Minimizing Expected Energy in Real-Time Embedded Systems [*]

Ruibin Xu, Daniel Mossé, Rami Melhem
Computer Science Department, University of Pittsburgh
Pittsburgh, PA 15260
{*xruibin,mosse,melhem*}*@cs.pitt.edu*

ABSTRACT

We study the problem of minimizing energy consumption in real-time embedded systems that execute variable workloads and are equipped with processors having dynamic voltage scaling (DVS) capabilities. This problem is about how to decide tasks' running speeds (speed schedule) before they are scheduled to execute. In this paper, we show that it is possible to incorporate the dynamic behavior of the tasks into the speed schedule to, along with the dynamic slack reclamation technique, minimize the expected (total) energy consumption in the system.

Categories and Subject Descriptors: D.4.1 [*Operating Systems*]: Process Management - Scheduling; D.4.7[*Operating Systems*]: Organization and Design - Real-time systems and embedded systems

General Terms: Algorithms

Keywords: Real-time, Dynamic Voltage Scaling, Power management, Processor Acceleration to Conserve Energy

1. INTRODUCTION

Energy conservation is critically important for many real-time systems such as battery-operated embedded systems which have a restricted energy budget. Dynamic voltage scaling (DVS), which involves dynamically adjusting the voltage and frequency of the CPU, has become a well-known technique in power management for real-time embedded systems. Through DVS, quadratic energy savings can be achieved at the expense of just linear performance loss [13, 16]. Thus, the execution of tasks can be slowed down in order to save energy, as long as the deadline constraints are not violated. A natural problem that rises from this context is how to minimize the energy consumption in the system while still meeting the deadlines. The problem is about determining a task's speed (or equivalently, determining the amount of time allotted to a task) before it is scheduled to execute in the system.

The systems under our consideration are frame-based hard real-time embedded systems that execute variable workloads. In these systems, tasks usually run for less than their worst-case execution time (WCET), creating the opportunity for dynamic slack reclamation to further slow down the future tasks. Furthermore, the execution time of tasks is usually unpredictable before their execution. Therefore, the design goal of DVS schemes becomes *minimizing the expected (total) energy consumption* in the system.

In this paper, we show that failing to capture the dynamic behavior of the tasks by the existing DVS schemes leads to suboptimal power management, and that it is possible to incorporate the dynamic behavior of the tasks into the speed schedule to, along with the dynamic slack reclamation technique, minimize the expected (total) energy consumption in the system. The dynamic behavior of the tasks is captured by the probability density function of the workload of the tasks, which, in practice, is represented by a histogram. Profiling on training data sets, or online learning for a certain number of frames, can be used to obtain the probability density function of the workload of a task. Provided with the probability density functions for all the tasks, our DVS scheme can derive the optimal speed schedule to minimize the expected energy consumption in the system. Our DVS scheme is divided into two phases: (a) the offline phase precomputes the speed schedule, which consists of the percentage of the time left before the deadline allotted to each task; (b) the online phase is invoked before the execution of each task, obtaining the time left before the deadline and computing the execution speed for the task. Both phases are efficient: the offline phase runs in polynomial time and the online phase only takes constant time.

This paper is organized as follows. We first briefly review the closely related work in Section 2. The system and task model are described in Section 3. Section 4 presents our optimal DVS scheme and proof of correctness. In Section 5, we discuss issues when applying our optimal DVS scheme in practice. We end the paper in Section 6 with concluding remarks and future work directions.

2. CLOSELY RELATED WORK

Real-time applications usually exhibit a large variation in actual execution times [3, 12]. Thus, DVS schemes must take into consideration unused computation time of tasks.

DVS in real-time applications is categorized as *inter-task* or *intra-task* voltage scaling [7]. Inter-task schedules speed changes at each task boundary, while intra-task schedules speed changes within a single task. For inter-task voltage scaling, Mossé et al. [11] introduced the concept of *speculative speed reduction* and proposed three DVS schemes (i.e., greedy, proportional, and statistical schemes) with different speed reduction. To be able to navigate the full spectrum of speculative speed reduction, in [2] system designers can set a parameter to control the degree of speed reduction aggressiveness. Our DVS scheme chooses the degree of speed reduction aggressiveness automatically, based the probability distribution of

[*]This work has been supported by NSF grant ANI-0125704 and ANI-0325353.

the workload of the tasks, to minimize the expected energy consumption.

