An Efficient Dynamic Task Scheduling Algorithm for Battery Powered DVS Systems*

Jianli Zhuo
Dept. of Electrical Engineering
Arizona State University
Tempe, AZ, 85287, USA
E-mail: jianli.zhuo@asu.edu

Chaitali Chakrabarti
Dept. of Electrical Engineering
Arizona State University
Tempe, AZ, 85287, USA
E-mail: chaitali@asu.edu

Abstract— Battery lifetime enhancement is a critical design parameter for mobile computing devices. Maximizing battery lifetime is a particularly difficult problem due to the non-linearity of the battery behavior and its dependence on the characteristics of the discharge profile. In this paper we address the problem of dynamic task scheduling with voltage scaling in a battery-powered DVS system. The objective is to maximize the battery performance measured in terms of charge consumption during execution of the tasks. We present a new battery-aware dynamic task scheduling algorithm, *darEDF*, based on an efficient slack utilization scheme that employs dynamic speed setting of tasks in run queue. We compare *darEDF* with three state of the art energy-efficient algorithms, *lpfpsEDF*, *lppsEDF*, *lpSEH*, with respect to battery performance and energy consumption. We show that *darEDF* has better performance than *lpSEH* (which has close to optimal energy value), and has lower run-time complexity.

I. Introduction

Battery-operated portable devices are widely used in mobile computing and wireless communication applications. Maximizing battery lifetime is the most important design metric for such systems. This problem is quite challenging due to the non-linear behavior of the battery. Since the amount of energy delivered by the battery depends on the discharge current profile [1], the battery life can be extended by controlling the discharge current level and shape. In this paper, we propose an approach based on modifying the discharge current profile of real time tasks during task scheduling on a DVS (Dynamic Voltage Scalable) processor such that the charge consumption during task execution is minimized.

Task scheduling for DVS processors has been studied extensively in recent years. The model assumes that the processor is connected to an infinite source of energy. Strategies that have been developed to reduce the energy consumption of such models do not work well for limited energy sources like batteries. Furthermore, batteries exhibit non-linear discharge behavior that need to be exploited.

In recent years, there has been significant amount of work done in studying battery characteristics [1] and using these characteristics to shape the discharge profile [2], [3]. All of the earlier work on battery aware task scheduling ([2], [3], [4]) has been for static tasks where complete information about the tasks is known apriori. To the best of our knowledge, this is the first attempt at battery-aware dynamic task scheduling.

Dynamic task scheduling has been extensively investigated in the context of ideal power sources. Several of the algorithms [5], [6], [7], [8] operate in two phases: off-line phase (based on WCET or other execution time estimates) followed by an online phase where the slack is greedily absorbed. The accuracy of the off-line assignment is increased in [6] by employing search techniques, in [7] by using stochastic estimates and in [8] by using static timing analysis. The 'average rate heuristic' proposed in [9] is an online EDF algorithm which absorbs the slack better than greedy algorithms by considering the speed requirements of all ready tasks. The on-line slack estimation heuristic proposed for dynamic-priority task scheduling (e.g. EDF) in [10] and fixed-priority task scheduling (e.g. RM) in [11] distribute the slack more evenly among tasks and have superior performance. Better slack utilization is also achieved in [8] by intra task scheduling.

This paper proposes a new EDF based scheduling algorithm for periodic tasks that has superior energy and battery performance. The algorithm employs a dynamic version of the average rate heuristic first proposed in [9], and delays slack absorption, resulting in better battery performance. The main contributions of this work are:

(1) Development of a new battery-aware dynamic algorithm, *darEDF*, with an efficient speed setting mechanism.

(2) Comparison with three existing energy-efficient algorithms (*lpfpsEDF* [12], *lppsEDF* [5], *lpSEH*[10]) with respect to battery performance and energy consumption. *darEDF* has better energy and battery performance than *lpSEH*[10] (which has near-optimal energy performance [13]), and has lower run time complexity compared to *lpSEH*.

The rest of the paper is organized as follows. The paper begins with preliminaries like system configuration, battery models, etc in Section II. In Section III, the new scheduling algorithm is presented with a illustrative example. The simulation results with random examples are described in Section IV. The paper is concluded in Section V.

II. Background

A. System Level Configuration

The system configuration for the battery-operated processor under consideration is described in Fig 1. The system consists of one DVS processor driven by a single battery. The battery is used to power the processor through a DC-DC converter. The

*This research was funded in part by the NSF-S/IUCRC Center for Low Power Electronics (Grant No. EEC-9523338), and by the NSF/IUCRC center, Connection One.

