s_1 \tag{24}$$

where s_1 denotes the largest integer i such that $X_i^L > 1$ ($s_1 = 0$ means that all X^L's are less than or equal to 1), i.e.,

$$X_1^L \geq \cdots \geq X_{s_1}^L > 1 \geq X_{s_1+1}^L \geq \cdots \geq X_n^L.$$

Evidently, if $X_n^L > 1$, then $\hat{X} = X^L$ and $r - 1 = s_1$.

Proof On the contrary suppose that $r-1 > s_1$. We would have $r-1 \geq s_1+1$ so that $X_{r-1}^L \leq 1$. From (19)

$$X_{r-1}^L = \frac{T_{r-1}^{1/3} K}{\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j}} \leq 1 \Longleftrightarrow K \leq \frac{\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j}}{T_{r-1}^{1/3}}.$$

So

$$\begin{aligned} K \left(\sum_{j=r}^n T_j^{1/3} \frac{C_j}{T_j} \right) &\leq \frac{\left(\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j} \right)}{T_{r-1}^{1/3}} \left(\sum_{j=r}^n T_j^{1/3} \frac{C_j}{T_j} \right) \\ &= \left(\sum_{j=r}^n \left(\frac{T_j}{T_{r-1}} \right)^{1/3} \frac{C_j}{T_j} \right) \left(\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j} \right) \\ &\leq \left(\sum_{j=r}^n \frac{C_j}{T_j} \right) \left(\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j} \right) \end{aligned} \tag{25}$$

since $T_1 \geq \cdots \geq T_n$. Now

$$\begin{aligned} \frac{\hat{X}_{r-1}}{X_{r-1}^L} &= \left[\frac{T_{r-1}^{1/3} K'}{\sum_{j=1}^{r-1} T_j^{1/3} \frac{C_j}{T_j}} \right] \left[\frac{\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j}}{T_{r-1}^{1/3} K} \right] \\ &= \frac{K'}{K} \left[\frac{\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j}}{\sum_{j=1}^{r-1} T_j^{1/3} \frac{C_j}{T_j}} \right] \\ &= \left[\frac{K - \sum_{j=r}^n \frac{C_j}{T_j}}{K} \right] \left[\frac{\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j}}{\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j} - \sum_{j=r}^n T_j^{1/3} \frac{C_j}{T_j}} \right] \\ &= \frac{K \left(\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j} \right) - \left(\sum_{j=r}^n \frac{C_j}{T_j} \right) \left(\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j} \right)}{K \left(\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j} \right) - K \left(\sum_{j=r}^n T_j^{1/3} \frac{C_j}{T_j} \right)} \end{aligned}$$

By (25) we have $\hat{X}_{r-1} \leq X_{r-1}^L \leq 1$, contradicting the fact that $\hat{X}_{r-1} > 1$.

■

Proposition 1 leads to an algorithm for the determination of $r-1$ and thus $\hat{X}$. In general we may not be able to locate $r-1$ right after the first application of Lagrange multiplier (see Proposition 2). However we can repeat the process and eventually will get to the value of $r-1$.

The following describes a general property among consecutive iterations. The proof is similar to that of (24) and is omitted. It implies that in general one iteration is not enough in order to determine $\hat{X}$ and $r-1$.

Proposition 2 Suppose $T_1 \geq \cdots \geq T_n$. For each $1 \leq i \leq s_\ell$ $(X_i^{(\ell)} > 1)$, $X_i^{(\ell+1)} < X_i^{(\ell)}$. In other words, each needed iteration will decrease those $X_i > 1$.

Example 1 For instance, take a look at the following experimental example. Suppose

$$T := (T_1, T_2, T_3, T_4) = (25391, 14905, 12913, 5758)$$

and

$$C := (C_1, C_2, C_3, C_4) = (4616, 6073, 575, 515).$$

Notice that $T_1 \geq T_2 \geq T_3 \geq T_4$. So three iterations are needed to get $\hat{X} =$

ith iteration	$X_1^{(i)}$	$X_2^{(i)}$	$X_3^{(i)}$	$X_4^{(i)}$
1	1.23	1.03	0.99	0.75
2	1.19	0.99	1	1
3	1.18	1	1	1

$(1.18, 1, 1, 1)$ and $r - 1 = 1$.

