BATTERY AWARE TASK SCHEDULING FOR A SYSTEM-ON-A-CHIP USING VOLTAGE/CLOCK SCALING

Princey Chowdhury & Chaitali Chakrabarti

Arizona State University
Center for Low Power Electronics
Department of Electrical Engineering
Tempe AZ 85287

ABSTRACT

Battery lifetime is a critical parameter in the operation of mobile computing devices. The lifetime of such devices is directly dependent on the battery discharge profile. In this paper we address the problem of task scheduling in single processor and multiprocessor systems such that the battery lifetime is maximized. We propose a procedure that achieves this by shaping the current load profile. The shaping algorithm makes extensive use of voltage/clock scaling and is guided by heuristics that are derived from the properties of the battery model. Simulations show that the proposed algorithm improves the battery lifetime significantly.

1. INTRODUCTION

Battery operated portable devices are widely used in many mobile computing and wireless communication applications. Battery lifetime maximization is the most important design metric for such systems. Battery behavior is non-linear: the amount of energy delivered by a battery depends on the discharge current profile. Thus battery lifetime can be extended by controlling the discharge current level and shape. In this paper, we address the problem of task scheduling on battery operated single processor and multiprocessor systems. Our procedure is based on shaping the current profile so that the battery lifetime is maximized.

The areas of research most closely related to this paper are non-ideal battery behavior [1, 2, 3, 4, 5] and dynamic voltage scheduling for low power [6, 7, 8, 9]. In dynamic voltage task scheduling, the operating voltage of the processor is dynamically lowered whenever a slack is identified. The objective is to minimize the energy consumption of the processor core and the capacity of the power source is not taken into account. However since battery capacity is finite and is a function of both the load and the discharge time, the existing dynamic voltage task scheduling approaches are not applicable for a battery powered device.

The non-linearity of the battery behavior and its effect on the CPU speed setting were first demonstrated by Martin and Sewiorek [2]. The non-linearity of the battery has been utilized for modulation scaling in designing energy aware communication systems in [4] and for traffic shaping in [5]. Benini et.al [10, 11]studied battery-aware time out and threshold based dynamic power management policies. Luo and Jha [12] considered static scheduling of tasks in a battery operated multiprocessor environment. The proposed method was evaluated based on the battery model combining Peukert's law and the ideas of Pedram and Wu [3]. The battery-optimized schedule was achieved by reducing the variance and the peak power of a generated discharge current profile. Our work is closest to the [12]; however we use Li-Ion based battery model of [13] to guide the scheduling process. While the model in [12] is purely empirical, the model from [13] is based on simplified physical analysis of the battery behavior and accurately accounts for many non-linear effects such as charge recovery.

In this paper we treat the problem of task scheduling of battery operated devices as that of shaping the current load profile so that the residual charge after the successful completion of a task set is as large as possible. The scheduling algorithm is guided by heuristics that are derived from the properties of the battery model. The algorithm operates in two phases: In the first phase, a feasible schedule (without battery failure) is generated. In the second phase, voltage scaling is used to utilize the available slack. Since voltage scaling by a factor of s translates to battery current scaling by a factor of s^3, this procedure helps in improving the battery lifetime significantly.

The rest of the paper is organized as follows. Section 2 is a brief discussion of the battery model, cost function and other preliminaries for a better understanding of the scheduling technique. The proposed task scheduling technique for a single processor and multi-processor systems are described in Section 3 and 4 respectively. Section 5 concludes the paper.

