Static Task-Scheduling Algorithms for Battery-Powered DVS Systems

Princey Chowdhury and Chaitali Chakrabarti

***Abstract*—Battery lifetime enhancement is a critical design parameter for mobile computing devices. Maximizing the battery lifetime is a particularly difficult problem due to the nonlinearity of the battery behavior and its dependence on the characteristics of the discharge profile. In this paper, we address the problem of task scheduling with voltage scaling in a battery-powered single and multiprocessor system such that the residual charge or the battery voltage (the parameters for evaluating battery performance) is maximized. We propose an efficient heuristic algorithm using a charge-based cost function derived from the analytical battery model. Our algorithm first creates a task sequence that ensures battery survival, and then distributes the available delay slack so that the cost function is maximized. The effectiveness of the algorithm has been verified using DUALFOIL, a low-level Li-ion battery simulator. The algorithm has been validated on synthetic examples created from applications running on Compaq's handheld computing research platform, ITSY.**

***Index Terms*—Battery optimizations, DVS processors, low power, scheduling, voltage scaling.**

I. Introduction

BATTERY-OPERATED portable devices are widely used in mobile computing and wireless communication applications. Maximizing the battery lifetime is the most important design metric in such systems. This problem is quite challenging due to the nonlinear 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 during task scheduling of a dynamic voltage scalable (DVS) processor such that the residual charge or the battery voltage (the parameters for evaluating battery performance) at the end of a task profile is maximized.

Task scheduling for DVS processors has been studied extensively in recent years [2]–[6]. The algorithms can be broadly classified into static (or off-line) scheduling algorithms where the task parameters (arrival times, deadline times, execution times) are known *a priori*, and dynamic (or on-line) scheduling algorithms where all the task parameters are not known until execution time. Both classes of algorithms assume 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 nonlinear discharge behavior that need to be exploited.

In recent years, there has been significant amount of work done in studying battery characteristics [1], [7]–[12] and using these characteristics to shape the discharge profile [9], [13]–[15]. The method proposed in [13] combines Peukert's model [16] with those in [17] to design a schedule where the load profile has a reduced variance. The method in [14] is based on the use of the analytical model in [12] and uses a combination of recovery insertion and voltage scaling.

In this paper we address the problem of static task scheduling for battery-operated systems which is also based on shaping the discharge profile. We propose an efficient heuristic method using a charge-based cost function derived from the accurate analytical battery model in [12]. First, our algorithm creates a task sequence that ensures battery survival[1], and second, it distributes the available delay slack so that the residual charge is maximized. This is an extension of our earlier work presented in [18]. The proposed sheduling algorithm can be used in applications where predictability is of utmost importance (example, aircraft controller, automotive controller, robot arm controller, etc.) or in real-time scheduling where static scheduling is a first step that is done during compile time (example, scheduling in sensor cluster-heads, PDAs, cellphones, etc.). The main contributions of this work are:

- developing several key properties of the cost function with respect to voltage scaling in task scheduling;
- validating these properties using DUALFOIL [19], a low-level electrochemical simulator;
- utilizing these properties to develop efficient heuristics to guide scheduling of tasks with deadlines and dependencies;
- application of the task-scheduling algorithm to aperiodic and periodic task sets in single-processor as well as multiprocessor system configuration;
- demonstrating the superiority of this method by running simulations on synthetic examples created with tasks having time-varying profiles obtained from ITSY, Compaq's handheld computing research platform.

The rest of this paper is organized as follows. The paper begins with two motivational examples in Section II. Section III describes the battery models and other preliminaries like task definition, voltage scaling and the impact of voltage scaling on the battery current. The basis for the proposed heuristics is explained in Section IV. The proposed battery-aware

Manuscript received April 4, 2003; revised August 24, 2004. This work was supported in part by the NSF/SIUCRC Center for Low Power Electronics and by the NSF/IUCRC Connection One.

P. Chowdhury is with Maxim Integrated Products, Sunnyvale, CA 94086 USA (e-mail: princeyc@design.mxim.com).

C. Chakrabarti is with the Department of Electrical Engineering, Arizona State University, Tempe, AZ 85287-5706 USA (e-mail: chaitali@asu.edu).