DC-DC converter has an efficiency η which is typically in the range [0.8,0.9]. The efficiency $\eta = \frac{I_{proc}V_{proc}}{I_{batt}V_{batt}}$, where V_{batt} and I_{batt} are the battery voltage and current and V_{proc} and I_{proc} are the processor voltage and current.

Fig. 1. System Level Configuration.

Throughout the paper, we assume that the change in voltage is always associated with a change in frequency. For a long channel CMOS gate with threshold voltage V_t and supply voltage V_{dd}, voltage scaling by a factor of s causes the processor current I_{proc} to scale by a factor of s^2. Assume that the DC-DC conversion efficiency η and the battery voltage V_{batt} are averaged constants for the duration of a task execution, then the battery current I_{batt} scales by ηs^3 [2]. Thus, scaling results in a large drop in the battery load and leads to significant maximization of the residual charge at the end of a task profile.

B. Battery Model and Its Non-linear Properties

In this work, we have considered a high-level accurate charge based model [14]. The model gives an analytical relationship between the current load, discharge time and the charge consumed due to the discharge. For a time-varying load represented by a piecewise approximate function $\{I_k, \Delta_k: \Delta_k = t_k - t_{k-1}, t_N \leq T, k = 1, 2..N\}$, the charge consumption C in duration T is given by

$$C = \sum_{k=1}^{N} I_k \times (\Delta_k + 2 \sum_{m=1}^{\infty} \frac{e^{-\beta^2 m^2 (T - t_k - \Delta_k)} - e^{-\beta^2 m^2 (T - t_k)}}{\beta^2 m^2}) \quad (1)$$

β^2 is related to the diffusion rate within the battery and captures its nonlinearity. In our simulations, we have used the Li-ion battery used in DUALFOIL[3] with $\beta = 0.273 min^{-1/2}$.

We evaluate the battery performance by its charge consumption. A lower value of charge consumption implies better battery behavior.

There are several important properties of the battery with respect to voltage scaling that have been derived from the analytical model [2], [3]. Two of those properties have been used here for developing the real-time scheduling heuristics.

Property 1: *Scaling voltage to absorb slack is more battery efficient than using the slack for battery recovery.*

Property 2: *A decreasing current profile is battery-efficient.*

III. Battery-aware dynamic task scheduling

A. Task Definition

In this paper, we consider periodic tasks in which the relative deadline of a task is equal to its period. We use T_k to denote the $k-$th periodic task in the task set, $T_{k,i}$ is the $i-$th instance (released in the $i-$th period) of task T_k. Task T_k is associated with the following parameters: the current I_k, the arrival time a_k, the start time t_k, the deadline d_k, the period P_k, the worst case execution time $WCET_k$, and the actual execution time AET_k. $WCET_k$ and AET_k is the required time when task is run in the highest voltage.

The slack associated with a task is due to two factors: (1) the inherent slack due to the difference between the deadline and the $WCET$ and (2) the slack generated due to the actual execution time being less than the worst case execution time i.e slack due to workload variation.

B. Dynamic Scheduling Algorithm darEDF

B.1 Dynamic Average Rate

Slack Forwarding : Slack forwarding is a derivative of battery property 2, according to which, it is better to make the current profile decreasing with time. By not greedily absorbing the slack but letting the later tasks absorb it, this method results in later tasks having a lower current. NTA (stretch to the arrival of next task) used in the algorithm *lpfps* [12] is an example of slack forwarding.

Slack Lookahead : Because of the difference between AET and $WCET$, and the limitation of the number of possible scaled voltage values, the amount of slack that is forwarded maybe much higher than the absorption ability of the last task. This will cause increase in system idle time, which is neither energy efficient nor battery efficient (see property 1). In order to compensate for the lost slack, algorithm *lpps* [5] applies global scaling before NTA. As a result, slack is more evenly distributed, and then NTA is used to absorbs the slack further.