2. BACKGROUND

2.1. Battery Model

The battery model used in this paper has been described in detail in [13]. In this section we briefly outline the key assumptions, derivations and results. There are two battery specific parameters: α and β. These parameters are estimated statistically from the battery profiling data. The battery model can be described as:

$$\alpha = \int_0^L \left[1 + 2\sum_{m=1}^{\infty} e^{-\beta^2 m^2 (L-\tau)}\right] i(\tau) d\tau \quad (1)$$

where $i(t)$ is the discharge current and L is the battery time to failure or lifetime. Thus the parameter α can be loosely termed as the battery charge capacity before the battery started to discharge and β is the measure of the battery non-linearity. In this paper we have used the Li-Ion battery in the Compaq Itsy Pocket Computer whose α and β values are 35220 and 0.637 respectively [14]. For a given load profile of duration T, let

$$\sigma = \int_0^T \left[1 + 2\sum_{m=1}^{\infty} e^{-\beta^2 m^2 (T-\tau)}\right] i(\tau) d\tau \quad (2)$$

If the discharge current $i(t)$ is expressed by a set of step functions, then σ can be expressed as

$$\sigma = \sum_{k=0}^{n-1} I_k \left[\Delta_k + 2\sum_{m=1}^{\infty} \frac{e^{-\beta^2 m^2 (T-t_k-\Delta_k)} - e^{-\beta^2 m^2 (T-t_k)}}{\beta^2 m^2}\right] \quad (3)$$

The battery model has been experimentally validated against realistic data, details of which can be found in [15, 16]. It was tested using constant, interrupted, periodic and non-periodic discharge profiles derived from standard applications run on a pocket computer. The quality of the model was evaluated with respect to the results produced from the low-level simulator DUALFOIL [17, 18]. For the constant loads, the predictions were within 10%, and for the variable loads, the maximum error was less than 5%.The Peukert's model though pretty accurate for a constant load, had an error margin of more than 100% for variable load cases [16].

2.2. Cost Function

Our objective is to maximize the charge slack (α-σ) referred to as a profile quality metric Q. Intuitively, Q is the *instantaneous* charge slack (the amount of charge less than the capacity α). Battery failure can be estimated from the sign of Q; Q negative implies that the battery has failed while executing some load and a Q positive means a successful completion of a task set.

2.3. Voltage Scaling

The system configuration for a battery-operated device in a single processor system is described in figure 1. The battery voltage and current are denoted by V_{batt} and I_{batt} while the operating voltage of the processor and the current drawn by the processor are denoted by V_{proc} and I_{proc} respectively. The conversion efficiency of the DC-DC converter is $\eta = \frac{I_{proc} V_{proc}}{I_{batt} V_{batt}}$. For a $CMOS$ gate with threshold voltage V_t, supply voltage V_{dd} and velocity saturation index γ , the delay is proportional to $V_{dd}/[V_{dd} - V_t]^{\gamma}$. In our calculations, we assume γ to be 2. Assuming that the efficiency η is considered to be a constant in the region of operation, since I_{proc} is proportional to $(V_{proc})^2$, I_{proc} scales by s^2 as a result of voltage/clock scaling. Further since V_{batt} is considered constant , the battery current I_{batt} scales by s^3. Thus voltage scaling by a factor of s at the processor level leads to battery current scaling by a factor of s^3. Thus slack utilization by voltage scaling leads to significant battery lifetime improvement.

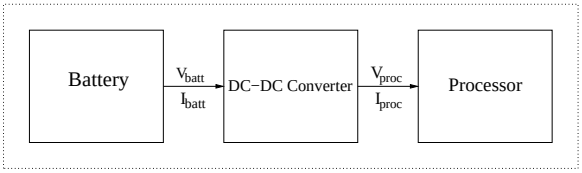


Fig. 1. System Level Configuration for a Single Processor System.

3. SCHEDULING ON A SINGLE PROCESSOR SYSTEM

3.1. Scheduling Algorithm

A given task is associated with three parameters: the average current I_k, the average duration D_k and the start time t_k. Task scheduling with voltage scaling affects not only t_k but also I_k and D_k. Given the battery parameters α and β, the task profile and the set of discrete voltages that the processor can operate on, the scheduling algorithm provides a feasible schedule that maximizes the quality metric Q. The constraints are as follows: (1) The task deadlines are not violated, and (2) at any time the battery is alive. The assumption here is that the system can operate at any of the discrete voltages V_i from the set $S_V = \{V_0, V_1, ..., V_{m-1}\}$ as supported by the DC-DC converter.