Remark 3 In case Theorem 2(2) occurs,

$$Y := (X_1^L, \ldots, X_{s_1}^L, 1, \ldots, 1) \notin R$$

since $\sum_{i=1}^n \frac{Y_i C_i}{T_i} > \sum_{i=1}^n \frac{X_i^L C_i}{T_i} = K$ and because of (14). So Y is not a solution to the minimization problem (11). Thus there is no contradiction though $f(Y) < f(X^L) (\leq \min_{X \in R} f(X))$. Similar conclusion can be reached for consecutive iterations with respect to Proposition 2.

The following is another description from a top-down view. Similar algorithm can be derived from it.

Theorem 3 Let $T_1 \geq \cdots \geq T_n$. Suppose that the minimization problem (15) has solution $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_n) \in R$.

1. $X_n^L > 1$ if and only if $\hat{X}_n > 1$. In this case, $\hat{X} = X^L$ is the unique solution to the minimization problem (11).
2. If $X_n^L \leq 1$, then the unique solution $\hat{X}_1 \geq \cdots \geq \hat{X}_{r-1} > \hat{X}_r = \cdots = \hat{X}_n = 1$ has r where $r-1$ is the first integer i such that

$$\frac{T_i^{1/3} K_{(i)}}{\sum_{j=1}^{i} T_j^{1/3} \frac{C_j}{T_j}} > 1 \quad \text{and} \quad \frac{T_{i+1}^{1/3} K_{(i+1)}}{\sum_{j=1}^{i+1} T_j^{1/3} \frac{C_j}{T_j}} \leq 1,$$

where $K_{(k)} := K - \sum_{j=k+1}^{n} \frac{C_j}{T_j}$.

■

4.1 Scaling Factors Algorithm

On the basis of above proofs, we restate our solution as follows:

Step 1 Sort taks t_i in descending order of their time periods, i.e., $(T_1 \geq \cdots \geq T_n)$.
Step 2 Compute scaling factors X_i by the method of Lagrange multiplier, $(X_1 \geq \cdots \geq X_n)$. If all $X_i > 1$, then they are the solution. Exit.
Step 3 Otherwise, find the first $X_r' \leq 1$ in the sorted list. Set $X_i = 1$ for all $i = r', \ldots, n$ and $K = K - \sum_{i=r'}^{n} \frac{C_i}{T_i}$. Set $n = r' - 1$ to restrict future iterations to operate only on $t_1 \cdots t_{r'-1}$.
Step 4 Repeat Steps 2 and Step 3 until all $X_i \geq 1$.

We then design our algorithm according to the above solution. The off-line scaling factor assignment algorithm is listed in Table 1. Once the frequency scaling factors X_i are found, the minimum energy consumption can be computed using the formula from (10).

The worst case complexity of the algorithm is $O(n^2)$ (when in each iteration, only one X_i is set to 1), while the best case complexity is $O(n \cdot \log\ n)$ (when all $X_i \geq 1$ at the beginning, (the complexity comes from sorting); note: the denominator of X_i from (19)is the same for all iterations when n does not change and therefore can be computed outside the for loop in $O(n)$ time, a for loop for its computation has not been shown above to maintain clarity).

5 Results and Analysis

Consider the task set in Table 2. Table 2 contains three tasks that are given priorities through RM algorithm. Computation time and request period are measured in the number of clock cycles per second. Note that priorities are presented as integers in descending order. Namely, the higher the integer, the greater the priority.

Task a is given the highest priority since it has the shortest request period. Their combined CPU utilization is 0.75 and passes RM test, which equals 0.78 from (3). Thus, this task set is guaranteed to meet all its deadlines.

Procedure ScaleFactors
Inputs: **float** $T[n], C[n]$; **int** n
//Time periods, Computation Times, number of tasks
Output: **float** $X[n]$ //ScaleFactors
Locals: **int** r', i, j; **float** K

Sort tasks t_i in descending order of their periods T_i .
$K \leftarrow n(2^{1/n} - 1)$
for $i = 1$ to n ***do***
//for each task t_i in the sorted task set
$X_i \leftarrow \frac{T_i^{1/3} K}{\sum_{j=1}^{n} T_j^{1/3} \frac{C_j}{T_j}}$
//using Lagrange multiplier, from (19)
if $(X_i \leq 1)$
$r' \leftarrow i$
for $i = r'$ to n ***do***
$X_i \leftarrow 1$
$K \leftarrow K - \frac{C_i}{T_i}$
endfor
//Reset i and n to iterate
//and recompute $X_1, \ldots, X_{r'-1}$
$n \leftarrow r' - 1$
$i \leftarrow 1$
endif
endfor
end ScaleFactors

