Energy Saving EDF Scheduling for Wireless Sensors on Variable Voltage Processors

Hussein EL Ghor
Lebanese University - IUT Saida,
SNCS Research Center, UT, Saudi Arabia
B.P. 813 Saida, Lebanon
Email: hussein.ghor@ul.edu.lb

El-Hadi M Aggoune
Electrical Engineering Department,
Sensor Networks and Cellular Systems (SNCS) Research Center,
University of Tabuk, 71491 Tabuk, Saudi Arabia
Email: haggoune.sncs@ut.edu.sa

Abstract—**Advances in micro technology has led to the development of miniaturized sensor nodes with wireless communication to perform several real-time computations. These systems are deployed wherever it is not possible to maintain a wired network infrastructure and to recharge/replace batteries and the goal is then to prolong as much as possible the lifetime of the system. In our work, we aim to modify the Earliest Deadline First (EDF) scheduling algorithm to minimize the energy consumption using the Dynamic Voltage and Frequency Selection. To this end, we propose an Energy Saving EDF (ES-EDF) algorithm that is capable of stretching the worst case execution time of tasks as much as possible without violating deadlines. We prove that ES-EDF is optimal in minimizing processor energy consumption and maximum lateness for which an upper bound on the processor energy saving is derived. In order to demonstrate the benefits of our algorithm, we evaluate it by means of simulation. Experimental results show that ES-EDF outperforms EDF and Enhanced EDF (E-EDF) algorithms in terms of both percentage of feasible task sets and energy savings.**

I. INTRODUCTION

Wireless sensors have gained wide interest as a new generation of embedded systems with a broad range of real-time applications. Examples include agriculture, environment, health care, urban development, habitat monitoring, medical care, military applications [2], fire monitoring [3], volcano monitoring [4] and highway traffic coordination. Many wireless sensors are powered by batteries with limited capacity and in many scenarios it is impossible to replace them after deployment, therefore a fundamental objective is to optimize the sensor life time.

The problem of reducing energy consumption imposes additional challenges on the design of many real-time embedded systems. Such systems are characterized by a time varying processor utilization. Simply adapting the operating voltage and frequency of the processor results in improving energy efficiency and therefore battery life of wireless sensors. Dynamic voltage and Frequency scaling (DVFS) is the most well-known technique that trades off the performance for energy consumption by lowering the operating voltage/frequency [5].

In our work, we deal with dynamic scheduling for uniprocessor systems that support periodic tasks. EDF has been shown to be an optimal dynamic scheduling algorithm in the sense that if a set of tasks can be scheduled by any algorithm, then it can be scheduled by EDF [6]. EDF algorithm is typically preemptive, in the sense that, a newly arrived task may preempt the running task if its absolute deadline is shorter. This dynamic priority assignment allows EDF to exploit the full processor, reaching up to 100% of the available processing time [7].

Already having the EDF scheduler, it was only necessary to find a way to reduce energy consumption of tasks so as to prolong as much as possible the lifetime of the system. Dynamic voltage and frequency scaling (DVFS) is the most efficient technique for reducing CPU energy. It is feasible to run the processor at the weakest frequency while still admiring the deadlines of tasks. In other words, when the frequency is reduced, the processor can operate at a lower supply voltage and so reducing the energy consumption. However, when reducing the processor speed, tasks must take more time to complete their execution. Therefore, it is important to identify the slack time under which we can safely slow down the processor without missing any deadline.

In this paper, we present an approach to find the least-energy voltage schedule for executing real-time tasks on a DVFS processor according to a dynamic priority, preemptive policy, denoted by Energy Saving EDF (ES-EDF). For the minimization of energy consumption, we use DVFS technique that reduces the processor energy by slowing down the DVS processor and stretching the task execution time. We propose a slack-based method for stretching tasks as much as possible while still guaranteeing deadlines. Off-line computing by how long the tasks should be stretched is possible thanks to EDL properties [8].

The rest of the paper is organized as follows. The paper begins with a summary of the related work. Section III defines the system model and terminology used throughout this paper. The necessary background is presented in section IV. In section V, we propose the Energy Saving EDF (ES-EDF) algorithm. In section VI, we present the feasibility analysis for our proposed algorithm. An upper bound on energy savings is derived in section VII. The simulation results for performance evaluation are presented in section VIII. The paper is concluded in section IX.

II. RELATED WORK

The majority of real-time schedulers are on-line and based on the concept of priority. If the priority is fixed at the initialization for all tasks, the algorithm is called fixed priority algorithm. Rate monotonic scheduling (RM) [9] and deadline monotonic scheduling (DM) [11] are examples of such algorithms. If it evolves over time, the algorithm is said to be driven by a dynamic priority. The most known algorithm among such scheduling approaches is the Earliest Deadline First (EDF) algorithm [7]. The study reported in this work deals with dynamic priority scheduling, preemptive and without resource and precedence constraints.

Recently, researchers have started exploring energy-efficient scheduling for real-time embedded systems such as wireless sensors. Algorithms proposed in literature have either dynamic or fixed priority, also they can be preemptive or non-preemptive. Although DVS is one of the most important techniques, still some of them consider non-DVS techniques especially when the study of the energy consumption is for processor and devices.

Among the earliest works, Yao et al. [13] proposed an optimal offline-scheduling algorithm for independent set of jobs to minimize energy consumption. The same problem was targeted in [14], but for dependent tasks. Authors have proposed a scheduling algorithm using a variable voltage processor core.

Shin and Choi [22] proposed a power reduction technique for a processor by exploiting the slack times inherent in the system and those arising from variations of execution times of task instances. In this technique, the processor can either be shut down if there is no current active job or adopt the speed such that the current active job finishes at its deadline or the release time of the next job. Later in [24], the same authors have proposed a method that combines an off-line component with an on-line one. By applying this method, they first determine the lowest possible maximum processor speed where the task set is feasible while all deadlines are met. These tasks apply the WCET at all times and consequently some idle time will be obtained. An on-line component is then introduced that can dynamically reduce the processor speed according to the status of task set in order to exploit execution time variations and idle intervals, the only situation a task is stretched is when it is the only one running and has enough time until the next task arrives.

With the aim to find the least energy schedule for executing real-time tasks, authors in [15] proposed an optimal fixed priority policy. Later in [16], Kwon et al. presented important results for task scheduling over a fixed number of voltage levels.

Based on exploiting slack times, Aydin et al. [17] have addressed the problem of minimizing energy by proposing a dynamic speculative scheduling algorithm. Later in [18], authors have considered that task deadlines are different from the task period. Under this assumption, authors have addressed the problem of computing task slowdown factors. Recent works have proposed further dynamic voltage scaling techniques to enhance the energy gains at run-time [19] [20] [21].