Dynamic Average Rate : We combine the concepts in slack forwarding and slack lookahead and propose a new task speed setting algorithm called dynamic average rate. The speed setting in traditional AVR (*average-rate heuristics*) is given by the summation of the densities (WCET/Period) of all live tasks [9]. In our algorithm, AVR is modified to DAR (*Dynamic Average Rate*), where the task density is updated whenever the run queue is changed. The density of a live task is not the original value given in [9] but set to the residual $WCET$ divided by residual available time (time before deadline). As a result, the slack generated by the difference of AET and $WCET$ of the previous task is taken into the speed calculation of the current task. Assume task j is the active task (task instance executed by processor at current time), its execution speed is given by

$$s(t_j) = \sum_{k \in run\_queue} \frac{Residual\_WCET_k}{d_k - t_j} \quad (2)$$

where the run_queue includes all the task instances ready to be executed (arrival time passed). $Residual\_WCET_k$ is the amount of WCET that is yet to be executed (a parameter that is required for task preemption), d_k is absolute deadline of a task, t_j is the start time of task j.

DAR is battery-efficient in the following two ways: 1) It absorbs the slack as much as possible thereby resulting a lower battery load current. 2) Because of the difference between AET and $WCET$, the DAR speed setting for the tasks in a run queue is expected to result in a decreasing current profile. A detailed analysis is given in section B.4.

B.2 Dynamic Scheduling Algorithm darEDF

The algorithm proposed in this paper, *darEDF*, uses DAR in dynamic EDF scheduling. This algorithm consists of: (1) moving all released task into the run queue; (2) selecting the active task and determining its speed by DAR; (3) executing the active task. The flow chart of *darEDF* is given in Fig 2.

B.3 Schedulability

darEDF results in a feasible schedule if the task set is EDF-schedulable. This is because the relative deadline of the periodic task equals the period and there is at most one instance of

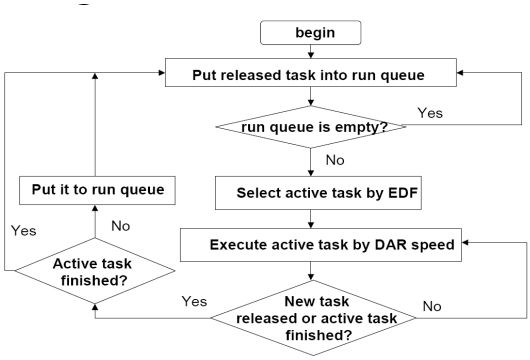

Fig. 2. Flow chart of dynamic scheduling algorithm darEDF

each task in the run queue, so μ of the run queue tasks still satisfies the feasibility condition. From the definition of *DAR*, if we scale all the live tasks in the run queue by speed $s(t_j)$ (see equation (2), then the run queue has utilization $\mu = 1$. However we only scale the active task, so $\mu \leq 1$, and the schedulability condition still holds for *darEDF* [15].

B.4 Mathematical Analysis

The following analysis gives the relative performance comparison of *lpfpsEDF*, *lppsEDF* and our algorithm *darEDF*. The comparison is with respect to slack utilization since larger the slack utilization, lower the current and better the battery performance. Comparing using charge consumption given in equation (1) is cumbersome.

To simplify the proof (without loss of generality), we assume there are 3 periodic tasks(T_1, T_2, T_3) with only one instance, and they are all released at time 0, so all of them are in the run queue. The parameters of a task include: deadline d_i ($i = 1, 2, 3$, and $d_1 \leq d_2 \leq d_3$), worst case execution time $WCET_i$, and actual execution time AET_i. We define $w_i = \frac{WCET_i}{d_i}$ and $h_i = \frac{AET_i}{WCET_i}$, where w_i is task density satisfying constraint $\mu_{edf} = w_1 + w_2 + w_3 \leq 1$, and $h_i \leq 1$.

Let $t_{i_{algx}}$ and $\tau_{i_{algx}}$ denote start time and scaled execution time of task i determined by Algorithm $algx$ respectively . Our objective is to maximize the summation of scaled execution time $EX_{algx} = \tau_{1_{algx}} + \tau_{2_{algx}} + \tau_{3_{algx}}$.

We assume the DVS processor is continuously scalable.