Table 1 Scaling Algorithm

Task	Computation Time, C	Request Period, T	Utilization, U	Priority
a	3	8	0.38	3
b	3	10	0.30	2
c	1	14	0.07	1

Table 2 Example Task Set A

Figure 1 shows the timeline for task set A before scaling. The y axis represents CPU frequency of the processor, and the x axis represents task computation time. The time a_2 is the deadline of task a upon first release; likewise, b_2, c_2 are the deadlines of tasks b and c upon first release. Notice that all the three tasks finish their execution before their deadlines. After scaling all computation times according to each corresponding scaling factor from our solution, Tables 3 and 4 show the results of energy reduction. Energy in Tables 3 and 4 are calculated from (9, 10). Although these values are not in Joules, they form

Fig. 1 Timeline for Task Set A before scaling, where computation times are specified at the maximum CPU frequency

Task	Computation Time, C	Utilization, U	Energy
a	3	0.38	
b	3	0.30	7
c	1	0.07	

Table 3 Energy Consumption before Scaling for Task Set A

Task	Computation Time, C	Utilization, U	Freq. Scale, X	Minimum Energy
a	3	0.38	1	
b	3.20	0.32	0.94	6.35
c	1.19	0.08	0.84	

Table 4 Energy Consumption after Scaling for Task Set A

a good basis for comparison; the actual energy in Joules can be computed by multiplying the values in the table by a constant factor.

As we see in Table 4, the CPU frequency and computation time of each task change individually. For instance, CPU frequency and computation time of task a remain the same after scaling, but in task b, CPU frequency is slightly reduced to 0.94 and its computation time expands from 3 to 3.21. The CPU frequency of task c is reduced even more. This phenomenon just illustrates the scaling algorithm: the longer the period of a task, the greater its scaling factor. The combined CPU utilization after scaling just reaches 0.78 (the upper bound of RM test), but the scaled task set saves 17.79% energy.

The scaled timeline is depicted in Figure 2. Note that not only both CPU frequencies of tasks b and c are reduced, but slack time is also shortened. However, each task still completes its execution before deadline. Consider another

Fig. 2 Scaled Timeline for Task Set A

example represented by the task set B in Table 5. Task b in Table 5 is given the highest priority and task a the lowest. Their combined CPU utilization is 0.49, much less than 0.78. The timelime scheduled by RM algorithm for task set B is as follows: The slack time in task set B is much longer than in task set A, since its combined CPU utilization is much less than in task set A. Task b has the greatest priority and executes first. Again, we assume that every task executes at the maximum CPU frequency 1.0.

Task	Computation Time, C	Request Period, T	Utilization, U	Priority
a	2	14	0.14	1
b	1	10	0.10	3
c	3	12	0.25	2

Table 5 Example Task Set B

Fig. 3 Timeline for Task Set B

Tables 6 and 7 show the result of power reduction after scaling. The scaled task set saves almost 60.22 % energy. The timeline for task set B after CPU

Task	Computation Time, C	Utilization, U	Energy
a	2	0.14	
b	1	0.10	6
c	3	0.25	

Table 6 Energy Consumption before Scaling for Task Set B

Task	Computation Time, C	Utilization, U	Freq. Scale, X	Minimum Energy
a	3.32	0.24	0.60	
b	1.48	0.16	0.63	2.39
c	4.73	0.37	0.67	

Table 7 Energy Consumption after Scaling for Task Set B

Fig. 4 Scaled Timeline for Task Set B

frequency scaling has been shown in Figure 4.

The characteristics of tasks of a real-workload from the real-time systems of aircraft avionics [23] have been shown in Table 8. The code CP denotes Critical Periodic Task, EP denotes Essential Periodic Task, BU Background Unlikely and BC Background Certain. We can schedule all the CP tasks on one processor and the remaining tasks on another processor. The combined utilization of the seven (7) CP tasks is 0.5655 which is less than the upper

Task	Period	Comp Time	Utilization	Energy
Aircraft Flight Data (CP)	55	8	0.145	
Steering (CP)	80	6	0.075	
Radar Search (CP)	80	2	0.025	
Radar Tracking (CP)	40	2	0.050	30
Target Tracking (CP)	40	4	0.100	
Weapon Trajectory (CP)	100	7	0.070	
Weapon Release (CP)	10	1	0.100	
HUD Display (EP)	52	6	0.115	
MPD Tactical Display (EP)	52	8	0.154	
Keyset Response (BU)	100	1	0.010	
RWR Prog. Input (BU)	100	1	0.010	23
Poll RWR (BU)	100	2	0.020	
Periodic BIT (BC)	1000	5	0.005	