For scheduling periodic real-time tasks on a variable speed processor with realistic discrete speeds, Mejia et al. [25] have proposed a heuristic algorithm that finds near-optimal solutions at low cost. This method produces a 2-approximate solution to the optimization problem. Later in [26], authors proposed a polynomial time $(1 + \epsilon)$-approximation algorithm for the scheduling of periodic real-time tasks, where ϵ is the tolerable error margin given by users $(0 < \epsilon < 1)$.

Recently, reliability became as important as energy efficiency especially in real-time embedded systems like satellite and monitoring systems. Following this idea, several scheduling policies have been proposed for various task models. Zhu et al. [28], [27] proposed a reliability-aware power management (RAPM) algorithm for periodic real-time tasks that can study the negative effects of voltage scaling on system reliability. This work was later extended in [29], authors improved the quality of assurance for all tasks by managing only a subset of jobs from each task.

In a preemptive scheme certain low priority tasks may be suspended if higher priority tasks need to be executed. This will lead to a more flexible scheme but with a certain time overhead. Jejurikar et al. [30] focused on the system level power management via the computation of static slowdown factors under synchronization constraints where tasks are scheduled based on a preemptive scheduling policy. For a similar task model, authors in [31] proposed the concept of frequency locking and extended the Priority Ceiling Protocol (PCP) by locking the processor frequency in a restricted way, so that the cost in frequency switching is better managed. The major inconvenient is that frequency switching is shown to be not found. This work was later extended in [32] by avoiding voltage emergency. Authors explored one of the pioneering real-time task synchronization with the minimization of energy consumption and voltage emergency prevention.

III. System Model and Terminology

A. Task Set

The real-time system considered in this work consists of two major units: Real-time Operating System and the Storage unit. The considered RTOS is equipped with a DVFS-enabled processor. The variable speed processor is assumed to be working with N discrete frequencies ranging from $f_{min} = f_1 \leq f_2 \leq \cdots \leq f_n = f_{max}$. The power consumption of the tasks running in the processor and frequency levels are in a way coupled together. When we change the speed of a processor, its operating frequency is changed and hence the power consumption of tasks is proportionately changed the voltage to a value which is supported at that operating frequency. We denote by P_n and V_n respectively the power consumption and voltage level correspondent to clock frequency f_n. We consider that P_n is the overall power consumption of the RTOS. This means that P_n is a combination of both dynamic power consumption and leakage power consumption. We also ignore the time and energy overhead incurred in changing the frequency and voltage of the processor.

We use the term slowdown factor S_n as the ratio of the scheduled speed to the maximum processor speed. S_n ranges from S_{min} to 1:

$$S_n = \frac{f_n}{f_{max}} \qquad (1)$$

We consider in our work that each job has different power dissipation that varies according to its frequencies. Consequently, a task will have maximum power dissipation at its maximum frequency and this power consumption decreases as the frequency decreases. Consequently, the power dissipation of a task must be defined as function of the task index and its corresponding slowdown factor $P_i(\tau_i, S_i)$.

Any application executed on this RTOS is normally composed of multiple tasks with different levels of priority. We consider here independent and preemptive periodic tasks. A task set Γ of n tasks is denoted as follows: $\Gamma = \{\tau_i \mid 1 \leq i \leq n\}$. τ_i is characterized by four-tuple (r_i, C_i, D_i, T_i) where r_i, C_i, D_i and T_i indicate the release time, the worst case execution time (WCET), the relative deadline and the period respectively. Release time r_i of task τ_i is equal to kT_i, $k = 0, 1, 2, \cdots$. We assume that $0 \leq C_i \leq D_i \leq T_i$ for each $1 \leq i \leq n$.

When a task τ_i is stretched by a slowdown factor S_i, then its actual execution time $(C_i(a))$ at frequency f_i will be C_i/S_i. When the processor is running at it maximum frequency, then $C_i(a) = C_i$. The energy dissipation (E_i) of a task τ_i is computed as:

$$E_i = P_i(\tau_i, S_i) \times (C_i/S_i) \tag{2}$$

B. Energy Storage

Our RTOS relies on an ideal energy storage unit, battery for example, that has a nominal capacity, namely E, corresponding to a maximum energy (expressed in Joules or Watts-hour). The energy level has to remain between two boundaries E_{min} and E_{max} with $E = E_{max} - E_{min}$.

We denote by $E(t)$, the energy stored in the battery at time t. At any time, the stored energy is no more than the storage capacity, that is

$$E(t) \leq E \quad \forall t \tag{3}$$

IV. Necessary Background

The main objective behind a scheduling algorithm is to determine, for a given set of jobs, the order in which tasks are to be executed [23]. In real-time systems, the main goal of the scheduling algorithm is to complete the execution of all jobs while guaranteeing their deadlines. Before we get into the details of the scheduler implementation it is important to understand some of the more important real-time scheduling approaches, namely fixed-priority algorithms, including rate monotonic [9] and deadline monotonic [11], and dynamic-priority algorithms, including the earliest deadline first (EDF) algorithm [10]. EDF schedules at each instant of time t, the ready task (i.e. the task that may be processed and is not yet completed) whose deadline is closest to t. EDF is an optimal scheduling algorithm on preemptive uniprocessors. The EDF algorithm can achieve an utilization of 100% of the available processing time. The processor utilization of a system is computed as follows:

$$U_p = \sum_{i=1}^{n} \frac{C_i}{T_i} \tag{4}$$

A periodic task set with deadlines equal to periods is schedulable by EDF if and only if the total processor utilization U_p is less than or equal to one [9].

A. Static EDS Scheduling

The implementation of EDF consists in ordering tasks according to their priority and executing them as soon as possible with no inserted idle time. Such implementation is known as Earliest Deadline as Soon as possible (EDS) [12]. For a given periodic task set, the EDS schedule can be pre-computed and memorized in order to reduce scheduling overheads at run time.

B. Static EDL Scheduling

EDL algorithm is based on the notion of delaying the execution of jobs by as much as possible without causing their deadlines to be missed. Although the usual scheduling scheme is EDS, EDL is very often considered for processor idle time analysis. In [12], Chetto and Chetto presented a simple method to determine the location and length of idle time in any window of a sequence generated by the two different implementations of EDF and EDL.

Before the system begins to operate, static EDL schedule is computed for a given task set. More precisely, the duration and position of the idle times is determined by mapping out the EDL schedule produced from time zero up to the end of the first hyperperiod. Let T_{LCM} the hyperperiod be equal to the least common multiple of the task periods where $T_{LCM} = lcm(T_1, T_2, \cdots, T_n)$. Hence, determining the EDL schedule for the interval $[0, T_{LCM}]$ is realized by means of the two following vectors [8]:

Static deadline vector $\mathcal{K}$: it represents the times at which idle times occur within the first hyperperiod. $\mathcal{K} = \{k_0, k_1, \cdots, k_i, k_{i+1}, \cdots, k_q\}$ where $k_0 = 0$, $x_i = T_i - D_i$ and $k_i < k_{i+1}$ for all $1 \leq i \leq n$.