The heuristic scheduling algorithms are based on the following three theorems derived from the properties of the battery model. The proofs of the theorems have been omitted due to lack of space.

Theorem 1: *For a fixed voltage assignment (only task start times can be changed), sequencing tasks in the non-*

increasing order of their currents is optimal.[15]

Theorem 2: *If a battery fails during some task k, it is always cheaper to repair it by down-scaling its voltage than by inserting an off-line period before k.*

Theorem 3: *Given a pair of two identical tasks in the profile and a delay slack to be utilized by voltage down-scaling, it is always better to use the slack on the later task than on earlier task.*

The proposed scheduling algorithm operates in two phases. In the first phase, a feasible schedule is obtained by (i) using the earliest deadline first EDF algorithm, (ii) trying to generate a non-increasing order of loads, and (iii) ensuring that there is no failure during the battery discharge. In the second phase, the algorithm continues voltage down-scaling in order to fully utilize the available delay slack. Figure 2 describes the proposed algorithm .

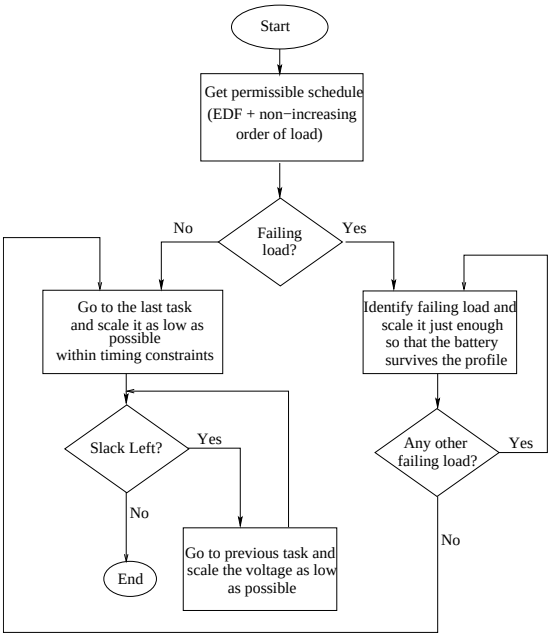


Fig. 2. Top level view of the algorithm for single processor system.

Phase One: Generating a Feasible Schedule
At the start of this phase, all the tasks are assigned to the highest voltage. First the tasks are arranged according to the EDF policy and then a greedy approach is used to reschedule them so that a non-increasing order of task currents is obtained if possible. This step is justified by Theorem 1. The algorithm then checks the Q value of the new schedule. If $Q < 0$ the failure recovery procedure is called to repair the battery failure. Once the Q value is positive, the slack utilization procedure is called.

Failure Recovery: This procedure reshapes the profile so that the battery survives the task set. In each call, the procedure repairs the earliest failing load as follows. It down-scales the task voltage by the minimum amount such that the following conditions are met: (1) the task no longer fails, and (2) the deadlines are met (voltage task scaling increases the task delay). If condition (2) is violated the program control shifts to the previous task in the sequence and the same procedure is repeated. Minimum scaling is justified by Theorem 3 as it leads to delay slack being maximal for the later tasks. The above-mentioned process is repeated until all the failing loads are identified and the final Q at the end of the whole profile is positive. At this stage, the battery has survived the profile and now the slack utilization algorithm can be used to further increase the charge slack Q. Note that recovery insertion is not considered as an alternative to voltage scaling in keeping with Theorem 2.