For intra-task voltage scaling, Lorch et al. [9] have shown that if a task's computational requirement is only known probabilistically, there is no constant optimal speed for the task and the expected energy consumption is minimized by gradually increasing speed as the task progresses, which is an approach named as *Processor Acceleration to Conserve Energy* (PACE). Practical PACE (PPACE) [15] takes into consideration a number of practical issues and improves the performance of PACE. However, PACE and PPACE have only been studied for single task when considering hard real-time guarantee. In [10], PACE is used for soft real-time systems when the system has only one task but the maximum speed is used when the system has multiple tasks. In Section 4.1, we present the theoretical results of using PACE for multiple tasks with a single hard deadline (frame length).

AbouGhazaleh et al. [1] proposed a hybrid compiler-operating system intra-task DVS scheme for energy consumption of time-sensitive embedded applications. Our scheme is implemented at the operating system level and assumes no access to application source codes.

3. TASK AND SYSTEM MODEL

We consider a frame-based task model with N periodic tasks in the system, all ready at time zero. The task set is denoted by $T = \{T_1, T_2, \ldots, T_N\}$. Each task T_i ($1 \leq i \leq N$) is characterized by its worst-case execution cycles (WCEC) W_i and the probability density function of its execution cycles $P_i(x)$, which denotes the probability that task T_i executes for x ($1 \leq x \leq W_i$) cycles. Obviously, we have $\sum_{x=1}^{W_i} P_i(x) = 1$ and $P_i(W_i) \neq 0$. All task periods are identical and all task deadlines are equal to their period. The common deadline/period (also known as frame length) is denoted by D. The execution of the frame is to be repeated and all tasks must be executed during each frame. There are two possible relationships among the tasks: (1) they are all independent, which means that the execution order is flexible; (2) they must execute consecutively in a specific order, where the tasks can be treated as sequential sections of a single application.

The tasks are to be executed on a variable voltage processor with the ability to dynamically adjust its frequency and voltage on application requests. We assume that the processor is the major power consumer, which is true for many embedded systems. We also assume that the processor frequency can be adjusted continuously from 0 to infinity (we shall discuss the more realistic cases, such as the processor has minimum and maximum frequencies, in Section 5). The processor power consumption when running at frequency f is $c_0 + c_1 f^{\alpha}$ (α is a constant that is at least 2) where c_0 and c_1 denote the power consumption of the processor when idle and the maximum dynamic power, respectively. The dynamic power is determined by the processor operating frequency and the maximum dynamic power is the dynamic power consumed when the processor is operating at the maximum frequency. We assume that the system is never turned off, therefore we can ignore c_0 and c_1 because they do not affect the analysis results. Thus, the processor power consumption used in the analysis is $p(f) = f^{\alpha}$. We assume that the power consumption for all other components in the system is constant, and thus can be ignored without affecting the analysis results.

4. THE OPTIMAL DVS SCHEME

In this section, we will give the details of our new DVS scheme, which is based on an important property of optimal expected energy consumption of a sequence of tasks. We first assume inter-task voltage scaling, and we consider intra-task voltage scaling in Section 4.1. Next, let us look at a preliminary lemma.

LEMMA 1. *Both the optimal worst-case and expected energy consumption of executing a single task T are proportional to $\frac{1}{d^{\alpha-1}}$ (α is defined in Section 3) where d is the amount of time allotted to execute the task.*

PROOF. Suppose that W is the worst-case number of execution cycles of T, and $P(x)$ is the probability that T executes for x cycles. Obviously, we should use the lowest possible speed ($\frac{W}{d}$) such that T will finish within time d in the worst case. Therefore, the optimal worst-case energy consumption of executing T is

$$p\left(\frac{W}{d}\right) d = \left(\frac{W}{d}\right)^{\alpha} d = \frac{W^{\alpha}}{d^{\alpha-1}}$$

and the optimal expected energy consumption of executing T is

$$\sum_{x=1}^{W} P(x) p\left(\frac{W}{d}\right) \frac{x}{\frac{W}{d}} = \frac{W^{\alpha-1} \sum_{x=1}^{W} P(x)x}{d^{\alpha-1}}$$

which are both proportional to $\frac{1}{d^{\alpha-1}}$. $\square$

Interestingly, the result of Lemma 1 still holds for multiple tasks.