Digital Object Identifier 10.1109/TVLSI.2004.840771

[1]If the battery does not survive the discharge profile, then some tasks are not completed.

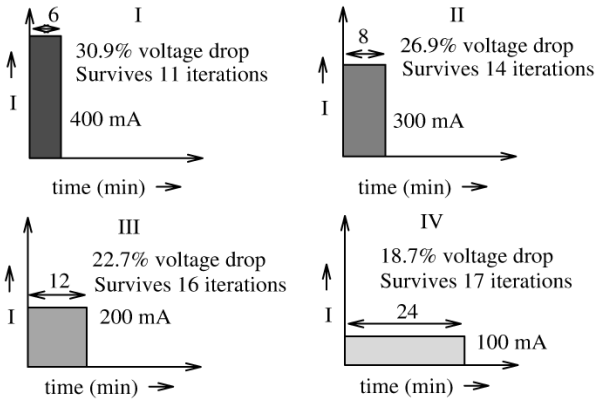


Fig. 1. Example 1: Same energy but different battery performance.

Task	Duration(min)	Deadline(min)	Current(mA)
I	7	25	250
II	5	32	300
III	8	26	100
IV	10	38	75

(a) Initial Task Specification

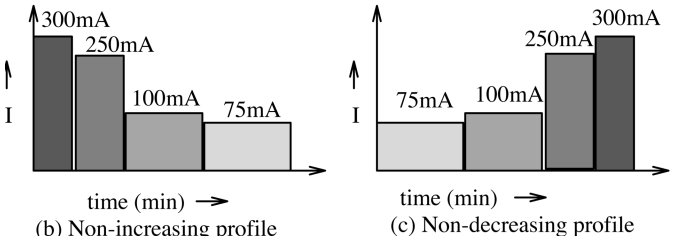


Fig. 2. Example 2: Nonincreasing and nondecreasing load sequences for the same task set. (a) Initial task specification. (b) Nonincreasing profile. (c) Nondecreasing profile.

task-scheduling technique for aperiodic and periodic tasks running on single-processor systems is described in Section V. Task scheduling for multiprocessor systems is described in Section VI. Experimental results are listed in Section VII. The paper is concluded in Section VIII.

II. Motivation

To motivate the need for battery-aware system design, we present two examples. The first example (see Fig. 1) describes a scenario where four algorithms (Cases I–IV) have equal energy consumption ($E = I * V * t$) but different discharge current profiles. Simulations with DUALFOIL show voltage drops of 30.9% (for Case I), 26.9% (for Case II), 22.7% (for Case III), and 18.7% (for Case IV). If the current profile is repeated, then the number of iterations that each profile supports before the battery goes dead is 11 (for Case I), 14 (for Case II), 16 (for Case III) and 17 (for Case IV). Thus Case IV which corresponds to the lowest load current has the lowest voltage drop and supports the largest number of iterations, implying that it has the best battery performance. This simple example shows that battery performance is dependent on the discharge characteristics of an application rather than its energy consumption.

The second example (refer Fig. 2) demonstrates the importance of task sequencing on the battery performance. Assume that there are four tasks with arrival time $t = 0$ described in Fig. 2(a). For each task, the duration, current and deadline is given. Fig. 2(b) and (c) show a nonincreasing profile and a nondecreasing profile for the same task set. From Table I we see that the nonincreasing profile does significantly better—a 13.8% improvement in the residual charge and a 9.1% improvement in the battery voltage for a task set duration of only 30 min. Thus the task-scheduling algorithm should generate load profiles that enhance battery performance.

TABLE I
Performance of the Nonincreasing and Nondecreasing Profile Using the Dualfoil Battery

Profile	Analytical Q (mA-min)	Voltage	% drop in voltage
Non-Increasing	31724	4.04	23.6
Non-decreasing	26145	3.94	32.7

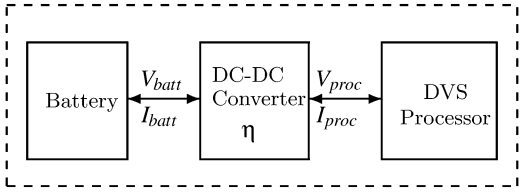


Fig. 3. System-level configuration.

