Energy Aware Dynamic Voltage and Frequency Selection for Real-Time Systems with Energy Harvesting

Shaobo Liu, Qinru Qiu, and Qing Wu
Department of Electrical and Computer Engineering
Binghamton University, State University of New York
Binghamton, New York 13902, USA
{sliu5, qqiu, qwu}@binghamton.edu

Abstract

In this paper, an energy aware dynamic voltage and frequency selection (EA-DVFS) algorithm is proposed. The EA-DVFS algorithm adjusts the processor's behavior depending on the summation of the stored energy and the harvested energy in a future duration. Specifically, if the system has sufficient energy, tasks are executed at full speed; otherwise, the processor slows down task execution to save energy. Simulation results show that when the utilization is low, the EA-DVFS algorithm gives a deadline miss rate that is at least 50% lower than the one given by the lazy scheduling policy. Similarly, when the workload is low, the minimum storage size is reduced by at least 25%.

1. Introduction

Though a lot of research efforts have been made to solve the energy problem of battery-powered embedded electronic system, it continues to be a chief challenge for researchers. This is particularly true for systems where replacing or recharging batteries manually is impracticable. One example is sensor nodes deployed in radioactive surroundings, where the energy constraint has become a main obstacle for increasing its lifespan. In order to solve the energy problem and prolong the system operating duration, a new technology called *energy harvesting*, also known as *energy scavenging*, has recently been explored. Energy harvesting is regarded as an especially prospective approach to decouple the energy constraint of battery-powered systems and increase system autonomy.

Simply put, energy scavenging system is a system that draws parts or all of its operating energy from its ambient energy sources. It has two advantages:

1) It has potential to overcome the energy constraint imposed by traditional battery-powered embedded systems;
2) It may operate perennially until its mechanical failure.

Possible energy harvesting sources include solar, thermal, vibrational and kinetic energy, *etc*. [1]. Two prototypes, namely *Heliomote* [2] and *Prometheus* [3], have been developed to extract solar energy. Both prototypes show that systems may operate perpetually through scavenging solar energy. However, the common drawback of these two prototypes is that they do not target at real-time task sets.

Targeting at real-time task sets, A. Allavena *et.al* proposed an offline algorithm using dynamic voltage and frequency selection (DVFS) [4]. Specifically, this algorithm is designed to schedule a set of independent periodic tasks in a frame. Although it does allow the system to slow down task execution under deadline constraints and therefore reduce power consumption, this approach is based on an unpractical assumption that the harvested energy from the ambient energy source is constant.

The performance of a practical energy-harvesting-capable real-time system, measured by the deadline miss rate, heavily depends upon the stored energy and the energy harvested from the environment. Unfortunately, the scavenging power is time-varying and thus very unstable. Therefore, the accurate modeling for energy source plays a key role in designing a good policy to schedule the task and reduce the deadline miss rate.

The work in [5] moves toward the direction of accurately modeling energy source. The authors assume that the energy source is scavenged from solar cells, which work in two modes: day and night; and correspondingly output two types of power. But the modeling is still coarse-grained because the output power is changing due to the varying sunlight strength in daytime.

A better model is proposed by A. Kansal et al. [6]. Their model captures the behavior of an actual solar energy source through tracing its power profile. In that paper, the authors proposed algorithms for tuning system duty cycle based on the parameters of the solar energy source. The system switches between active mode and sleep mode depending on harvesting energy. As such, it may operate perpetually. However, the proposed algorithms do not target at concrete tasks in a real-time pattern. It therefore remains unclear how to deal with the real-time tasks under the strong variation of energy source with respect to time.

Based on the work in [6], the authors in [7] proposed a real-time scheduling algorithm called lazy scheduling algorithm (LSA). According to LSA, the processor executes all tasks at full power; and the system starts executing a task if the following three conditions are met simultaneously:

1) The task is ready;
2) The task has the earliest deadline among all ready tasks
3) The system is able to keep on running at the maximum power until the deadline of the task.

Since a task is executed at the maximum power, it may be finished well before its deadline. In this case, the task slack would be wasted; more importantly, the limited energy is unnecessarily squandered. As a consequence, future tasks have to violate their deadlines because of energy shortage.

In this paper, we propose an energy aware dynamic voltage and frequency selection (EA-DVFS) algorithm. The purpose of our algorithm is to efficiently use the task slack and further reduce the deadline miss rate. In our algorithm, whether or not the system slows down the task execution for energy saving depends on the available energy. If the system has sufficient energy, the task is executed at its full speed; otherwise, it is stretched and executed at a lower speed. Compared to LSA, the proposed EA-DVFS algorithm significantly reduces the deadline miss rate and the storage size when the utilization ratio is not high. We will show this point later in section 5.