Phase Two:Slack Utilization
This procedure is based on Theorem 3 which states that the greatest improvement in battery lifetime is obtained if the slack can be used as much as possible by the later tasks. The tasks are considered one by one, starting from the last tasks in the sequence. The last task is scaled to the lowest possible voltage in the set S_v subject to deadline constraints. The process is repeated until there is no slack available or none of the tasks can be assigned to a lower voltage.

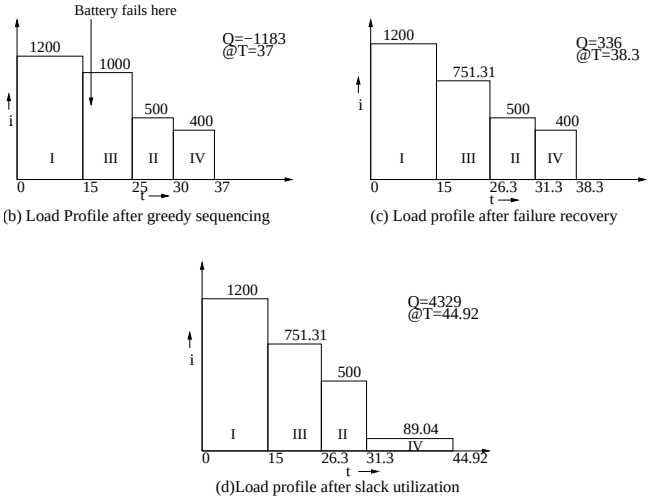
3.2. Examples

Example 1: To illustrate the execution of the algorithm in its entirety, we use the example shown in Figure 3. We assume that the operating voltages are selected from the set $S_V = \{3.3, 3.0, 2.7, 2.5, 2.0\}$ and the threshold voltage is assumed to be 0.4V. The battery parameters are $\alpha = 35220$ and $\beta = 0.637$(the parameters for a battery used in the Compaq Itsy Pocket Computer[12]). The initial task specification is described in Figure 3(a). Figure 3(b) displays the load profile after initial sequencing. The task loads are chosen (unusually) high to demonstrate the failure recovery mechanism. The battery fails during task III, and the quality factor Q is negative. Figure 3(c) displays the load profile after repairing task III. Note that its voltage has been down scaled from 3.3V to 3.0V. The length of the profile now is 38.3 minutes. Since the last task is not due until $t = 45$ minutes, the available slack can be utilized. The profile length increases to 44.92 minutes and the profile quality metric Q increases from 336mA-min to 4392mA-min(i.e more than 10X improvement) as shown in figure 3(d).

Example 2: Figure 4 describes an example illustrating the advantage of our approach. Figure 4(a) describes the initial task specification. Assuming a uniprocessor system, our algorithm generates the profile as given in figure 4(d). For comparison purposes, we also present two alternative profiles. The first alternative, shown in figure 4(b) levels out the voltages, a strategy used for minimizing energy consumption in ideal power sources [9]. The second alternative,

Task	Duration, min	Deadline, min	Current, mA
I	15	20	1200
II	5	35	500
III	10	28	1000
IV	7	45	400

(a) Initial task specifications



(b) Load Profile after greedy sequencing

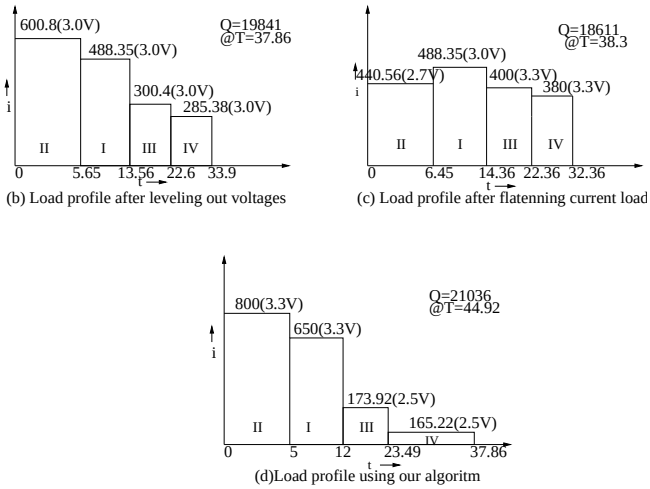
(c) Load profile after failure recovery

(d)Load profile after slack utilization

Fig. 3. Example of task scheduling for single processor system.