Let us consider $q \leq N + 1$ where N denotes the number of jobs within the first hyperperiod. Then k_q is equal to

$$k_q = T_{LCM} - min\{x_i \mid 1 \leq i \leq n\} \tag{5}$$

Static idle time vector $\mathcal{D}$: it represents the lengths of the idle times which start at time instants given by $\mathcal{K}$.

$\mathcal{D} = (\Delta_0, \Delta_1, \cdots, \Delta_i, \Delta_{i+1}, \cdots, \Delta_q)$. Δ_i corresponds to the length of the idle time that starts at time k_i.

Vector $\mathcal{D}$ is defined by a recurrent formula as follows:

$$\Delta_q = min\{x_i \mid 1 \leq i \leq n\} \tag{6}$$

$$\Delta_i = max(0, F_i), \ where\ i = q - 1\ down\ to\ 0 \tag{7}$$

$$F_i = (T_{LCM} - k_i) - \sum_{j=1}^{n} \lceil \frac{T_{LCM} - x_j - k_i}{T_j} \rceil C_j - \sum_{k=i+1}^{q} \Delta_k \tag{8}$$

Where $\lceil y \rceil$ is the least integer greater than or equal to y.

Under energy constraints, executing the jobs as late as possible within the time interval $[0, T_{LCM}[$ consists in first ordering the jobs according to the EDF rule and second stretching the execution time C_i by its corresponding Δ_i.

Illustrative Example: Consider a periodic task set Γ that is composed of three tasks, $\Gamma = \{\tau_i \mid 1 \leq i \leq n\}$ and $\tau_i = (C_i, D_i, T_i)$. Let $\tau_1 = (1,3,5)$, $\tau_2 = (2,7,10)$ and $\tau_3 = (3,12,20)$.

From formulae 6, 7 and 8, we have $\mathcal{K} = (0,3,7,8,12,13,17,18)$ and $\mathcal{D} = (2,2,0,1,0,2,0,2)$. The EDL schedule for Γ produced at the first hyperperiod is described in figure 1.

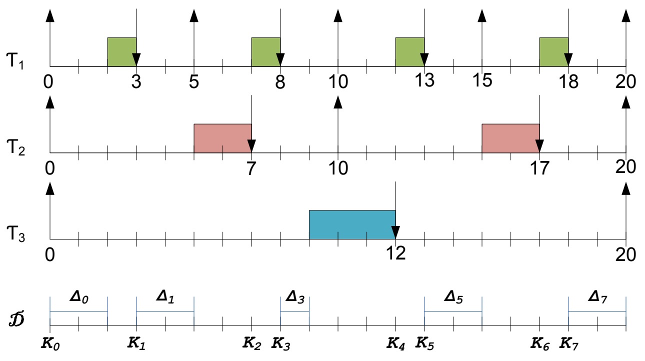


Fig. 1. Static EDL scheduling of task set Γ

V. Energy Saving - EDF (ES-EDF) Scheduling Algorithm

In this section, we describe the ES-EDF task scheduling algorithm that minimizes the CPU energy. In ES-EDF, an optimal voltage and frequency for a given task instance is computed that guarantees schedulability and minimizes energy consumption.

A. Presentation of the Algorithm

The aim from ES-EDF is to develop dynamic task scheduling algorithm based on EDF that schedule any periodic task set feasibly, and minimize the CPU energy consumption. We use a modified Earliest Deadline First (EDF) strategy to reduce the CPU energy consumption by using the Dynamic Voltage and Frequency Selection. For this sake, We propose a slack-based method for stretching tasks as much as possible while still guaranteeing deadlines. Off-line computing by how long the tasks should be stretched is possible thanks to EDL properties.

B. Slack Time Method

In the following analytical analysis, we determine the maximum scaling time of a DVFS system that results in minimum energy consumption for the processor and without any deadline violation. That is, we have to determine the lowest maximum processor speed to execute a real-time task set on a variable speed processor while guaranteeing the deadlines of tasks. This scaling time for each task instance is called the *optimal scaling time*.

Our approach is based on the assumption that the parameters of each task is well known off-line and that all tasks are released at time $t = 0$. This assumption is very important since otherwise we may not be able to fully utilize the benefits provided by the used variable speed processor.

As an initial schedule, we try to execute all task instances according to the earliest deadline first strategy. Let us consider that there are M task instances in the ready queue. The start time and finish time of task τ_i are represented by St_i and Ft_i respectively.

Assume that the start time of the first task instance τ_1 in the ready queue is equal to its release time.

$$St_1 = k_0 = 0 \tag{9}$$

In order to stretch the execution time C_i of task τ_i as much as possible without violating the deadline D_i, the determination of the latest start time for every task instance requires preliminary construction of the schedule produced by the so-called Earliest Deadline as Late as possible (EDL) algorithm.

To involve an acceptable overhead at run-time, off-line computations are done by ES-EDF in order to compute efficiently the static EDL schedule without losing any time. Before the system begins to operate, we estimate the localization and the duration of the idle times produced at time $t = 0$ till the end of the hyperperiod. This means that ES-EDF computes the static deadline vector $\mathcal{K}$ and static idle time vector $\mathcal{D}$.

Thus, the start time of the remaining task instances is:

$$St_i = k_{i-1} \tag{10}$$

The total time executed by the task set within $[0, St_i]$ is denoted by A_k where

$$A_k = \sum_{D_k \leq St_i} C_k \tag{11}$$

Consequently, the finish time of the remaining task instances is:

$$Ft_i = C_i + \sum_{k_j \leq St_i} \Delta_j + A_k \tag{12}$$

where $1 \leq i \leq M - 1$

To decrease the processor speed as much as possible and as long as the system will be able to meet all the deadlines, we have to compute the static idle time vector Δ_i at each task start time St_i. Consequently, the task's execution time will be stretched to the actual execution time $C_i(a)$ where:

$$C_i(a) = C_i + \Delta(St_i) \tag{13}$$

The slowdown factor is thus calculated thanks to the following equation:

$$S_i = \frac{C_i}{C_i(a)} \tag{14}$$

C. ES-EDF Algorithm

The ES-EDF works as follows: First ES-EDF computes the static EDL schedule including static deadline vector $\mathcal{K}$ and static idle time vector $\mathcal{D}$. Before authorizing the execution of the task instance with highest priority, ES-EDF adds the static idle time at St_i to the execution time C_i. Now, the execution time of the task instance will be stretched to its actual execution time ($C_i(a)$) without violating deadlines. Upon stretching the execution time of a task instance, the energy dissipation decreases. ES-EDF can now compute the slowdown factor of the corresponding task instance and consequently the energy dissipation can be chosen.