The remainder of this paper is organized as follows. In Section 2, we illustrate a motivational example. The energy harvesting system model and some assumptions are presented in section 3. In section 4, the proposed EA-DVFS algorithm is described in details. Simulation results and discussions are presented in Section 5. Finally section 6 concludes the paper.

2. A Motivational Example

Consider the following application: there are two tasks τ_1 and τ_2, shown in Figure 1. Task τ_1 and task τ_2 are represented by triples (a_1, d_1, w_1) = (0, 16, 4) and (a_2, d_2, w_2) = (5, 16, 1.5), where symbols a_i, d_i, w_i (i = 1,2) indicate the arrival time, the relative deadline and the worst case execution time of task τ_1 and task τ_2, respectively.

Assuming at time instance 0, the stored energy in the energy storage is 24, the system maximum power consumption is 8, and the harvested power from time 0 to 25 is set to 0.5. Now we know that the harvested energy from 0 to 16 is 8, the total energy available for the system up to time instance 16 is 32, and the running time is 4 if the

978-3-9810801-3-1/DATE08 © 2008 EDAA

system operates at the maximum power.

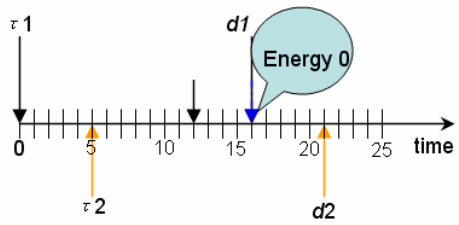

Figure 1: A motivational example

Based on lazy scheduling policy, the system starts running task τ_1 at time 12, shown in a short arrow in Figure 1, and finishes it at time 16. The system depletes all energy exactly at time 16, shown in Figure 1. As to task τ_2, the system needs 12 units of energy to finish it at the maximum power before its deadline time instance 21. However, from time instance 16 to 21, the energy that the system harvests is only 2.5(5*0.5), which is 9.5 shy of 12. In the end, the deadline of task τ_2 is violated because of the energy shortage.

On the other hand, we assume that the processor operates in two speeds: a high speed and a low speed, with the former twice as fast as the latter. The power at high speed is 3 times as much as that in low speed. We run task τ_1 at the low speed, the remaining energy at 16 is 32/3, which is calculated from the expression 24+8-4/(1/2)*8/3. At time 21, the available energy is 32/3+5*0.5 = 13.16. This time the system has enough available energy to finish task τ_2 by its deadline even if the task is executed at the maximum power.

The example suggests that we should slow down the task execution if possible to reduce the deadline miss rate. Before moving to describe the proposed energy aware DVFS algorithm, we would like to introduce the system model and some assumptions.

3. Energy Harvesting System Model and Assumptions

This paper deals with a real-time system with the energy harvesting ability, shown in Figure 2. At some time t, the energy source module harvests the energy from its ambient environment and then converts it into electrical energy at power $Ps(t)$. The electrical energy can be stored in the energy storage, whose capacity is C. The stored energy is donated as $E_C(t)$ at time t. Apparently, at any time, the stored energy is no more than the storage capacity, that is

$$E_C(t) \le C \quad \forall t \tag{1}$$

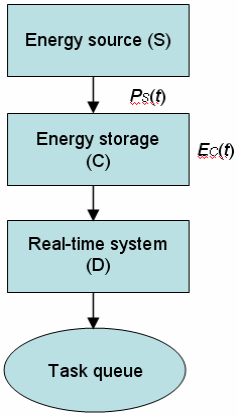

Figure 2: A real-time system with energy harvesting module

On the other hand, the real-time system draws the energy and uses it to process tasks with arrival time, deadline, and worst case execution time. The worst case execution time represents the maximum energy demand of the task. We assume that the real-time system is a uniprocessor system, and only one task is executed at any time t. The preemption is allowed in this system. The high priority task has privilege to preempt the low priority task for the execution. The earlier the task deadline, the higher priority the task has. Therefore, the task with the earliest deadline always has the highest priority among all ready tasks, and it is always executed earliest by the system. The more details about the real-time system are introduced in subsection 3.3 real-time system model.