Task	Duration, min	Deadline, min	Current, min
I	7	18	650
II	5	10	800
III	8	26	400
IV	10	38	380

(a) Initial task specifications



(b) Load profile after leveling out voltages

(c) Load profile after flatenning current load

(d)Load profile using our algoritm

Fig. 4. Different approaches to slack utilization.

shown in figure 4(c) is generated with the goal of flattening out the discharge current profile(minimizing the peak current) [12]. It can be observed that our algorithm gives a higher value of the residual charge Q.

4. SCHEDULING ON A MULTIPROCESSOR SYSTEM

The system configuration for the SoC under consideration is described in figure 5. A single battery drives multiple processors and each processor has a dedicated DC-DC converter.

The charge that the battery has to deliver is proportional to the sum of the current loads of the processors. The cost function properties (Theorems 1,2,3) for single processor systems are applicable to the multi-processor configuration too. In addition, the following property also holds for a multi-processor set up:

The assignment of loads among the processors should be such that the resulting load profile is the steepest.

This is because the steepest load profile leads to the highest value of the quality factor Q.We illustrate this with the help of the following example.

Example: Consider a two processor system that has to execute 6 tasks with current values 100mA, 90mA, 80mA, 70mA, 60mA and 50mA. The possible assignments are:

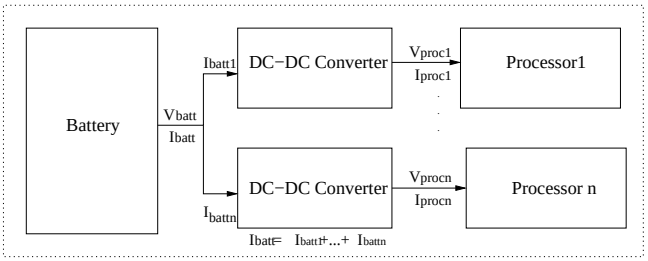


Fig. 5. System level configuration for a multiprocessor system.

Assignment 1: 100+90,80+70,60+50
Assignment 2: 100+80,90+70,60+50
Assignment 3: 100+70,90+80,60+50
Assignment 4: 100+60,80+70,90+50
Assignment 5: 100+50,90+60,80+70

This is illustrated in figure 6.

Distributing the load such that PE1 (processor1) is assigned tasks with loads 90, 70, 50 (mA) and PE2 (processor2) is assigned with loads 100, 80, 60 (mA), result in the largest residual battery charge Q. Such an assignment results in the current drawn from the battery being 190, 150 and 110 mA.

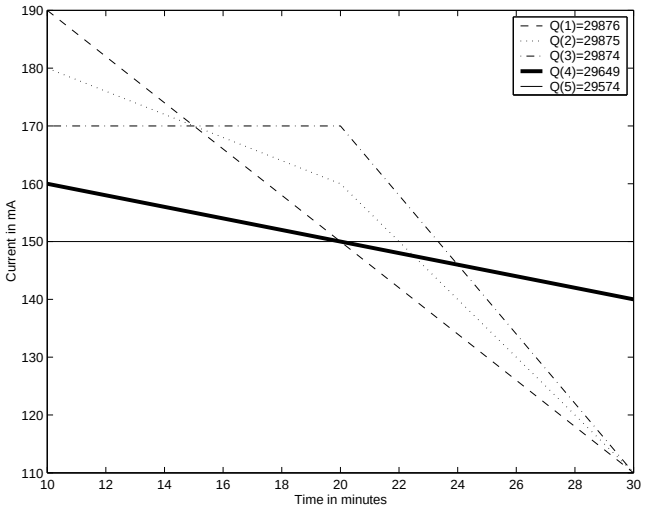


Fig. 6. Effect of current slope on the quality factor.