The major components of ES-EDF are: $E(t)$, $\mathcal{K}$ and $\mathcal{D}$ where t is the current time, $E(t)$ the amount of energy that is currently stored at time t that means the remaining amount of energy in the energy storage at time t. $\mathcal{K}$ and $\mathcal{D}$ are respectively the static deadline vector and the static idle time vector. Moreover, we use the function $execute()$ to put the processor to run the ready job with the earliest deadline.

We describe in algorithm 1 the pseudo code of the ES-EDF scheduler:

Algorithm 1 Energy Saving - Earliest Deadline First (ES-EDF) Algorithm

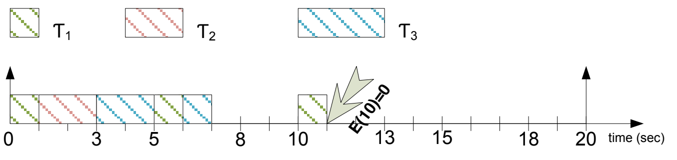
```
Require: A Set of $M$ periodic Tasks $\Gamma = \{\tau_i | \tau_i = (r_i, C_i, D_i, T_i, E_i) \quad i = 1, \cdots, M\}$ According to $EDF$, current time $t$, battery with capacity ranging from $E_{max}$ to $E_{min}$, energy level of the battery $E(t)$.
Require: A processor working with $N$ discrete frequencies ranging from $f_1$ ($f_{min}$) to $f_N$ ($f_{max}$).
Ensure: $ES-EDF$ Schedule.
 1: Sort task instances according to the EDF rule
 2: Determine the start time of task instances
 3: Compute the static EDL vectors $\mathcal{K}$ and $\mathcal{D}$
 4: for i=1:M do
 5:     if i==1 then
 6:         $St_1 = k_0 = 0$
 7:     else
 8:         $St_i = k_{i-1}$
 9:     end if
10: end for
11: while $E(t) > 0$ do
12:     Actual Execution Time $C_i(a) = C_i + \Delta(St_i)$
13:     $Ft_i = St_i + C_i + C_i(a)$
14:     Slowdown factor $S_i = C_i/C_i(a)$
15:     Update execution time
16:     Select the relative Energy Consumption ($E_i$)
17:     Calculate the remaining energy in the battery at the end of the execution.
18:     $E(Ft_i) = E(St_i) - E_i(St_i, Ft_i)$
19:     execute()
20:     Remove task $\tau_i$ from ready task list
21: end while
```

D. Illustrative Example

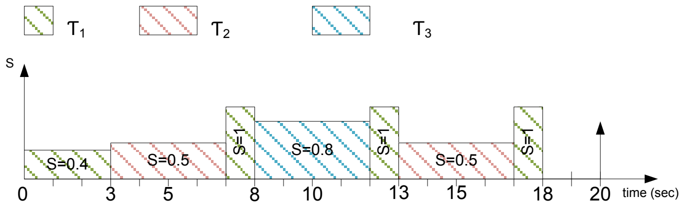
Consider a periodic task set Γ that is composed of three tasks, $\Gamma = \{\tau_i \mid 1 \leq i \leq n\}$ and $\tau_i = (C_i, D_i, T_i)$. Let $\tau_1 = (1, 3, 5)$, $\tau_2 = (2, 7, 10)$ and $\tau_3 = (3, 12, 20)$. We assume that the energy storage capacity is $E = 350$ energy units at $t = 0$. The processor is assumed to be working with ten discrete slowdown factors $S_i = \{1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$. The power dissipation of tasks τ_i is shown in table I:

First of all, we have to schedule Γ according to EDF within the first hyperperiod, from 0 to 20. We verify that Γ is not schedulable since the battery capacity is equal to zero at $t = 11$ and consequently the deadline miss rate is about 30%. In details:

At time $t = 0$, all tasks are ready. τ_1 is the highest priority task and is executed until $t = 1$ where $E(1) = 320$ energy units. At time $t = 1$, τ_2 is the highest priority task and is executed until $t = 3$ where $E(3) = 240$ energy units. τ_3 is now the highest priority task and is executed until $t = 5$ where it is preempted by τ_1. τ_3 resumes its execution at time $t = 6$ and is completed at $t = 7$ where $E(7) = 30$ energy units. The remaining energy in the battery unit is sufficient only to execute τ_1 until $t = 11$. Now, the battery is empty and consequently the scheduling is terminated where the deadline miss rate is about 30% (3(a)).



(a) EDF Scheduling



(b) ES-EDF Scheduling

Fig. 2. Scheduling of task set Γ

To increase the efficiency of the processor and decrease the energy consumption, let us try scheduling the same task set Γ but with ES-EDF. We find that Γ is schedulable since all tasks are executed without violating deadlines and without getting out of energy. Let us explain how ES-EDF constructs the schedule (3(a)). Before beginning the execution of tasks, ES-EDF computes $\mathcal{K}$ and $\mathcal{D}$. $\mathcal{K} = (0, 3, 7, 8, 12, 13, 17, 18)$ and $\mathcal{D} = (2, 2, 0, 1, 0, 2, 0, 2)$.

At time 0, the residual capacity i.e. remaining energy is maximum since the storage is full. τ_1 is the highest priority task where $\Delta_0 = 2$. Thus, the actual execution time for τ_1 is equal to three and the slowdown factor S_1 is $S1 = 1/3 = 0.33$. Consequently, the the energy dissipation for τ_1 is $E_1 = 12$ (see table I). Now, τ_1 is executed from $t = 0$ to $t = 3$ with a slowdown factor $S_1 = 0.33$ where $E(3) = 388$ energy units.

τ_2 has now the highest priority and it begins its execution at time $t = k_1 = 3$. Its execution time is stretched until $t = C_2 + \Delta_1 = 4$. The slowdown factor is then $S_2 = 0.5$ and the residual capacity at time $t = 7$ is $E(7) = 298$ energy units.

At time 7, $\Delta_2 = 0$. Consequently τ_1 must be executed at full processor speed. τ_1 executes completely until time 8 and consumes 30 energy units. The residual capacity then equals 268 energy units.

This procedure continues until $t = 18$ where no task requires to be processed in the interval $[18, 20]$. At the end of the hyperperiod, the battery capacity is equal to 28 energy units.

TABLE I. ENERGY DISSIPATION OF TASKS τ_i

Energy Dissipation	$S = 1$	$S = 0.9$	$S = 0.8$	$S = 0.7$	$S = 0.6$	$S = 0.5$	$S = 0.4$	$S = 0.3$	$S = 0.2$	$S = 0.1$
Task τ_1	30	27	24	21	18	15	12	9	6	3
Task τ_2	80	72	65	56	48	40	33	25	16	8
Task τ_3	180	160	140	125	110	90	70	55	36	18

VI. FEASIBILITY ANALYSIS

In contrast to EDF, ES-EDF feasibly schedules the task set in the first hyperperiod, with the same characteristics of the storage unit and the processor. It is important here to note that the processor remains busy as long as there are ready tasks in the queue. This means that the processor is busy from $[0, k_q]$. Thus, the processor utilization is equal to $1-\Delta_q/T_{LCM} = 0.9$. This leads to deduce that ES-EDF is optimal with respect to minimizing processor energy.