3.1. Energy source

Most of the environmental energy sources are strongly varying with time. Hence, in many cases we need auxiliary circuitry [8] to stabilize the output power of the energy source so that the system works properly. In our energy source model, the output power $P_S(t)$ has excluded the loss incurred by the auxiliary circuitry. In another word, $P_S(t)$ is the net power to feed the storage. The harvested energy $E_S(t_1, t_2)$ by $P_S(t)$ at time interval $[t_1, t_2]$ is given as

$$E_S(t_1,t_2) = \int_{t_1}^{t_2} P_S(t)dt \tag{2}$$

The energy source is a time varying variable. We can not determine $P_S(t)$ before time t. What we need to do is to predict $P_S(t)$ by tracing its profile[9].

3.2. Energy storage

In this paper, we assume the energy storage is ideal. It can be recharged up to its capacity. Likewise, it can also be completely discharged to as less as zero. If the stored energy reaches the capacity, the incoming harvested energy overflows the storage and is discarded. When the processor runs the task, the system draws the energy from the storage. Let $E_D(t_1, t_2)$ indicate the energy the system draws from time t_1 to t_2, we have the following relations:

$$E_D(t_1,t_2) \le E_C(t_1) + E_S(t_1,t_2) \qquad \forall t_1 < t_2 \tag{3}$$

$$E_C(t_2) \le E_C(t_1) + E_S(t_1,t_2) - E_D(t_1,t_2) \qquad \forall t_1 < t_2 \tag{4}$$

Since the energy needed for the system solely comes from the stored and harvested energy, the system cannot consume more than the summation of these two at any time. So the constraint imposed by inequality (3) must hold. The harvested energy $E_S(t_1, t_2)$ contributes to the stored energy; on the contrary, the processor draws energy $E_D(t_1, t_2)$ and reduces the reservoir of the storage by $E_D(t_1, t_2)$. Therefore, the stored energy $E_C(t_2)$ at time t_2 can not be more than $E_C(t_1)+E_S(t_1, t_2)-E_D(t_1, t_2)$, which is guaranteed by inequality (4).

3.3. Real-time system

We assume all tasks are independent and preemptive. Task τ_m is represented by a triple (a_m, d_m, w_m), where a_m, d_m, w_m indicate the arrival time, the relative deadline and the worst case execution time of task τ_m under the maximum frequency, respectively. The arrival time of the task is not known beforehand. We do not have the knowledge of the deadline as well as the worst case execution time of the task before it is released, either. However, as long as the task is released, all these parameters are determined.

We consider a real-time system equipped with a DVFS-enabled processor. The processor has N discrete clock speeds: $f_{min} = f_1 < f_2 < \ldots < f_N = f_{max}$; each clock speed f_n corresponds to a power consumption P_n, where $1 \le n \le N$ and $P_{max}=P_N$. We define a slowdown factor S_n associated with each clock speed f_n, where S_n is the normalized frequency of f_n with respect to the maximum frequency f_{max}. Obviously, S_n is equal to f_n/f_{max}. If the clock speed of the processor decreases to f_n, its power consumption reduces to P_n correspondingly.

If the execution of task τ_m is slowed down by S_n, then its actual

execution time at frequency f_n increases to w_m/S_n from w_m. All tasks are scheduled according to earliest deadline first (EDF) policy. The task with the earliest deadline has the highest priority, and it will preempt any other task if the processor is busy, letting the processor execute itself first.

4 Energy-aware Dynamic Voltage and Frequency Selection Algorithm

DVFS has been proven to be a powerful way to reduce the system power consumption [12-15]. We are going to introduce the energy aware DVFS algorithm in this section.

4.1 Execute tasks at full speed

The energy storage usually consists of a rechargeable battery or a super capacitor or both, so the capacity of the energy storage could not be unlimited. When the energy storage is full, the incoming harvested energy can not be stored any more and has to be discarded. In that situation, there is no need to slow down task execution for energy saving. More generally, system can run tasks at its maximum power as long as there is enough remaining energy in the storage. The reason is that the harvested energy will automatically replenish the storage when the system is idle afterwards. In some cases, the harvested energy even overflows the storage, which completely counteracts the previous effort to save energy by slowing down task execution.

4.2 Slow down the task execution

When the available energy is zero, the system has to stop running any tasks. If it happens some task, say τ_m, needs to be executed while the available energy is zero, the system will delay task τ_m execution until it has scavenged energy. So the deadline of task τ_m may be violated. However, if the system slows down the execution of tasks prior to task τ_m when the system nearly depletes the available energy, the system may still have enough energy to finish task τ_m before its deadline, which has been illustrated in Figure 1.

Therefore, we need develop an algorithm to automatically let the system know when it runs the task at the full speed, and when at a reduced speed by DVFS. Before making the decision, the system should know the available energy, and that is measured indirectly in our algorithm.