THEOREM 1. *The optimal expected energy consumption of executing N tasks $T_1, T_2, \ldots, T_N$ consecutively is proportional to $\frac{1}{D^{\alpha-1}}$ where D is the amount of time allotted to execute the tasks.*

PROOF. Suppose that the worst-case number of execution cycles of T_i is W_i, and the probability that T_i executes for x cycles is $P_i(x)$. Let the optimal expected energy consumption of executing tasks $T_i, T_{i+1}, \ldots, T_N$ consecutively with allotted time d be denoted by $E(i, d)$. Therefore, we are to prove that $E(1, d)$ is proportional to $\frac{1}{d^{\alpha-1}}$. This can be done by induction.

The base case for $E(N, d)$ is obviously true by Lemma 1.

In the induction step, assume that $E(i + 1, d)$ is proportional to $\frac{1}{d^{\alpha-1}}$, that is, $E(i + 1, d) = \frac{C_{i+1}}{d^{\alpha-1}}$, where C_{i+1} only depends on W_j and $P_j(x)$ ($i + 1 \leq j \leq N$). To compute $E(i, d)$, we first compute the expected energy consumption E' of executing tasks $T_i, T_{i+1}, \ldots, T_N$ with allotted time d when allotting time d' out of time d for task T_i. The running speed for T_i is obviously $\frac{W_i}{d'}$, and the time left for executing tasks $T_{i+1}, T_{i+2}, \ldots, T_N$ is $d - x/\frac{W_i}{d'}$ when T_i has only executed x cycles. Therefore,

$$\begin{aligned} E' &= \sum_{x=1}^{W_i} P_i(x) \left(p\left(\frac{W_i}{d'}\right) x/\frac{W_i}{d'} + \frac{C_{i+1}}{(d - x/\frac{W_i}{d'})^{\alpha-1}} \right) \\ &= \sum_{x=1}^{W_i} P_i(x) \left(x\left(\frac{W_i}{d'}\right)^{\alpha-1} + \frac{C_{i+1}}{(d - \frac{xd'}{W_i})^{\alpha-1}} \right) \end{aligned}$$

Let $d' = \beta d$ where $0 \leq \beta \leq 1$. Thus,

$$\begin{aligned} E' &= \frac{\sum_{x=1}^{W_i} P_i(x) \left(x\left(\frac{W_i}{\beta}\right)^{\alpha-1} + \frac{C_{i+1}}{(1-\frac{x\beta}{W_i})^{\alpha-1}} \right)}{d^{\alpha-1}} \\ &= \frac{\sum_{x=1}^{W_i} P_i(x) x\left(\frac{W_i}{\beta}\right)^{\alpha-1} + \sum_{x=1}^{W_i} \frac{P_i(x)C_{i+1}}{(1-\frac{x\beta}{W_i})^{\alpha-1}}}{d^{\alpha-1}} \\ &= \frac{f(\beta) + g(\beta)}{d^{\alpha-1}} \end{aligned}$$

where $f(\beta) = \sum_{x=1}^{W_i} P_i(x)x\left(\frac{W_i}{\beta}\right)^{\alpha-1}$ and $g(\beta) = \sum_{x=1}^{W_i} \frac{P_i(x)C_{i+1}}{(1-\frac{x\beta}{W_i})^{\alpha-1}}$.

Let

$$C_i = \min_{0 \leq \beta \leq 1} \left(f(\beta) + g(\beta)\right)$$

Then

$$E(i, d) = min(E') = \frac{C_i}{d^{\alpha-1}}$$

Thus proved the claim. $\square$

Theorem 1 shows that the optimal expected energy consumption of a sequence of tasks is of the same form as that of a single task, that is, both are inversely proportional to the $(\alpha - 1)$st power of the allotted time. This is a very powerful result because it enables us to treat a sequence of tasks as a single task. When a sequence of tasks are to be executed, they are partitioned into two parts: the first task and the rest of the tasks, which can be treated as one task. Thus, the problem of allotting time to multiple tasks is reduced to allotting time to just two tasks, which can be efficiently solved thanks to the nice form of the power function. In fact, this is the basic idea of the proof of Theorem 1.