4.1. Scheduling Algorithm

Problem Definition: *Given the battery parameters α and β, the task graph describing the task profile and the available processing elements (PEs), allocate the tasks among the PEs such that all the deadline and precedence constraints are met and the residual charge slack Q at the end of the successful completion of a task set is maximized.*

Each task is specified by the same parameters as in the uniprocessor system. The output is the set S_t of the start time of each task after the scheduling has taken place, the new current values (if the voltage is scaled for that task), the assignment of the tasks to the different PEs and the profile quality metric $Q = \alpha\text{-}\sigma$.

Top Level View: The scheduling algorithm for the multi processor system is an extension of the algorithm for the single processor system. A *list based scheduling* technique with a *battery-aware priority function* is used to assign the tasks among the processing elements. The priority function ensures that the load profile has the steepest slope and that the deadline and precedence constraints are not violated. Initially all the tasks are assigned to the highest voltage level of the PE. Successful completion of all the tasks without use of voltage scaling marks the end of the first phase of the algorithm. In the second phase the algorithm applies voltage downscaling to completely utilize the available slack.

Algorithm Description

Phase one: Permissible schedule assuming no battery failure

At the start of this phase the task voltages are assigned to the highest voltage value. The proposed scheduling technique is *list based*, where the ready nodes are assigned to the processing elements according to a battery-aware priority function. The tasks are arranged in the ready set according to the deadline(first priority) and non-increasing order of currents(second priority). The list-based scheduler assigns the nodes from the ready set. The assignment is such that in each processor the tasks are arranged in a non-increasing order of load currents. Since the current load that the battery sees is the summation of the loads assigned to each processor and since each processor also has the current loads arranged in non-increasing order, this assignment results in the slope of the battery load current being the sharpest, thus conforming to the steepest slope heuristic. After all the tasks have been assigned, if there is slack available in the task set, the slack utilization procedure is called to enhance the Q value. It must be noted that in the processor assignment, inter processor communication is avoided as far as possible. This might lead to some processors being active most of the time and some others being relatively idle.

Phase two:Slack Utilization

This phase is based on Theorem 3 which suggests that the greatest improvement in the profile cost is obtained when the slack is used as much as possible by the later tasks. The tasks are considered one by one starting with the tasks with the latest finish time and scaled to the lowest possible voltage in the set S_v subject to deadline constraints. The process is repeated until there is no slack available or none of the tasks can be assigned to a lower voltage. This is essentially the same technique that is used for slack utilization in a single processor system and described in section 3.

4.2. Example

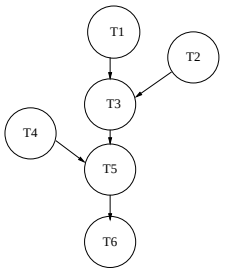
Figure 7 illustrates the algorithm in its entirety. Figure 7(a) describes the initial task specification. The task graph in figure 7(b) defines the precedence relation amongst the tasks. Figure 7(c) is the feasible schedule generated after phase one of the algorithm. Since the last task completes at $t = 27.01$ and its deadline is $t = 35$ the slack utilization algorithm can be applied. Figure 7(d) shows the new schedule after slack utilization. E1 and E2 in 7(c) and (d) refer to the inter-processor communication delay and cost which are assumed to be 0.01 minutes and 1mA respectively. The Q value for figure 7(c) is 26508 and is 28439 for figure 7(d) after slack adjustment. The final battery load profile is shown in 7(e).

5. CONCLUSION

In this paper we addressed the problem of aperiodic task scheduling for battery-operated single processor and multiprocessor systems. The objective throughout the paper has been to maximize the charge slack at the end of the task sequence by controlling the discharge current profile. The proposed heuristic based scheduling algorithm utilizes the battery properties to guide scheduling and task assignment.