Table 8 Energy before Scaling for Two Task Sets of a Real-Workload

Task	Freq. Scale	Comp Time	Util.	Energy
Aircraft Flight Data (CP)	0.76	10.50	0.191	
Steering (CP)	0.67	8.92	0.112	
Radar Search (CP)	0.67	2.97	0.037	
Radar Tracking (CP)	0.85	2.36	0.059	19.84
Target Tracking (CP)	0.85	4.72	0.118	
Weapon Trajectory (CP)	0.62	11.21	0.112	
Weapon Release (CP)	1.00	1.00	0.100	
HUD Display (EP)	0.45	13.26	0.255	
MPD Tactical Display (EP)	0.45	17.69	0.340	
Keyset Response (BU)	0.36	2.75	0.027	
RWR Prog. Input (BU)	0.36	2.75	0.027	3.54
Poll RWR (BU)	0.36	5.50	0.055	
Periodic BIT (BC)	0.17	29.61	0.030	

Table 9 Energy after Scaling for Two Task Sets of a Real-Workload

bound of 0.7286 for seven tasks. The combined utilization of the other six (6) tasks is 0.3142 which is less than the upper bound of 0.7347 for six tasks. (Note that the combined utilization of all thirteen tasks is 0.88 which exceeds the upper bound of 0.712). The periods of the scaled tasks are the same as the periods of unscaled tasks. Table 9 shows that the minimum energy for the critical periodic tasks is $2/3^{rd}$ of the base case and that of the "other tasks" is $1/6^{th}$ of the base case. We note that a futuristic implanted medical device (e.g. insulin injectors for diabetes patients) could have a similar task set and using this approach its battery life could be extended in a similar fashion. Another

example would be a similar task set used in space probe where the battery energy depends upon the duration of sunlight falling on the solar panels.

The results show that the percentage of power saving varies dramatically because this approach is independent of task sets. In general, the more difference between CPU combined utilization and the upper bound of RM test, the more power reduction.

6 Conclusion

This paper has provided provably optimal scaling factors, on the worst-case execution time of RM tasks, when using DVS technique. The scaled tasks still satisfy the utilization bound and therefore remain schedulable by RMPO method. Since RM is widely used, we expect this work to lead to better energy management.

In practice the CPU may not allow continuous frequency scaling and only allow discrete scales. In such a case, we should use a specific floor function for the scaled computation time to achieve power reduction. In such cases, the savings in energy consumption will be less than the optimal generated in this paper due to the limited CPU voltage variety. But, we also note that our analysis is a conservative estimate of the energy savings with frequency scaling because of the following. First, there is idle energy which is incurred in the base case. Second, at lower supply voltages, static power is also reduced. Third, the lower supply voltage may result in somewhat lower operating temperature which in turn reduces the leakage current and the leakage power.

However, this work still provides a lower bound on the dynamic energy reduction possible using DVS when using the RM approach which may be helpful. We should also consider the effect of the overhead of voltage switching [18]. We could further incorporate our approach with different dynamic reclaiming algorithms. We will extend this work with real life examples in our future work.

7 Acknowledgements

We wish to thank Drs. James Anderson and Sanjoy Baruah of University of North Carolina, Chapel Hill for pointing to the real workloads discussed in this paper and Dr. Adit D. Singh, Dept of Electrical and Computer Engineering, Auburn University for discussing issues related to power and frequency in CMOS circuits. We gratefully wish to thank and acknowledge the anonymous reviewers for providing valuable feedback.

References

1. A. Burns and A. Wellings, *Real Time Systems and Programming Languages: Ada 95, Real-Time Java and Real-Time C/POSIX*, 3rd ed., Addison Wesley, 2001.