The proof of Theorem 1 indicates that in order to minimize the expected energy consumption of executing a sequence of tasks within a given amount of time t, one should allocate to the first task a fixed percentage of t and set the speed such that the first task is guaranteed to finish within the time allotted to it in the worst case. When the first task finishes, the same procedure can be applied recursively to the rest of the tasks. The proof of Theorem 1 also shows how to go about computing the time allocation percentage for each task. As in the proof, let C_i denote the constant in the optimal expected energy consumption of executing $T_i, T_{i+1}, \ldots, T_N$ consecutively and β_i denote the time allocation percentage for T_i. We compute C_i and β_i in the reverse order. That is, first compute C_N, β_N, then $C_{N-1}, \beta_{N-1}, \ldots$, and last C_1, β_1. The efficiency of the algorithm depends on how to find the minimum value of $f(\beta) + g(\beta)$ as in the proof. In fact, by deriving the first and second derivatives of $f(\beta)$ and $g(\beta)$, we find that $f(\beta)$ is a convex decreasing function and $g(\beta)$ is a convex increasing function. It is easy to show that $f(\beta) + g(\beta)$ is a convex function with only one minimum when $0 \leq \beta \leq 1$. Thus, finding the minimum value of $f(\beta) + g(\beta)$ can be efficiently solved using existing minimization methods, such as the gradient descent. The algorithm for computing time allocation percentages is shown in Figure 1.

```
   ALGORITHM Offline($\{W_i\}, \{P_i(x)\}$)
1. $\beta_N := 1$
2. $C_N := W_N^{\alpha-1} \sum_{x=1}^{W_N} P_N(x)x$
3. for $i := N-1$ downto 1 do
4.     $F(\beta) = \sum_{x=1}^{W_i} P_i(x)x \left(\frac{W_i}{\beta}\right)^{\alpha-1} + \sum_{x=1}^{W_i} \frac{P_i(x)C_{i+1}}{(1-\frac{x\beta}{W_i})^{\alpha-1}}$
5.     $C_i = \min_{0 \leq \beta \leq 1} F(\beta)$
6.     $\beta_i = \arg\min_{0 \leq \beta \leq 1} F(\beta)$
7. return $\{C_i\}$ and $\{\beta_i\}$
   END
```

Figure 1: The offline phase

The algorithm in Figure 1 is done offline. The online scheduling phase is straightforward: when starting executing task T_i and having time d left for executing $T_i, T_{i+1}, \ldots, T_N$, set the speed to $\frac{W_i}{\beta_i d}$.

4.1 Enabling Intra-Task DVS

So far we use a fixed speed for executing a task and never change the speed during the execution. One would wonder whether we can apply the PACE technique to execute a task in our DVS scheme to obtain more energy savings. Theoretically, the answer is positive and it is due to the following theorem.

THEOREM 2. *If intra-task DVS is allowed, then the optimal expected energy consumption of executing T is proportional to $\frac{1}{d^{\alpha-1}}$ where d is the amount of time allotted to execute the task.*

PROOF. Suppose that the worst case number of execution cycles of T is W, and the probability that T executes for x cycles is $P(x)$

Define the cumulative density function, cdf, associated with the number of cycles, X, that task T executes as $cdf(x) = Prob(X \leq x) = \sum_{i=1}^{x} P(i)$ ($cdf(0) = 0$). Let the running speed for the i^{th} cycle of T be f_i. Then the problem of finding the optimal speed schedule of T such that the expected energy consumption is minimized can be expressed as the following mathematical program [9][15]

$$\texttt{Minimize} \quad \sum_{1 \leq i \leq W} (1 - cdf(i-1))\, f_i^{\alpha-1}$$

$$\texttt{Subject to} \quad \sum_{1 \leq i \leq W} \frac{1}{f_i} \leq d$$

and the solution [9][15] is

$$\frac{\left(\sum_{j=1}^{W} (1 - cdf(j-1))^{\frac{1}{\alpha}}\right)^{\alpha}}{d^{\alpha-1}} \tag{1}$$

which is proportional to $\frac{1}{d^{\alpha-1}}$. $\square$

Lemma 1 and Theorem 2 show that the optimal expected energy consumption for executing a task within a given amount of time using a fixed speed is of the same form as that using PACE: they are both inversely proportional to the $(\alpha-1)$st power of the time allotted to the task. We can also see that using PACE for every task in frame-based real-time systems would obtain more energy savings. This is because Theorem 2 can serve as the base case for using PACE as Lemma 1 does for not using PACE, and thus the proof can proceed with just a slight modification (i.e., executing the task using the speed schedule for a single task computed by (1)). Considering the two base cases (Lemma 1 and Theorem 2), since the expected energy consumption using PACE for a single task is less than that using a constant speed [9], by induction it is easy to show that using PACE in our DVS scheme would obtain more energy savings than not using PACE. This also indicates that our DVS scheme can unify intra-task voltage scaling and inter-task voltage scaling in order to minimize the expected energy consumption in the system.