Assume task τ_m arrives at time instance a_m, its worst case execution time is w_m and its relative deadline is d_m; the stored energy in the energy storage is $E_C(a_m)$; the harvested energy from a_m to $a_m + d_m$ is $E_S(a_m, a_m + d_m)$, we can calculate the system running time sr_n such that all available energy is completed depleted at power P_n until time instance $a_m + d_m$.

$$sr_n = \frac{E_C(a_m) + E_S(a_m, a_m + d_m)}{P_n}. \quad (5)$$

It will take the system w_m/S_n to finish task τ_m at frequency f_n. When choosing frequency f_n to slow down the system, we must make sure the actual execution time of task τ_m is no more than the time interval between its arrival time and deadline; otherwise, there is no way to respect the deadline of task τ_m. Therefore, the following inequality holds,

$$\frac{w_m}{S_n} \le d_m - a_m \quad (6)$$

The slack of the task can be traded for energy savings. The slack is defined as the maximum amount of time that a task can use for slowing the processor down without violating timing constraints. The more slack of the task, the greater potential for energy saving. When the remaining energy is small, the task slack should be exploited sufficiently to save as much energy as possible. So we need to find the minimum frequency f_n ($1 \le n \le N$) under the constraint of inequality (6). Then the starting running time of task τ_m at frequency f_n can be determined

$$s_1 = \max(a_m, a_m + d_m - sr_n) \quad (7)$$

Specifically, if task τ_m is executed at full speed, the system should start running it at time

$$s_2 = \max(a_m, a_m + d_m - sr_{\max}) \quad (8)$$

where $sr_{\max}$ is calculated by the following equation

$$sr_{\max} = \frac{E_C(a_m) + E_S(a_m, a_m + d_m)}{P_{\max}} \quad (9)$$

If s_1 is equal to s_2, it means they are both equal to a_m. So we have

$$a_m + d_m - sr_{\max} \le a_m, \quad (10)$$

and

$$a_m + d_m - sr_n \le a_m \quad (11)$$

That is

$$sr_n \ge sr_{\max} \ge d_m \quad (12)$$

The constraint (12) means that the system running time at maximum power is larger than d_m. In another words, the system still has enough available energy $E_C(a_m) + E_S(a_m, a_m + d_m)$ between task arrival time and its deadline, so the task is executed in the maximum power $P_{\max}$.

If s_1 is not equal to s_2, their inequality is interpreted as that the available energy $E_C(a_m) + E_S(a_m, a_m + d_m)$ between time instance a_m and $a_m + d_m$ is not sufficient, so task τ_m is executed at a reduced clock speed f_n for energy savings.

4.3 Prevent from stealing excessive time from future tasks

Meanwhile, the task can not be stretched greedily for energy saving; otherwise, the current task steals excessive time from the future task and so that the future task has no way to respect its deadline, although the available energy is sufficient. Assume there are two tasks τ_1 and τ_2, shown in Figure 3, where task τ_1 and task τ_2 are represented by tripe $(a_1, d_1, w_1) = (0, 16, 4)$ and $(a_2, d_2, w_2) = (5, 12, 1.5)$ respectively. If task τ_1 is stretched excessively, then under no circumstance is the system able to finish task τ_2 before its deadline. We will explain it in details in the following paragraphs.

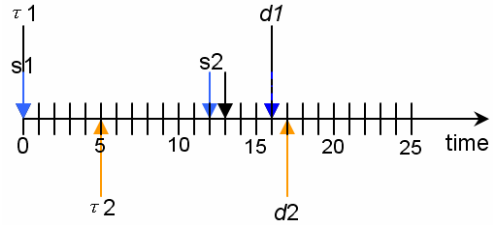

Figure 3: Preventing from excessively stretching task

Assuming at time instance 0, the predicted available energy $E_C(0) + E_S(0, 16)$ between time instance 0 and 16 is 32, and the system maximum power is 8. In terms of equation (5), the system running time at maximum power is 32/8 = 4. If task τ_1 is executed at the full speed, it should be run at time instance $s_2 = \max(0, 16\text{-}4) = 12$, as shown in Figure 3.