THEOREM *1:* A task set Γ that is schedulable with EDF remains schedulable with ES-EDF.

Proof: Let us consider the U_p and U'_p as processor utilization of EDF and ES-EDF respectively. If Γ is schedulable by EDF, then $U_p \leq 1$. We have to prove that if $U_p \leq 1$, then $U'_p \leq 1$.

Suppose that Γ is not schedulable by ES-EDF, then $U'_p > 1$. But $U'_p = \sum_{i=1}^{M} \frac{C_i(a)}{T_i} = \sum_{i=1}^{M} \frac{C_i+\Delta(St_i)}{T_i} = U_p + \sum_{i=1}^{M} \frac{\Delta(St_i)}{T_i}$. This implies that $U_p + \sum_{i=1}^{M} \sum_{j=1}^{q} \frac{\Delta_j}{T_i} > 1$. By multiplying the whole equality by T_{LCM}, we get $T_{LCM}U_p + \frac{T_{LCM}}{\sum_{i=1}^{M} T_i} \sum_{j=1}^{q} \Delta_j > T_{LCM}$.

But since Γ is periodic, then $\frac{T_{LCM}}{\sum_{i=1}^{M} T_i} \sum_{j=1}^{q} \Delta_j = t_{idle}$, where t_{idle} is the total idle time. By substitution, we get $T_{LCM}U_p + t_{idle} > T_{LCM}$, then $t_{idle} > T_{LCM}(1 - U_p)$

According to EDL, if Γ is schedulable, then the total idle time during a whole hyperperiod T_{LCM} is equal to $T_{LCM}(1-U_p)$. Contradiction. Consequently, U'_p must be less than or equal to one. ■

COLLABORY *1:* The processor utilization U'_p for ES-EDF is equal to $1 - \frac{min(x_i)}{T_{LCM}}$ where $x_i = T_i - D_i$.

Proof: Let $\mathcal{K}$ and $\mathcal{K}$ be respectively the static deadline vector and the static idle time vector, as defined in IV-B. According to ES-EDF, the busy period within the initial window , denoted $W(l)$, is equal to K_q where K_q is the latest component of $\mathcal{K}$. We note that $\Delta_q = T_{LCM} - k_q$ so that the length of busy period at k_q is zero. As no task requires to be processed in such interval, the collabory is true. ■

THEOREM *2:* Given a set of independent tasks with arbitrary computation times, deadlines and periods, ES-EDF is optimal with respect to minimizing the maximum lateness.

Proof: Given a set of independent tasks τ_i with arbitrary computation times, deadlines and periods. According to Horn [6], any schedule that puts the jobs in order of non-decreasing deadlines minimizes the maximum lateness. Alternatively, we concern ourselves with constructing the ES-EDF schedule that complete all tasks by their deadlines. To capture this, given the ES-EDF schedule that at every scheduling instant k_i executes the task with the earliest absolute deadline among all the ready tasks and stretches its computation time C_j until its deadline D_j. Consequently, ES-EDF minimizes the maximum lateness. What is left in this theorem is the optimality criterion.

The maximum lateness is defined in [7] as $L_{max} = max_i(f_i - D_i)$ where f_i is the finish time of the i^{th} task. We can show that ES-EDF has a maximum lateness equal to zero. Let us consider τ_1 to be the ready task with the highest priority at $t = k_0 = 0$. According to ES-EDF, τ_1 will be executed until $t = D_1$ and hence its lateness is zero. If we repeat the above schedule until there are no more tasks in the queue, then we have constructed the ES-EDF schedule. Since this schedule has a maximum lateness $L_{max} = 0$, the ES-EDF schedule is optimal with respect to minimizing the maximum lateness. This optimum processing utilization is equal to $1 - \frac{\Delta_q}{T_{LCM}}$ where T_{LCM} is the period, $\Delta_q = min\{x_j \mid 1 \leq j \leq n\}$ and $x_j = T_j - D_j$. ■

VII. UPPER BOUND ON ENERGY SAVINGS

Scaling the processor frequency and voltage based on the performance requirements can lead to considerable energy savings. This is due to the quadratic relationship between voltage and dynamic power. In addition, the processor slowdown is based on slowdown factors S_i in such a way that energy savings increase with decreased slowdown factor. However, the maximum energy savings possible by the any dynamic voltage and frequency selection algorithm must be bounded by the amount of energy savings with respect to the minimum slowdown factor (processing rate).

THEOREM *3:* [33] A set of periodic tasks is guaranteed to be schedulable with maximum energy savings iff the processing rate is

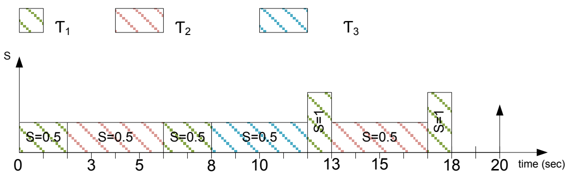
$$r_{min} = \sum_i \frac{C_i}{T_i} \tag{15}$$

From this theorem, authors concluded that the maximum energy savings occurs when all tasks have the same averaged processing rate. This means that the minimum processor energy consumption occurs when all tasks are slacked by the same amount, to the maximum allowable limit such that $r_{min} = \sum_i \frac{C_i}{T_i}$.

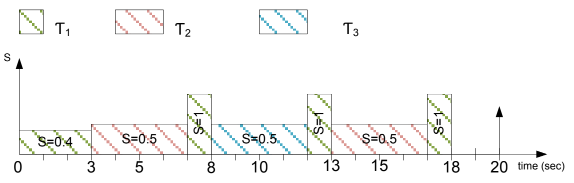
Unfortunately, this conclusion is not always evident. To prove this, let us consider the following illustrative example: Consider a periodic task set Γ that is composed of three tasks, $\Gamma = \{\tau_i \mid 1 \leq i \leq n\}$ and $\tau_i = (C_i, D_i, T_i)$. Let $\tau_1 = (1, 3, 5)$, $\tau_2 = (2, 7, 10)$ and $\tau_3 = (2, 12, 20)$. We assume that the energy storage capacity is $E = 350$ energy units at $t = 0$. The processor is assumed to be working with ten discrete slowdown factors $S_i = \{1, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1\}$. The power dissipation of tasks τ_i is shown in table I.

In order to demonstrate our theory, we have to schedule Γ according to the assumption in [33] and then according to different processing rates as in ES-EDF.