***lpfpsEDF*:** Tasks T_1 and T_2 are executed at full speed, only T_3 is scaled. The task start time $t_{il_{pfps}}$ and scaled execution time $\tau_{i_{lpfps}}$ for each task are

$$t_{1_{lpfps}} = 0, \tau_{1_{lpfps}} = AET_1; t_{2_{lpfps}} = \tau_{1_{lpfps}}, \tau_{2_{lpfps}} = AET_2;$$
$$t_{3_{lpfps}} = \tau_{1_{lpfps}} + \tau_{2_{lpfps}}, \tau_{3_{lpfps}} = h_3 \times (d_3 - t_{3_{lpfps}})$$

The summation of scaled task execution time is

$$EX_{lpfps} = h_3 \times d_3 + (1 - h_3) \times (AET_1 + AET_2)$$

***lppsEDF*:** It scales all tasks based on the utilization μ_{edf} initially, then scales task T_3 further. The task start time and scaled task execution time is

$$t_{1_{lpps}} = 0, \tau_{1_{lpps}} = \mu_{edf}^{-1} \times AET_1; t_{2_{lpps}} = \tau_{1_{lpps}}, \tau_{2_{lpps}} = \mu_{edf}^{-1} \times AET_2;$$
$$t_{3_{lpps}} = \tau_{1_{lpps}} + \tau_{2_{lpps}}, \tau_3 = h_3 \times (d_3 - t_{3_{lpps}});$$
$$EX_{lpps} = h_3 \times d_3 + (1 - h_3) \times \mu_{edf}^{-1} \times (AET_1 + AET_2)$$

$EX_{lpps} \geq EX_{lpfps}$ because of $\mu_{edf} \leq 1$, so *lppsEDF* does better slack utilization compared to *lpfpsEDF*.

***darEDF*:** Task T_1 is selected as active task first because of its earliest deadline. Its start time is $t_{1_{dar}} = 0$, speed and scaled task execution time for T_1 are

$$s(t_{1_{dar}}) = \sum_{j=1}^{3} \frac{Residual\_WCET_j}{d_j - t_{1_{dar}}} = w_1 + w_2 + w_3 = \mu_{edf}$$
$$\tau_{1_{dar}} = s(t_{1_{dar}})^{-1} \times AET_1 = \mu_{edf}^{-1} \times AET_1$$

Task T_2 is selected as active task at time $t_{2_{dar}} = \tau_{1_{dar}}$ when task T_1 is finished. The task speed and scaled task execution time are

$$s(t_{2_{dar}}) = \sum_{j=2}^{3} \frac{Residual\_WCET_j}{d_j - t_{2_{dar}}} = (\frac{WCET_2}{d_2 - \tau_{1_{dar}}} + \frac{WCET_3}{d_3 - \tau_{1_{dar}}})$$
$$\tau_{2_{dar}} = s(t_{2_{dar}})^{-1} \times AET_2 = (\frac{WCET_2}{d_2 - \tau_{1_{dar}}} + \frac{WCET_3}{d_3 - \tau_{1_{dar}}})^{-1} \times AET_2$$

Task T_3 is started at $t_{3_{dar}} = \tau_{1_{dar}} + \tau_{2_{dar}}$, the task speed and scaled task execution time are

$$s(t_{3_{dar}}) = \sum_{j=3}^{3} \frac{Residual\_WCET_j}{d_j - t_{3_{dar}}} = \frac{WCET_3}{d_3 - \tau_{1_{dar}} - \tau_{2_{dar}}}$$
$$\tau_{3_{dar}} = s(t_{3_{dar}})^{-1} \times AET_3 = h_3 \times (d_3 - \tau_{1_{dar}} - \tau_{2_{dar}})$$

The total scaled execution time is

$$EX_{dar} = h_3 \times d_3 + (1 - h_3) \times (\mu_{edf}^{-1} \times AET_1 + s(t_{2_{dar}})^{-1} \times AET_2)$$

$EX_{dar} \geq EX_{lpps}$ if we can prove $s(t_{2_{dar}}) \leq \mu_{edf}$.

$$s(t_{2_{dar}}) = \frac{WCET_2}{d_2 - \tau_{1_{dar}}} + \frac{WCET_3}{d_3 - \tau_{1_{dar}}}$$
$$\leq \frac{WCET_2}{d_2 - \mu_{edf}^{-1} \times WCET_1} + \frac{WCET_3}{d_3 - \mu_{edf}^{-1} \times WCET_1}$$
$$\leq \frac{w_2 \times d_2}{d_2 - \mu_{edf}^{-1} \times w_1 \times d_2} + \frac{w_3 \times d_3}{d_3 - \mu_{edf}^{-1} \times w_1 \times d_3} = w_1 + w_2 + w_3 = \mu_{edf}$$

Hence *darEDF* has better slack utilization than both Algorithm *lppsEDF* and Algorithm *lpfpsEDF*.