On the other hand, we assume the minimum frequency for task τ_1 under the constraint of inequality (6) is $f_n = 0.25 f_{\max}$, and the corresponding power consumption is $P_n = 1$. Hence, the system running time at frequency f_n is 32/1=32. If task τ_1 is executed at frequency f_n, the system starts running it at $s_1 = \max(0, 16 - 32) = 0$.
If the system executes task at frequency f_n until task τ_1 is finished at

time instance 0+4/0.25=16, then the system has no way to finish task τ_2 before its deadline. Even if the system runs task τ_2 at the maximum power, task τ_2 can not be finished until time instance 16+1.5=17.5, which is later than the deadline of task τ_2 17. However, if task τ_1 is executed at frequency f_n between s_1 and s_2, and it is executed at the maximum frequency after s_2, then the system finishes task τ_1 at time instance 12+(4-(12-0)*0.25)= 13, which is labeled by a short black arrow, shown in Figure 2. The energy consumption for task τ_1 is 12+8=19.

Before the deadline of task τ_2, the available energy for the system is (32-19)+ E_S(16, 17.5), which is no less than 13. If task τ_2 is executed at the full speed, the system will consume energy 8*1.5=12, which is less than the available energy. If task τ_2 is executed at the maximum power at time instance 14.5, then the system finishes running it at time 16, which is earlier than the deadline of task τ_2. Therefore, through carefully stretching tasks, we can guarantee the deadlines of both tasks.

In order to reduce the deadline miss rate of the system, we adopt the following policy to slow down tasks.

1. find the task with the earliest deadline among ready tasks.
2. compute the earliest starting execution time s_1 of the task if the system executes it at power p_n(frequency f_n) under the constraint of inequality (6)
3. compute the earliest starting execution time s_2 of the task if the system executes it at the maximum power
4. compare s_1 against s_2;
   a) if s_1 and s_2 are equal to each other, we assume that there is sufficient available energy in the system, and the task is executed at the full speed;
   b) if s_1 is not equal to s_2, the available energy is considered to be nearly depleted; the task is executed at frequency f_n between s_1 and s_2; the task is executed at the full speed after s_2 if the task has not been finished.

```
1.  Require: maintain a task queue Q containing all ready but
    not finished tasks τ_m.
2.  while (true) do
3.      d_j ⇐ min{d_m : τ_m ∈ Q}
4.      calculate s_1 and s_2 for task τ_j
5.      if s_1 = s_2, then
6.          execute task τ_j at the maximum power/the maximum
            frequency
7.      end if
8.      if s_1 < s_2, then
9.          execute task τ_j at power P_n/frequency f_n.
10.         if task τ_j has not been finished at time s_2, then
11.             execute task at the maximum power afterwards,
12.         end if
13.     end if
14.     t ⇐ current time
15.     if t = a_k, then
16.         task τ_k is put into the ready task queue Q
17.     end if
18.     if task τ_j is finished, then
19.         remove task τ_j from the ready task queue Q
20.     end if
21. end while
```

Figure 4: Energy aware DVFS algorithm

The proposed energy aware DVFS algorithm is shown in Figure4. The proposed algorithm schedule tasks in a preemptive way. The task with earliest deadline has the highest priority. Once the scheduler selects the task with the earliest deadline, the calculations of s_1 as well as s_2 has to be performed; then the scheduler determines at which frequency/power to run the task depending upon comparison of s_1 and s_2. If s_1 is not equal to s_2, then the system deems that the available energy is quite small. Therefore, system slows down the task execution between s_1 and s_2 for energy saving. After time instance s_2, if the task is not finished, it is executed at full speed to avoid its stealing excessive time from the future task. Otherwise, the system delays the future task too much, resulting in that the future task does not have enough time to finish before its deadline, although the system has sufficient energy, as illustrated in Figure 3.

Meanwhile, parameters s_1 and s_2 actually decouple the energy constraint. As long as the task is executed no earlier than s_1, the system is guaranteed to have enough available energy to keep on running until its deadline at a proper frequency. Parameter s_2 plays the same role as s_1.

Consider a special case: if the energy storage is infinite, what happens to the proposed energy-aware DVFS?

Assume the scheduler selects a task with the earliest deadline, the next step is to calculate s_1 and s_2. Before that, we need to know sr_n and sr_{max}. Since the capacity of the energy storage is infinite, sr_n and sr_{max} are both equal to infinite. In terms of equation (7) and equation (8), we have that s_1 and s_2 are both equal to the arrival time of the task. It means that as long as the task is ready, it is executed at the maximum power by the system. In effect, in that case the proposed energy aware DVFS is equivalent to that the system schedules the task following earliest deadline first (EDF) policy. That is a justifiable behavior. Because the available energy is infinite, the system no longer needs make any effort for energy saving.