Following the assumption in [33], the minimum processing rate (slowdown factor) is equal to $r_{min} = \sum_i \frac{C_i}{T_i} = 0.5$ and consequently, the computation time for each task instance is doubled (figure 3(a)). Since the same static slowdown factors are used, Γ is scheduled till the end of the hyperperiod where the battery capacity is equal to 90 energy units.



(a) Scheduling according to same slack amount



(b) ES-EDF Scheduling

Fig. 3. Scheduling of task set Γ

Now, let us try scheduling the same task set Γ but with ES-EDF (figure 3(b)). We find that Γ is schedulable since all tasks are executed without violating deadlines and the remaining energy in the battery at the end of the hyperperiod is equal to 108 energy units.

From this example, we can deduce that ES-EDF feasibly schedules the task set in the first hyperperiod and with more energy saving than when considering the same averaged processing rate.

THEOREM *4:* A set of periodic tasks is guaranteed to be schedulable with maximum energy savings iff the optimal slowdown factor is approximated by

$$S_{opt} = (1 - \frac{x_q}{T_{LCM}})U_p \tag{16}$$

Proof: It has been proved in [9] that a periodic task set is guaranteed to be schedulable by EDF *iff* $\sum_i \frac{C_i}{T_i} \leq 1$. Theorem 3 demonstrated that the maximum allowable limit, in case of equal slowdown factors, is bounded by $E_{save}(S_{min})$ where the minimum slowdown factor $S_{min} = \sum_i \frac{C_i}{T_i}$. We will now show the minimum energy consumption under different slowdown factors. Let us consider S_i and S_{opt} as the slowdown factor of task instance τ_i and the optimal slowdown factor respectively. From equation (14), $S_i = \frac{C_i}{C_i(a)}$. We can derive the condition that S_{opt} has to satisfy in order to guarantee the schedulability of the task set, then $S_{opt} = \sum_i \frac{C_i}{C_i(a)}$. But since tasks are periodic, then we have to multiply by $\frac{T_{LCM}}{\sum_i T_i}$. Therefore, the average of slowdown factors is equal to

$$S_{opt} = \frac{1}{T} \sum_i \frac{C_i}{(C_i(a))(T_i)} \tag{17}$$

Now, let's consider the following inequality that can be verified using the Cauchy-Schwarz inequality

$$(\sum_i \frac{C_i}{(C_i(a))(T_i)})(\sum_i C_i(a)) \geq \sum_i \frac{C_i}{T_i} \tag{18}$$

By making some arrangement and distribution of the terms, we get

$$S_{opt} \geq (1 - \frac{x_q}{T_{LCM}})U_p \tag{19}$$

Then, the minimum speed S_{opt} that ensures feasibility is $S_{opt} = (1 - \frac{x_q}{T_{LCM}})U_p$.

■

VIII. PERFORMANCE EVALUATION

This section provides performance evaluation of the algorithms. Algorithms under simulations are ES-EDF, EDF and Enhanced EDF (E-EDF). E-EDF is an enhanced version of EDF in a way that tasks are slacked by the same slowdown factor $S = \sum_i \frac{C_i}{T_i}$.

A. Experimental Setup

We implemented the proposed scheduling techniques in a discrete event simulator using C/C++. To evaluate the effectiveness of the ES-EDF algorithm, we consider a task generator of periodic tasks based on that described by Martineau in [34]. It accepts as input several parameters: the number of desired tasks n, the hyperperiod of task periods T_{LCM} and processor utilization U_p. At the output, we obtain a task configuration of the scheduled task set. The execution times of tasks are randomly generated such that $\sum_i \frac{C_i}{T_i} \leq 1$. The simulator generates 30 tasks with least common multiple of the periods equal to 3360. The worst-case computation times are set according to the processor utilization U_p. Deadlines are less than or equal to periods and greater than or equal to the computation times ($C_i \leq D_i \leq T_i$).

To estimate the processor energy consumption, we use Intel XScale processor supporting five frequency levels [35]. The discrete frequencies, supply voltage and consumed power the processor are listed in Table II. We assume that the energy storage is fully charged at the beginning of the simulation. After a deadline violation is detected, the simulation terminates for ES-EDF, EDF and E-EDF.

TABLE II. XSCALE FREQUENCIES, SUPPLY VOLTAGES, AND POWER

Frequency (MHz)	150	400	600	800	1000
Power (mW)	80	170	400	900	1600
Voltage (V)	0.75	1.0	1.3	1.6	1.8

B. Percentage of Feasible Task Sets by Varying U_p

Our simulation depicts the percentage of feasible task sets by varying the processor utilization U_p. Here, we take interest in the percentage of task sets which are feasible with ES-EDF, EDF and E-EDF. We report the results of this simulation study where U_p varies from 0.1 till 1 (figure 4).

Under low processor utilization, we observe that ES-EDF maintain 100% of feasible task sets, and exceeds that of E-EDF and EDF by about 31% and 49%. This is due to the

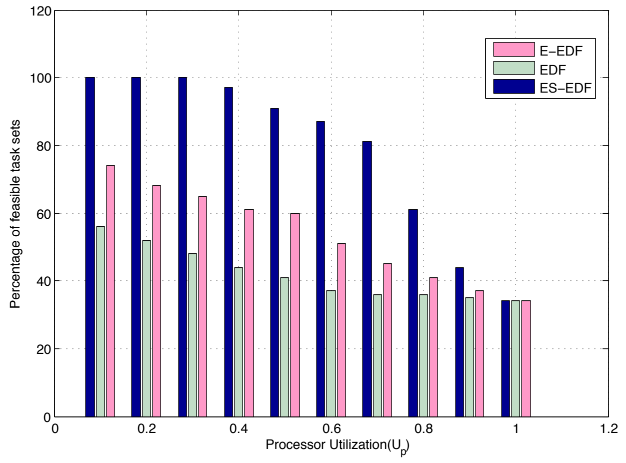


Fig. 4. Percentage of Feasible Task Sets by Varying U_p

fact that the EDF algorithm runs at full processor speed and does not utilize DVS technique to save energy. On the other hand, in E-EDF, tasks are slacked with equal slowdown factor and consequently not all deadlines will be met. Hence, ES-EDF system significantly reduces the deadline miss rate due to energy shortage. In other words, the more stored energy means that more tasks are able to be finished before their deadlines.

As U_p increases, the percentage of feasible task sets in ES-EDF decreases. This is because ES-EDF utilizes the slack time to slow down the execution for energy savings. Under high values of U_p, slack time decreases and most of tasks are just executed at the full speed, as EDF algorithm does. Hence, the ES-EDF system incurs much higher deadline miss rate but still exceeds that of E-EDF and EDF by about 34% and 46%.

It is important to note that when the processor utilization is set to one, ES-EDF, E-EDF and EDF has exactly the same percentage of feasible task sets. This is because the processor is always active and there is no processor idle time.