2. A. P. Chandrakasan, S. Sheng and R. W. Brodersen. "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 35, pp.473–484, Apr. 1992.
3. A. Chilambuchelvan, S. Saravanan and J. R. P. Perinbam. "A Simulation Software Development for Performance Analysis of DVS Algorithm for Low Power Embedded System," *ARPN Journal of Engineering and Applied Sciences*, vol. 2, pp.27–31, Aug. 2007.
4. F. Gruian, "Hard Real-Time Scheduling for Low-Energy Using Stochastic Data and DVS Processors," In Proc. the 2001 International Symposium on Low Power Electronics and Design, 2001, pp.46–51.
5. A. Hakan, R. Melhern, D. Mosse, P. M. Alvarez, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems," *Real-Time Systems*, pp.225–232, Jun. 2001.
6. R.A. Horn and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.
7. T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," In Proc. the 1998 International Symposium on Low Power Electronics and Design, 1998, pp.197–202.
8. W. Kim, D. Shin, H. S. Yun, J. Kim, and S. L. Min, "Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems," In Proc. the 8th IEEE Real-Time and Embedded Technology and Applications Symposium, 2002, pp.219–218.
9. Y. H. Tsai, K. Wang, and J. M. Chen, "A Deferred-Workload-based Inter-Task Dynamic Voltage Scaling Algorithm for Portable Multimedia Devices," In Proc. the 2007 international conference on Wireless communications and mobile computing, 2007, pp.677–682.
10. C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," Journal of the ACM (JACM), vol. 20, pp.46–61, Jan. 1973.
11. Y. Liu and A. K. Mok, "An Integrated Approach for Applying Dynamic Voltage Scaling to Hard Real-Time Systems," In Proc. the 9th IEEE Real-Time and Embedded Technology and Applications Symposium, 2003, pp.116–123.
12. A. Manzak and C. Chakrabarti, "Variable Voltage Task Scheduling Algorithms for Minimizing Energy/Power," In Proc. the 2001 International Symposium on Low Power Electronics and Design, 2001, pp.279–282.
13. D. A. Patterson and J. L. Hennessy, *Computer Architecture: A Quantitative Approach*, 1st ed., Morgan Kaufmann Pub, 1996.
14. P. Pillai and K. G. Shin, "Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating System," In Proc. the 18th ACM Symposium on Operating Systems Principles, 2001, pp.89–102.
15. 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, pp.20–30, Mar. 2001.
16. Y. Shin and K. Choi, "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems," In Proc. the 36th ACM/IEEE Conference on Design Automation , 1999, pp.134–139.
17. M. B. Srivastava, A. P. Chandrakasan, and R. W. Brodersen, "Predictive System Shutdown and Other Architectural Techniques for Energy Effecient Programmable Conputation," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 4, pp.42–55, Mar. 1996.
18. V. Swaminathan and K. Chakrabarty, "Investigating the Effect of Voltage-Switching on Low-Energy Task Scheduling in Hard Real-Time Systems," In Proc. the 2001 Conference on Asia South Pacific Design Automation, 2001, pp.251–254.
19. O. S. Unsal and I. Koren, "System-Level Power-Aware Design Techniques in Real-Time Systems," In Proc. the IEEE, 2003, pp.1055–1069.
20. M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," In Proc. the 1st USENIX conference on Operating Systems Design and Implementation, 1994, pp.13–23.
21. X. L. Zhong and C. Z. Xu, "Energy-Aware Modeling and Scheduling of Real-Time Tasks for Dynamic Voltage Scaling," In Proc. the 26th IEEE International Real-Time Systems Symposium, 2005.
22. T. Sakurai and A.R. Newton, "Alpha Power Law MOSFET Model and its applications to CMOS inverter delay and other formulas," IEEE J. of Solid-State Circuits, vol 25, no. 2, April 1990.

23. C.D. Locke, D.R. Vogel, L. Lucas, J.B. Goodenough, "General Avionics Software Specification," TR CMU/SEI-90-TR-8 ESD-TR-90-209, Software Engineering Institute Carnegie Mellon University, Pittsburgh, Dec. 1990.
24. K. De Vogeleer et. al., "The Energy/Frequency Convexity Rule: Modeling and Experimental Validation on Mobile Devices," http://arxiv.org/abs/1401.4655, Springer.
25. J. Luo and N. Jha, "Static and Dynamic Variable Voltage Scheduling Algorithms for Real-time Heterogeneous Embedded Systems," Proceedings of the 15th International Conference on VLSI Design, pp. 274-281, 2002.
26. M. Meijer and J. P. deGyvez, "Technological Boundaries of Voltage and Frequency Scaling for Power Performance Tuning," in A. Wang and S. Naffziger (eds.), Adaptive Techniques for Dynamic Processor Optimization, DOI:10.1007/978-0-387-76472-6-2, Springer, 2008.