In summary, when the energy storage capacity is infinite, the proposed energy aware DVFS algorithm is reduced to EDF. When the capacity is finite, the proposed algorithm changes its behavior based upon the available energy. If there is sufficient energy available for the system, our algorithm is tantamount to the algorithm proposed in [7, 10]. Otherwise, the task is stretched and the system slows down the task execution by dynamic voltage and frequency selection (DVFS). Though this way, the system trades the task slack for energy savings and reduces the deadline miss rate.

5. Simulations and Discussions

To evaluate the effectiveness of the proposed algorithm on energy saving and performance improvement, we develop a discrete-event simulation in C/C++. In the simulator, we implement the proposed energy-aware DVFS algorithm. For the sake of comparison, we also implement the lazy scheduling algorithm (LSA) proposed in [7, 10].

5.1 Simulation setup

Since the solar power not only exhibits the stochastic behavior, but also the periodic and deterministic aspect, we use the following random number generator [11] to model the solar energy source behavior,

$$P_S(t) = \left| 10 \cdot N(t) \cdot \cos(\frac{t}{70\pi}) \cdot \cos(\frac{t}{70\pi}) \right| \quad (13)$$

where $N(t)$ is subject to the normal distribution with mean 0 and variance 1. Its behavior is shown in Figure 5. In the realistic applications, we can not have the knowledge of the future harvested energy E_S. Therefore, in the simulation, we trace $P_S(t)$ profile to predict the harvested energy from a future period.

We consider a DVFS-enabled processor with five speeds similar to Intel's Xscale processor [16]: 150MHz, 400MHz, 600MHz, 800MHz and 1000MHz. And power consumptions are 80mW, 400mW,

1000mW, 2000mW and 3200mW respectively. The overhead from voltage switching is assumed to be negligible.

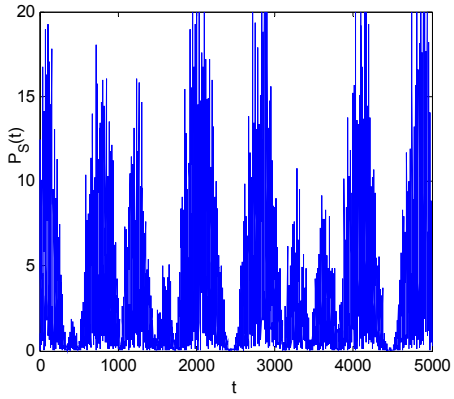

Figure 5: Energy source behavior

The number of periodic tasks in a task set is arbitrary, and the task period is chosen from a set {10, 20, 30, …, 100}, where each value has the same opportunity of being selected. For the sake of simplicity, the relative deadline of the periodic task is set to its period. The worst case execution time of the task is obtained in the following way: assuming the average harvested power is $\overline{P}_S$, and the task period is p. The energy consumption e for the task under the worst case is generated in terms of the uniform distribution [0, $\overline{P}_S \cdot p$]. Then its worst case execution time is equal to $e/P_{\max}$.

The utilization U of a processor is defined as

$$U = \sum_m \frac{w_m}{p_m} \tag{14}$$

where w_m is the worst case execution time of task τ_m, and p_m is the period of task τ_m. The utilization U cannot be larger than 1. Otherwise, the processor is not able to schedule tasks while respecting their deadlines. In order to get the specific utilization, we scale the worst case execution time of each task in a task set in the same ratio.

In the beginning of the simulation, the energy storage is full. The simulation terminates after 10,000 time units and is repeated for 5,000 task sets for a specific utilization.

5.2 Improvement in remaining energy

First we take interest in the remaining energy in the system. For the fair comparison of LSA and EA-DVFS, all simulations are performed under the same condition. The utilization has significant impact on the stored energy. In this subsection, we set U to 0.4 and 0.8 respectively.

As results from all simulation scenarios show similar trends of remaining energy if U is set to the same value, we only exhibit the results with 5 periodic tasks in a task set.

When the storage capacity is varying, the absolute remaining energy varies accordingly. In order to eliminate the effect of the storage capacity, the remaining energy is normalized. In this set of simulations, the capacity is set to 200, 300, 500, 1000, 2000, 3000 and 5000. Then the weighted average of normalized remaining energy for each capacity is calculated to get the final remaining energy, each normalized remaining energy having the same weight.

When U is set to 0.4, the remaining energy curves in the LSA-based system and the EA-DVFS-based system are both plotted in Figure 6. As shown in Figure 6, the EA-DVFS-based system stores significantly more energy than the LSA-based system on average. That is because EA-DVFS algorithm slows down the task execution for energy savings; however, LSA-based system always executes the task at the full speed and it consume more energy to finish an identical task.