Battery efficiency of *darEDF* : From the above analysis, we can see that $s(t_{2_{dar}}) \leq s(t_{1_{dar}})$ because $s(t_{2_{dar}}) \leq \mu_{edf}$ and $s(t_{1_{dar}}) = \mu_{edf}$. Similarly, we can show that $s(t_{3_{dar}}) \leq s(t_{2_{dar}})$. So the speed of the tasks in the run queue decreases with time. This results in a decreasing current profile. Note that algorithms *lpfpsEDF*, *lppsEDF*, *lpSEH* also result in a locally decreasing current profile.

C. Illustrative Example

We consider tasks T_1, T_2, T_3, with $P_k = d_k$ given by 2 min, 3 min and 6 min, $WCET$ given by 1 min, 1 min, 1 min, AET given by 0.25 min, 0.25 min, 0.5 min respectively. The processor used in the simulation is the StrongArm SA1100, which has 11 frequency values range from 206.4MHz to 59MHz. The voltage and current corresponding to 206.4MHz are 1.5V and 237.8mA [16].

The voltage profiles generated by *lpfpsEDF*, *lpfpsEDF*, *lpSEH*, *darEDF* are shown in Fig 6 along with the corresponding charge consumption C and energy consumption E. Note that *darEDF* consumes the least charge and least energy.

IV. Experimental Results

To evaluate the battery efficiency and energy efficiency of the proposed algorithm *darEDF*, we selected 3 existing algorithms: 1) *lpfpsEDF* [12] which implements slack forwarding;

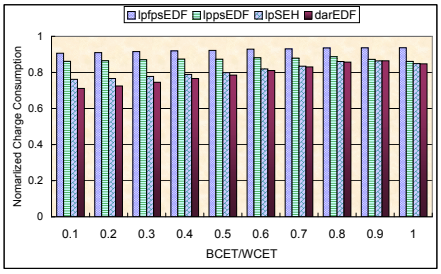

Fig. 3. Average CPU Charge Consumption

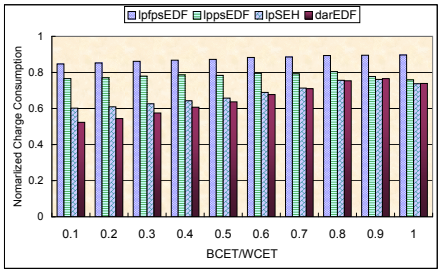

Fig. 4. Average CPU Energy Consumption

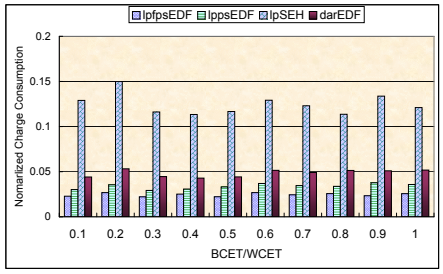

Fig. 5. Comparison of complexity in CPU time

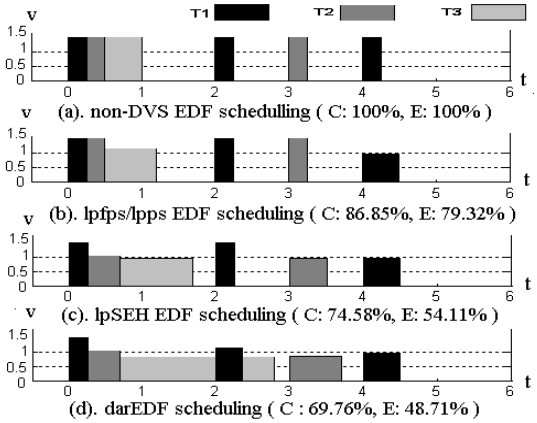

Fig. 6. Voltage profile of the example

2)*lppsEDF* [5] which implements slack lookahead as well as slack forwarding; 3) *lpSEH* algorithm [10] which uses slack estimation heuristics to absorb slack.

We use StrongArm SA1100 with 11 frequency values used in illustrative example. We assume that the processor does not consume energy when it is idle and the overhead of moving from one voltage to another is negligible.

Simulation results are presented for the tasks in the Videophone system listed in [10] . The number of periodic tasks is 4, the periods vary from 40 ms to 66.7 ms, and the $WCET$s vary from 1.4 ms to 50.4 ms. We use $BCET$ to denote the best case execution time; the actual execution time AET is given by a Gaussian distribution with mean $m = \frac{WCET+BCET}{2}$ and variance $\sigma = \frac{WCET-BCET}{6}$.