C. Percentage of Feasible Task Sets by Varying Battery Capacity

This experiment depicts the percentage of feasible task sets over the energy storage capacity E. Here, we take interest in the percentage of task sets which are feasible with ES-EDF and not feasible with E-EDF and EDF. We report the results of this simulation study where the processor utilization U_p is set to 0.4 and 0.8 respectively.

For each task set, we compute E_{feas} as the minimum storage capacity which permits to achieve neutral operation according to ES-EDF. i.e. all tasks are executed without violating deadlines and the battery is not empty at the end of the hyperperiod. After that, we vary E with $E > E_{feas}$ so as all task sets are feasible with E-EDF and EDF.

When U_p is set to 0.4 (figure 5 (a)), we observe that the battery capacity must be about 2.2 and 3.6 times bigger with E-EDF and EDF to maintain 100% feasible task sets compared to ES-EDF. This is due to the fact that EDF algorithm runs at full processor speed and does not utilize DVS technique to save energy. Thus, the SE-EDF and E-EDF systems significantly reduce the deadline miss rate due to energy shortage. Unfortunately, E-EDF considers equal slowdown factors and this will reduce the percentage of feasible task sets.

In other words, the more stored energy means that more tasks are able to be finished before their deadlines. Hence, the ES-EDF system incurs much higher deadline miss rate when compared to E-EDF and EDF.

Figure 5 (a) shows that the deadline miss rate of E-EDF and EDF exceeds that of the proposed scheduling algorithm by about 45% and 61% when battery capacity is respectively the same. Hence, the ES-EDF algorithm is favorable even for small battery capacity.

If we increase the processor utilization to 0.8 and run the simulation again, we find that the gain in capacity savings is decreasing (figure 5 (b)). ES-EDF obtains capacity savings of about 24% and 41% compared to E-EDF and EDF respectively. The decrease in capacity savings can be attributed to the fact that as U_p increases, the slacking gets harder and the SE-EDF schedule tends to the EDF schedule with processor utilization increasingly being set to 1.

D. Ratio of Energy Savings

We present in this section the energy gains achieved by ES-EDF when compared to E-EDF. Experiments were performed on task sets with varying processor utilization (U_p). Figure 6 shows the normalized energy gains for ES-EDF and EDF, that means the quantity of energy gains relative to battery capacity.

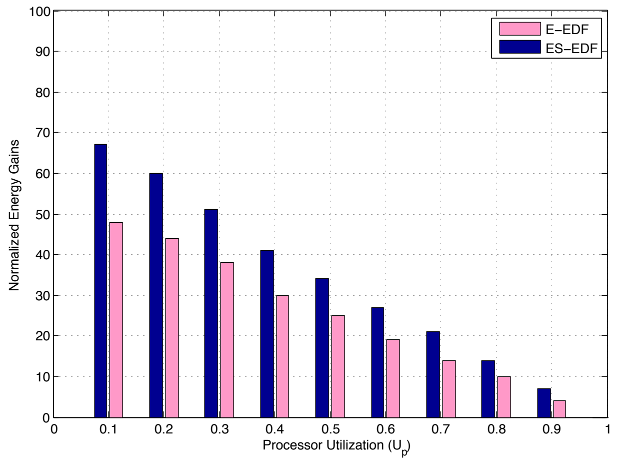


Fig. 6. Ratio of Energy Savings of ES-EDF, E-EDF and EDF

From the beginning, we state that there are no energy gains in EDF since it operates at maximum processor frequency. As for ES-EDF and E-EDF, when the task execution time is decreased, there is more slack which results in decreasing the energy consumption by operating at a lower voltage. Furthermore, the average energy gains in ES-EDF is about 28% more than E-EDF. This is due to the fact that ES-EDF stretches tasks as much as possible while still guaranteeing deadlines and consequently the processor is always active. On the other hand, tasks in E-EDF are slacked with equal slowdown factor and consequently not all deadlines will be met.

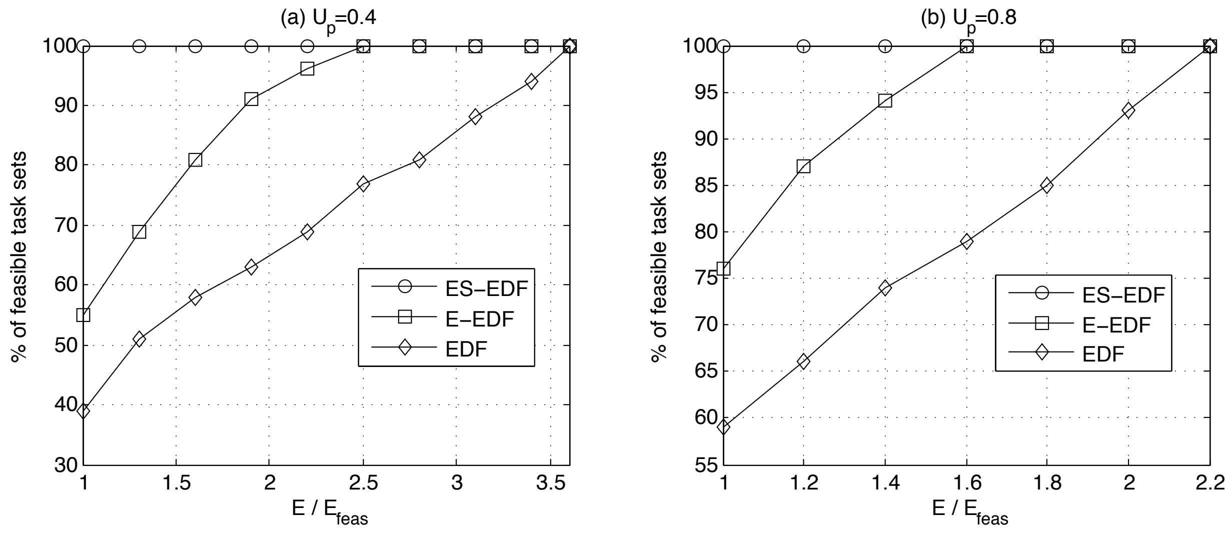


Fig. 5. Percentage of Feasible Task Sets by Varying Battery Capacity

IX. Conclusion

In this paper, we considered the problem of developing an energy saving algorithm for periodic task sets characterized by real-time deadlines using variable voltage and frequency assignments on a monoprocessor system. To this end, we proposed and Energy Saving EDF (ES-EDF) algorithm that is proved to be optimal in minimizing the maximum lateness and the processor energy consumption. In addition, we demonstrated through an illustrative example that it is not necessary that all tasks must be slacked by the same amount so as to obtain a minimum processor energy consumption. We then determined the optimal scaling factor by which a task should be stretched to maximize energy savings while still respecting all deadline constraints. The proposed algorithm achieved an average energy savings of about 28% when compared to EDF. Further, ES-EDF yields higher percentage of feasible task sets which exceeds that of E-EDF and EDF by about 33% and 47%.