However, applying the PACE technique for executing a task in our DVS scheme is not recommended due to the following reasons:

1. If taking into account the issues discussed in Section 5, applying the PACE technique in practice will result in much more energy consumption compared to the theoretical energy consumption computed by (1) in the proof of Theorem 2. Thus, using PACE could even consume more energy than not using PACE. Our evaluation results in [14] supports this conjecture.

2. Using PACE incurs more online scheduling overhead. This is because before executing a task, the speed schedule for PACE must be recomputed based on the time allotted to the task. The time complexity is $O(BM)$ where B is the number of bins in the histogram and M is number of discrete speeds available in the processor [8]. In fact, the computation of speed schedule is asymptotically inefficient considering that it involves $O(B)$ floating-point calculations.

We present Theorem 2 only for sake of completeness of the theoretical results.

4.2 On the Execution Order in A Frame

If all the tasks in the system are independent, we can execute the tasks in arbitrary order inside a frame. Different order results in different expected energy consumption. Finding the optimal order that results in minimum expected energy consumption appears to be a very hard problem [5]. Currently, we do not know of any solution except for enumerating all orders of the tasks and comparing the corresponding expected energy consumption. When the number of tasks is large, this brute-force approach becomes impractical. Heuristics have been proposed in [4, 5]. The basic idea is to eliminate the biggest uncertainty as early as possible. Through extensive experiments, we found that the quantity $\frac{\sum_{x=1}^{W_i} P_i(x)x}{W_i}$ is a good metric for measuring the uncertainty of task T_i. Thus, we sort the tasks in increasing order of this metric before applying the algorithm in Figure 1 and execute the tasks in the same order during each frame.

5. PRACTICAL CONSIDERATIONS

The algorithm and analysis in Section 4 are theoretically sound. However, when applying this theory to practice, some of the assumptions are no longer valid. In this section, we discuss the issues that arise when our DVS scheme is used in practice and present solutions to these issues.

Granularity In general, it is impractical to store the probability for each cycle considering that a task usually takes millions of cycles. In practice, histogram is used to approximate the probability density function. We treat each bin of the histogram as a *super cycle*. Thus, the analysis in Section 4 remains unchanged for the probability density function being represented by a histogram.

Maximum and Minimum Speed The analysis in Section 4 is based on the assumption that the processor speed can be tuned from 0 to infinity. However, every processor has a maximum speed s_{max} and a minimum speed s_{min}. The speed that is used to execute a task cannot violate this constraint. It is easy to fix this problem. When starting executing task T_i and having time d left, if the speed $\frac{W_i}{\beta_i d} < s_{min}$, then just use s_{min}. Similarly, if the speed $\frac{W_i}{\beta_i d} > s_{max}$, then just use s_{max}. Also, if the resulting speed is less than the speed obtained using the greedy scheme, we will use the greedy-derived speed; this is because greedy guarantees deadlines in the most aggressive form. It is easy to see that the resulting schedule is still valid as long as the tasks can be scheduled using the maximum speed when all tasks take WCEC.

Discrete Speeds So far we assume that the processor speed can be tuned continuously. But real-world processors only provide a finite set of discrete speeds. This problem can be easily fixed due to the fact that any speed can be simulated by using its two adjacent discrete speeds [6].

Speed Change Overhead Currently available commercial processors have speed change overhead, including time penalty and energy penalty. If there are too many speed changes in the system, for example, when using the PACE technique to execute a task as described in Section 4.1, speed change overhead cannot be ignored in designing DVS schemes. Since we do not recommend using PACE in our scheme, the number of speed changes in our scheme is at most N for continuous frequency and $2N$ for discrete frequency, we can ignore the speed change overhead in general (or the time penalty is subtracted from the frame length).