We vary the $\frac{BCET}{WCET}$ from 0.1 to 1.0 with a step of 0.1 and run 100 random task sets by all algorithms for each $\frac{BCET}{WCET}$ value, and compare the average normalized energy and charge consumption (normalized to the non-DVS case). All the task sets are run in a hyper-period (i.e. the L.C.M of all periods). The simulation results are shown in Fig 3 and Fig 4. A more detailed comparison of the performance of the individual algorithms shows *darEDF* has the best performance both with respect to charge consumption and energy consumption.

An analysis of the run time complexity reveals that algorithm *lpfps*, *lpps* and *darEDF* are a lot simpler compared to *lpSEH*. Fig 5 shows the average CPU time consumed by each algorithm. Algorithm *lpfps* and *lpps* do the initial speed setting offline and have an even lower time complexity.

V. Conclusion

In this paper we addressed the problem of dynamic task scheduling for battery operated single processor DVS systems. We (i) presented a new battery-aware algorithm *darEDF*, and (ii) compared its performance with 3 competing algorithms, *lpfpsEDF*, *lppsEDF*, *lpSEH*, with respect to charge consumption and energy consumption. We found that the trends in charge consumption are similar to the trends in energy consumption. For the Videophone benchmark run on StrongArm SA1100, our algorithm has the best performance both in energy consumption and charge consumption. Furthermore, our algorithm has much lower run-time complexity than *lpSEH* which has been proven to have close to optimal energy saving [13].

References

[1] T.Martin, "Balancing batteries,power,and performance:System issue in CPU speed-setting for mobile computing," *Ph.D Dissertation*, 1999.

[2] P.Chowdhury and C.Chakrabarti, "Battery-aware task scheduling for a system-on-chip using voltage/clock scaling," *Proc.SiPS*, pp. 201–206, 2002.

[3] D.Rakhmatov, S.Vrudhula , "Energy management for battery-powered embedded systems," *ACM Transactions on Embedded Computing Systems*, pp. 277–324, Aug. 2003.

[4] J.Luo and N.Jha, "Battery-aware static scheduling for distributed real-time embedded systems," *Proc. DAC*, pp. 444 –449, 2001.

[5] Y.Shin, K.Choi, and T.Sakurai, "Power optimization of real-time embedded systems on variable speed processors," *Proc. ICCAD*, pp. 365–368, 2000.

[6] C.M Krishna and Y.H Lee, "Voltage-clock-scaling adaptive scheduling techniques for low power in hard real time systems," *Proc. Real-Time Technology & Applications Symp.*, pp. 156–165, 2000.

[7] F. Gruian, "Hard real-time scheduling for low energy using stochastic data and DVS processors," *Proc. ISLPED*, pp. 46–51, 2001.

[8] D. Shin, J.Kim and S. Lee, "Low-Energy intra task voltage scheduling using static timing analysis," *Proc. DAC*, pp. 438–443, 2001.

[9] F.Yao, A.Demers, and S.Shenker, "A scheduling model for reduced cpu energy," *IEEE Annual Foundations of Computer Science*, pp. 374–382, 1995.

[10] W. Kim, J.Kim and S. Min, "A dynamic voltage scaling algorithm for dynamic-priority hard real-time systems using slack time analysis," *Proc. DATE.*, pp. 788–794, 2002.

[11] Woonseok Kim, Jihong Kim and Sang Lyul-Min, "Dynamic voltage scaling algorithm for fixed-priority real-time systems using work-demand analysis," *Proc. ISLPED.*, pp. 396–401, 2003.

[12] Y.Shin and K.Choi, "Power conscious fixed priority scheduling for hard real-time systems," *Proc. DAC*, pp. 134–139, 1999.

[13] W. Kim, D. Shin, H. Yun, J. Kim and S. Min, "Performance comparison of dynamic voltage scaling algorithms for hard real-time systems," *Proc. RTAS.*, pp. 219–228, 2002.

[14] D.Rakhmatov and S.Vrudhula, "An analytical high-level battery model for use in energy management of portable electronic systems," *Proc. ICCAD*, pp. 488–493, 2001.

[15] C.M. Krishna, Kang G. Shin, "Real-time systems," *New York : McGraw-Hill*, 1997.

[16] Amit Sinha, "JouleTrack : A web based software energy profiling tool," *http://carlsberg.mit.edu/jouletrack/JouleTrack/.*