We are currently looking at extending the proposed energy saving scheduling algorithm to operate at multiprocessor systems.

Acknowledgements

This work is fully supported by the SNCS Research Center, the University of Tabuk, and the Ministry of Higher Education in Saudi Arabia.

References

[1]

[2] G. Simon, M. Maroti, and A. Ledeczi, *Sensor Network-Based Countersniper System*. In Proceedings of the 2nd ACM Conference on Embedded Network Sensor Systems (SenSys04), Baltimore, Maryland, November, 2004.

[3] C. Hartung, R. Han, C. Seielstad, and S. Holbrook, *FireWxNet: A Multi-Tiered Portable Wireless System for Monitoring Weather Conditions in Wildland Fire Environments*. In Proceedings of the 4th International Confernce on Mobile Systems, Applications, and Services (MobiSys06), Uppsala, Sweden, June, 2006.

[4] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh. *Fidelity and yield in a volcano monitoring sensor network*. In Proceedings of the 7th USENIX Synposium on Operating Systems Design and Implementation (OSDI06), Seattle, Washington, November, 2006.

[5] J. Zhuo and C. Chakrabarti, *Energy-efficient Dynamic Task Scheduling for DVS Systems*. ACM Transactions on Embedded Computing Systems, February, 2008.

[6] W. Horn, *Some Simple Scheduling Algorithms. Naval Research Logistics Quaterly*, 21, 1974.

[7] ML. Dertouzos, *Control robotics: the procedural control of physical processes*. Proceeding of International Federation of Information Processing Congress, 1974.

[8] M. Silly-Chetto, *The EDL Server for scheduling periodic and soft aperiodic tasks with resource constraints*. Real-Time Systems, 17(1), pp.1-25, 1999.

[9] C-L Liu , J-W Layland, *Scheduling algorithms for multiprogramming in a hard real-time environment*. Journal of ACM, 46-61, 1973.

[10] Dertouzos ML, *Control robotics: the procedural control of physical processes*. Proceeding of International Federation of Information Process Cong, 80713, 1974.

[11] J-Y-T Leung , J. Whitehead, *On the complexity of fixed-priority scheduling of periodic real-time tasks*. Performance Evaluation, 1982.

[12] H. Chetto, and M. Chetto, *Some results of the earliest deadline scheduling algorithm*. IEEE Transactions on Software Engineering, 15(10): 1261-1269, 1989.

[13] Yao, F., Demers, A.J., Shenker, S., *A scheduling model for reduced CPU energy*. In: Proceedings of IEEE Symposium on Foundations of Computer Science, 374382, 1995.

[14] T. Ishihara and H. Yasuura, *Voltage scheduling problem for dynamically variable voltage processors*. In Proceedings of the International Symposium on Low Power Electronics and Design, Monterey, CA, pp. 197-202, 1998.

[15] Quan, G., Hu, X., *Minimum energy fixed-priority scheduling for variable voltage processors*. In Proceedings of Design Automation and Test in Europe, 2002.

[16] Kwon, W., Kim, T., *Optimal voltage allocation techniques for dynamically variable voltage processors*. In Proceedings of the Design Automation Conference, 125130, 2003.

[17] Aydin, H.,Melhem, R.,Mossé, D., Alvarez, P.M., *Determining optimal processor speeds for periodic real-time tasks with different power characteristics*. In Proceedings of EuroMicro Conference on Real-Time Systems, 2001.

[18] Jejurikar, R., Gupta, R., *Optimized slowdown in real-time task systems*. In Proceedings of EuroMicro Conference on Real-Time Systems, June, 2004.

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

[20] Zhang, F., Chanson, S.T., *Processor voltage scheduling for real-time tasks with non-preemptible sections*. In Proceedings of IEEE Real-Time Systems Symposium, Dec., 2002.

[21] Kim,W., Kim, J., Min, S.L., *A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis*. In Proceedings of Design Automation and Test in Europe, March, 2002.

[22] Y. Shin and K. Choi, *Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems*. In Proceedings of the 36th Design Automation Conference, DAC99, 1999.

[23] Arezou Mohammadi and Selim G. Akl, *Scheduling algorithms for real-time systems*. Technical report no. 2005-499, 2005.

[24] Y. Shin, K. Choi, and T. Sakurai, *Power optimization of real-time embedded systems on variable speed processors*. In Proceedings of the 2000 IEEE/ACM International Conference on Computer-Aided Design, pages 365-368, 2000.

[25] P. Mejia-Alvarez, E. Levner, and D. Mosse, *Adaptive scheduling server for power-aware real-time tasks*. ACM Transactions on Embedded Computing Systems, 3(2):284-306, 2004.

[26] J.-J. Chen, T.-W. Kuo, and C.-S. Shih, $1 + \epsilon$ *approximation clock rate assignment for periodic real-time tasks on a voltage-scaling processor*. In the 2nd ACM Conference on Embedded Software (EMSOFT), pages 247-250, 2005.

[27] D. Zhu and H. Aydin, *Reliability-aware energy management for periodic real-time tasks*. In Proc. of the IEEE Real-Time and Embedded Technology and Applications Symposium, 2007.

[28] D. Zhu, X. Qi, and H. Aydin, *Priority-monotonic energy management for real-time systems with reliability requirements*. In Proc. of the IEEE International Conference on Computer Design (ICCD), 2007.

[29] Dakai Zhu, Xuan Qi and Hakan Aydin, *Energy Management for Periodic Real-Time Tasks with Variable Assurance Requirements*. In Proceedings of the 2008 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications. Pages 259-268, 2008.

[30] Ravindra Jejurikar and Rajesh Gupta, *Energy-Aware Task Scheduling With Task Synchronization for Embedded Real- Time Systems*. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 25, no. 6, JUNE, 2006.

[31] Y.-S. Chen, C.-Y. Yang, and T.-W. Kuo, *Fl-pcp: Frequency locking for energy-efficient real-time task synchronization*. In the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2007.

[32] Ya-Shu Chen and Tay-Jyi Lin, *Voltage Emergence Prevention for Energy-Efficient Real-Time Task Synchronization*. IEEE 8th International Conference on Computer and Information Technology Workshops, 2008.

[33] Amit Sinha and Anantha P. Chandrakasan, *Energy Efficient Real-Time Scheduling*. Asia-South Pacific Design Automation Conference Proceedings (ASPDAC), 2001.

[34] P. Martineau: Ordonnancement en-ligne dans les systèmes informatiques temps-réel. PhD Dissertation, University of Nantes, 1994.

[35] Intel Corp, *Intel XScale Processor Family Electrical, Mechanical, and Thermal Specification Datasheet*. Santa Clara, CA, USA, 2004.