Task	Duration, min	Deadline, min	Current, mA
1	5	7	300
2	7	10	350
3	3	22	300
4	10	19	250
5	4	28	100
6	8	35	80

(a) Initial task specifications



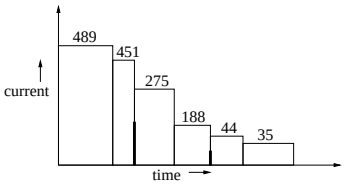
(b) Input Task Graph

PE1	T2,350mA 0−7 mins			T3,200mA 7.01−10.01 mins							
Bus			E1				E2				
PE2	T1,300mA 0−5mins			T4,250mA 5−15mins				T5,100mA 15.01−19.01 mins		T6,80mA 19.01−27.01mins	

(c)Processor assignment after Phase one

PE1	T2,263mA 0−7.91mins 3.0V			T3,87mA 7.92−12.23 mins 2.5V							
Bus			E1				E2				
PE2	T1,226mA 0−5.65mins 3.0V			T4,188mA 5.65−16.95mins 3.0V				T5,44mA 16.96−22.72 mins 2.5V		T6,35mA 22.72−34.22mins 2.5V	

(d)Processor assignment after Phase two



(e)Battery Load Profile after scheduling

Fig. 7. Example of task scheduling in a multiprocessor system.

The next step is to implement these algorithms on a Strong Arm platform.

Acknowledgements This work was carried out at the National Science Foundation's State/Industry/University Cooperative Research Centers' (NSF-S/IUCRC) Center for Low Power Electronics (CLPE). CLPE is supported by the NSF (Grant EEC-9523338), the State of Arizona, and a consortium of companies from the microelectronics industry .

6. REFERENCES

[1] D. Linden, *Handbook of Batteries*, McGraw-Hill, New York, 1995.

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

[3] M. Pedram and Q. Wu, "Design considerations for battery-powered electronics," *Proc. DAC*, 1999.

[4] C. Schurgers, O. Aberthorne, and M.B. Srivastava, "Modulation scaling for energy aware communication systems," *Proc. ISLPED*, 2001.

[5] C. Chiassereni and R.R. Rao, "Improving battery performance by using traffic shaping techniques," *IEEE Journal on Selected Areas in Communication*, vol. 19, July 2001.

[6] F. Yao, A. Demers, and S. Shankar, "A scheduling model for reduced CPU energy," *IEEE Found. Comp. Science*, 1995.

[7] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," *Proc. ISLPED*, 1998.

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

[9] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling algorithms for minimizing energy," *Proc. ISLPED*, 2001.

[10] L. Benini, G. Castelli, A. Macii, and R. Scarsi, "Battery-driven dynamic power management," *IEEE Design and Test*, March-April 2001.

[11] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "A discrete-time battery model for high-level power estimation," *Proc. DATE*, 2000.

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

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

[14] D. A. Wallach, "Interpreting the battery lifetime of the Itsy version 2.4," *Compaq WRL Technical Note TN-61*, December 2001.

[15] D. Rakhmatov, S. Vrudhula, and C. Chakrabarti, "Battery-conscious task sequencing for portable devices including voltage/clock scaling," *Proc.DAC*, 2002.

[16] D. Rakhmatov and S. Vrudhula, "Battery lifetime prediction for energy-aware computing," *Proc. ISLPED*, 2002.

[17] M.Doyle, T.Fuller, and J.Newman, "Modeling of the galvanostatic charge and discharge of the lithium/polymer/insertion cell," *J. Electrochemical Society*, 140(6) 1993.

[18] T.Fuller, M.Doyle, and J.Newman, "Simulation and optimization of the dual lithium ion insertion cell," *J. Electrochemical Society*, 141(1) 1994.