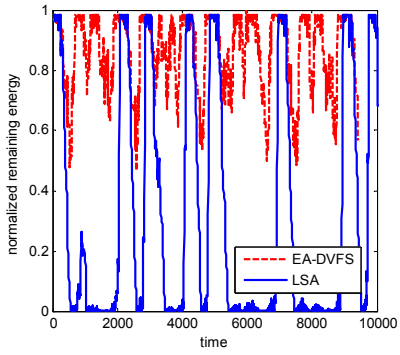

Figure 6: Remaining energy with low utilization

When U is set to 0.8, we get another plot shown in Figure 7. By observing the stored energy, we find that EA-DVFS-based system only has slightly more stored energy than the LSA-based system. The reason is twofold: On one hand, when the utilization is high, the processor rarely has chance to slow down the task execution for energy saving; most of time both algorithms execute the task at the full speed. On the other hand, the system also reduces the chance to be completely idle for energy harvesting from the environmental energy source, which results in that the consumed energy can not be supplemented in time.

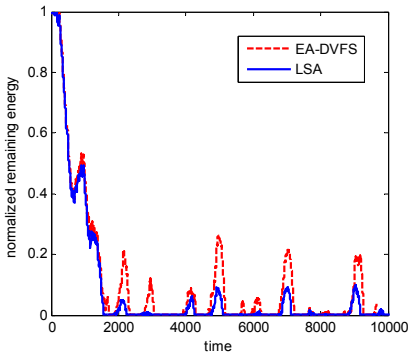

Figure 7: Remaining energy with high utilization

5.3 Reduction in deadline miss rate

As observed in Figure 6, the EA-DVFS based system has significantly more stored energy that LSA based system when utilization is low. The more stored energy means that more tasks are able to be finished before their deadlines. Hence, the EA-DVFS system significantly reduces the deadline miss rate due to energy shortage, compared to LSA-based system, as shown in Figure 8, where U is set to 0.4.

Through observing the results shown in Figure 8, we know that when both systems have the same storage capacity, EA-DVFS algorithm reduces the deadline miss rate over 50% on average, compared to LSA algorithm.

If we increase U to 0.8 and run the simulation again, we find EA-DVFS algorithms performs as well as LSA algorithm does, shown in Figure 9. That is because the EA-DVFS based system seldom has chance to slow down the execution for energy savings when utilization is high, and most of tasks are just executed at the full speed, as LSA algorithm does.

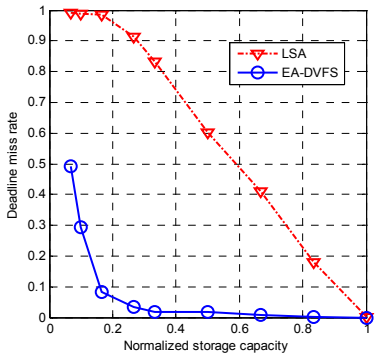

Figure 8: Deadline miss rate with low workload (U=0.4)

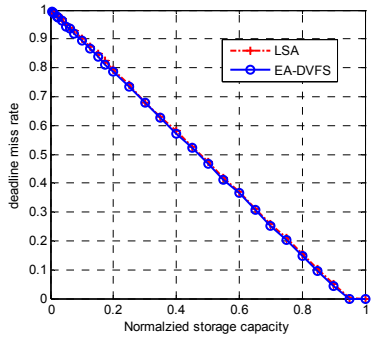

Figure 9: Deadline miss rate with high workload (U=0.8)

5.4 Decrease in storage capacity

We also run another set of simulations to get the minimum storage capacity which is the threshold capacity to maintain zero deadline miss rate. Symbols $C_{\text{min,EA-DVFS}}$ and $C_{\text{min-LSA}}$ are used to represent the minimum storage from EA-DVFS algorithm and LSA algorithm respectively. The utilization U is swept from 0.2 to 0.8 with a step 0.2. The simulations results are shown in table 1.

Table 1: The ratio of the minimum storage capacities

U	0.2	0.4	0.6	0.8
$C_{\text{min-LSA}}/C_{\text{min,EA-DVFS}}$	2.5	1.33	1.05	1.01

When utilization is as low as 0.2, $C_{\text{min-LSA}}$ is 2.5 times as large as $C_{\text{min,EA-DVFS}}$ to maintain the zero deadline miss rate. However, the difference reduces with the utilization increasing. When utilization is 0.8, $C_{\text{min-LSA}}$ is only 1% larger than $C_{\text{min,EA-DVFS}}$. The reason is just as before: when the utilization is low, the EA-DVFS-based system is able to reduce the task execution for energy savings. The saved energy is used for running future tasks, so the system just needs a smaller capacity to keep the zero deadline miss rate, compared to the LAS based system. While the workload increases, both algorithms execute the task in the similar way, accordingly they nearly have the same requirement on the energy storage to keep zero deadline miss rate.