Different Energy Consumption for Different Instruction In reality, different machine instructions consume different energy. Thus, the power consumption of executing a task not only depends on the running frequency, but also the instructions executed by the task. To capture this, we can use different c_1 for different tasks in the power function $c_0 + c_1 f^{\alpha}$. It is easy to see that our analysis can be done with only a slight modification. Note that all the existing schemes cannot capture this subtlety.

6. CONCLUSIONS

In this paper, we study the problem of minimizing the expected energy consumption in frame-based real-time embedded systems. We make the following contributions: (1) We show that the dynamic behavior of tasks needs to be captured in the speed schedule in order to minimize the expected (total) energy consumption in the system; (2) We provide an optimal DVS scheme for processors whose speed and voltage can be tuned continuously; (3) We extend the theoretical framework to take into consideration the practical issues, devising efficient solutions to the problems. Simulation results [14] using Intel XScale processor model on both synthetic and real-life workloads show that our new DVS scheme achieves a significant energy savings over the existing schemes; (4) We provide the theoretical results on using PACE for multiple tasks that share the same deadline, which is an open problem put forth in [9]. We also discuss the practicality of the theoretical results and validate it through simulation [14].

Future work will investigate the case of the problem where different tasks have different deadlines.

7. REFERENCES

[1] N. AbouGhazaleh, D. Mossé, B. Childers, R. Melhem, and Matthew Craven. Collaborative Operating System and Compiler Power Management for Real-Time Applications. In *The 9th IEEE Real-Time Embedded Technology and Applications Symposium(RTAS 2003)*, May 2003.

[2] H. Aydin, R. Melhem, D. Mossé, and P. Mejia-Alvarez. Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS'01)*, pages 95–105, December 2001.

[3] R. Ernst and W. Ye. Embedded Program Timing Analysis based on Path Clustering and Architecture Classification. In *International Conference on Computer Aided Design (ICCAD'97)*, San Jose, CA, 1997.

[4] F. Gruian. On Energy Reduction in Hard Real-Time Systems Containing Tasks with Stochastic Execution Times. In *IEEE Workshop on Power Management for Real-Time and Embedded Systems*, Taipei, Taiwan, May 2001.

[5] F. Gruian and K. Kuchcinski. Uncertainty-Based Scheduling: Energy Efficient Ordering for Tasks with Variable Execution Time. In *Proceedings of International Symposium on Low Power Electronics and Design 2003 (ISLPED)*, Seoul, Korea, August 2003.

[6] T. Ishihara and H. Yasuura. Voltage Scheduling Problem for Dynamically Variable Voltage Processors. In *International Symposium on Low Power Electronics and Design*, pages 197–202, Aug. 1998.

[7] W. Kim, D. Shin, H. Yun, J. Kim, and S. Min. Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2002.

[8] J. Lorch. *Operating Systems Techniques for Reducing Processor Energy Consumption*. PhD thesis, University of California at Berkeley, 2001.

[9] J. Lorch and A. Smith. Improving Dynamic Voltage Scaling Algorithms with PACE. In *ACM SIGMETRICS*, June 2001.

[10] J. Lorch and A. Smith. Operating system modifications for task-based speed and voltage scheduling. In *the First International Conference on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

[11] D. Mossé, H. Aydin, B. Childers, and R. Melhem. Compiler-Assisted Dynamic Power-aware Scheduling for Real-Time Applications. In *Workshop on Compiler and OS for Low Power (COLP)*, October 2000.

[12] C. Rusu, R. Xu, R. Melhem, and D. Mossé. Energy-Efficient Policies for Request-Driven Soft Real-Time Systems. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, Catania, Italy, July 2004.

[13] N.H.E. Weste and K. Eshraghian. *Principles of CMOS VLSI Design*. Addison-Wesley, Reading, MA, 1993.

[14] R. Xu, D. Mossé, and R. Melhem. Minimizing Expected Energy in Real-Time Embedded Systems. Technical Report TR-05-125, Department of Computer Science, University of Pittsburgh, 2005. http://www.cs.pitt.edu/∼xruibin/publications/TR-05-125.pdf.

[15] R. Xu, C. Xi, R. Melhem, and D. Mossé. Practical PACE for Embedded Systems. In *Proceedings of the 4th ACM International Conference on Embedded Software*, Pisa, Italy, September 2004.

[16] F. Yao, A. Demers, and S.Shankar. A Scheduling Model for Reduced CPU Energy. In *IEEE Annual Foundations of Computer Science*, pages 374–382, 1995.