6. Conclusions

We have proposed an energy aware dynamic voltage and frequency selection algorithm in this paper. The proposed algorithm runs tasks at the full speed if the stored energy is sufficient; otherwise, the system slows down the task execution for energy savings. The efficiency of EA-DVFS algorithm to save energy depends on the processor utilization. When the utilization is low, the processor has great potential to save energy by slowing down the task execution. Accordingly, when utilization is 0.4, the proposed EA-DVFS algorithm reduces deadline miss rate by at least 50%, compared to LSA algorithm. Similarly, the proposed EA-DVFS algorithm also decrease the minimum storage size by at least 25% to keep the zero deadline miss rate with low workload.

However, when utilization is high, the processor seldom has chance to trade the task slack for energy saving, the proposed EA-DVFS algorithm just performs as well as LSA algorithm does.

Finally we points out that the proposed EA-DVFS algorithm is very effective in reducing deadline miss rate and storage size for a real-time system with energy-harvesting ability when the workload is not high.

References

[1] S. Roundy, D. Steingart, L. Frechette, P.K. Wright, and J.m. Rabaey. Power sources for wireless sensor networks. In Wireless Sensor Networks, First Europeean Workshop, EWSN 2004, Proceedings, Lecture Notes in Computer Science, Pages 1-17, Berlin, Germany, January 19-21 2004.

[2] V. Raghunathan, A. Kansal, et al, "Design considerations for solar energy harvesting wireless embedded systems", In Proceedings of the Fouth International Symposium on Information Processing in Sensor Networks, IPSN 2005, pp457-462, UCLA, Los Angeles, California, USA, April 25-25, 2005

[3] X. Jiang, J. Polastre, and D. E. Culler, "Perpetual environmentally powered sensor networks", In Proceedings of the Fourth International symposium on Information Processing in Sensor Networks, IPSN 2005, pp463-468, UCLA, Los Angeles, California, USA, April 25-25, 2005

[4]. A. Allavena and D. Mosse, "Scheduling of frame-based embedded systems with rechargeable batteries," In Workshop on Power Management for Real-time and Embedded Systems, 2001

[5] C. Rusu, R. G. Melhen, and D. Mosse, "Multi-version scheduling in rechargeable energy-aware real-time systems", In 15th Euromicro Conference on Real-time systems, ECRTS 2003, pp95-104, Berlin, Germany, Jan., 2004.

[6] A. Kansal, J. Hsu, "Harvesting aware power management for sensor networks", In IEEE DAC 2006.

[7] C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Lazy scheduling for energy-harvesting sensor ndoes," in Fifth Working Conference on Distributed and Parallel Embedded Systems, DIPES 2006, pp125-134, Braga, Portugal, October, 2006

[8] Y. Lam, S. Koon, W.Ki and C. Tsui, " Integrated direct output current control switching converter using symmetrically-matched self-biased current sensors," ASP-DAC 2006: 102-103

[9] A. Kansal, J. Hsu, S. Zahedi and M. Srivastava, "Power Management in Energy Harvesting Sensor Networks," In *ACM Transactions on Embedded Computing Systems (in revision)* , 35 pages , May 2006.

[10]C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-time scheduling with regenerative energy," in Proc. of the 18th Euromicro Conference on Real-time Systems (ECRTS06), pp261-270, Dresden, Germany, October, 2006.

[11]C. Moser, D. Brunelli, L. Thiele, and L. Benini, "Real-Time Scheduling for Energy Harvesting Sensor Nodes,"MICS Scientific Conference and SNF Panel Review, Zurich, Switzerland October, 2006.

[12]F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in IEEE symposium on Foundations of Comp. Science, Pages 374-382, 1995.

[13]I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. srivastava. "Power Optimization of Variable-Voltage Core-Based systems," IEEE Trans. On Computer-Aided Design, 18(12):1702-1714, Dec 1999.

[14]J. Luo and N. K. Jha, "Static and dynamic variable voltage scheduling algorithms for real-time heterogeneous distributed embedded systems," In Proc. Of Int. Conf. on VLSI Design, pp.719-726, 2002

[15]N.K.Jha, "Low-power system scheduling, synthesis and displays," In Proc. IEE, 2005

[16] Intel-Xscale Micro-architecture, available at http://www.intel.com