III. Background

A. System Configuration

The system configuration for the battery-operated single-processor device under consideration is described in Fig. 3. The system consists of one DVS (dynamic voltage scalable) processor driven by a single battery. The battery is used to power the processor through a dc–dc converter which is essential for voltage shifting and stabilization. The dc–dc converter has an efficiency η which is typically in the range [0.8, 0.9] [7]. The efficiency $\eta = (I_{\rm proc}V_{\rm proc})/(I_{\rm batt}V_{\rm batt})$, where $V_{\rm batt}$ and $I_{\rm batt}$ are the battery voltage and current, and $V_{\rm proc}$ and $I_{\rm proc}$ are the processor voltage and current.

The DVS processor can change the voltage and speed of the processor according to system requirements. Examples include StrongARM, Intel's X-scale processor, and Transmeta's Crusoe processor. These processors utilize the available slack by downscaling the voltage resulting in significant reduction in energy consumption. Note that in this 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, supply voltage V_{dd}, and velocity saturation index $\gamma = 2$, the delay is proportional to $V_{dd}/[V_{dd} - V_t]^2$. Let Δ denote the initial task duration, and let Δ^* denote the new task duration after scaling the task voltage V_{dd} down by the factor of s (i.e., the new task voltage is $V_{\rm proc} = V_{dd}/s$). Using the CMOS gate voltage–delay relationship, we obtain

$$\frac{\Delta^*}{\Delta} = s\left[1 + \frac{2(s-1)V_t}{V_{dd} - V_t}\right]. \quad (1)$$

When devices are in saturation, voltage scaling by a factor of s causes the processor current $I_{\rm proc}$ to scale by $s^2[1-2V_t/V_{dd}][1+2sV_t/V_{dd}]$. For $V_t = 0.4$ V and $V_t \ll V_{\rm proc}$,

this can be approximated to s^2. This approximation has been validated by current–voltage values provided by StrongARM. Assume that the dc–dc conversion efficiency η and the battery voltage $V_{\rm batt}$ are averaged constants for the duration of a task execution. Then, the battery current $I_{\rm batt}$ scales by ηs^3. For short channel devices, $I_{\rm batt}$ scaling is projected to be between ηs^2 and ηs^3. Thus for both short and long channel devices, 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 as will be apparent in the subsequent sections.

B. Battery Models

In this work, we have considered two battery models: 1) a high-level analytical charge-based model [12] and 2) a low-level electrochemical simulator, DUALFOIL [19]. While [12] is sensitive to charge, [19] is sensitive to voltage. Lack of availability of a single model that handles both charge and voltage prompted us to consider both these models in our experiments.

1) High-Level Analytical Model: The analytical charge based model is based on the charge transport of the electrolyte and is very accurate for time-varying loads as compared to the Peukert's model [16]. It gives an analytical relationship between the current load, discharge time and the corresponding charge slack at that discharge time. Under a time-varying discharge $i(t)$, the battery model is of the following form [12]:

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

where L is the lifetime and α and β are battery-specific parameters. For a constant discharge current $(i(t) = I)$, (2) reduces to:

$$\alpha = I\left[L + 2\sum_{m=1}^{\infty}\frac{1-e^{-\beta^2 m^2 L}}{\beta^2 m^2}\right]. \quad (3)$$

Note that α is a constant that represents the the total capacity of the battery and is expressed in charge units (coulombs). β^2 is related to the diffusion rate within the battery and captures its nonlinear discharge profile. A small value of β implies that the load on the battery has to be reduced (possibly to zero) for the battery to *recover*. A large value of β means a much better battery; if β is sufficiently large, the second term in (3) becomes negligible, and we obtain the model of an ideal power source. The magnitude of the series terms in (3) diminish very rapidly as m grows. Experiments indicate that employing only the first 10 terms is sufficient for accurate lifetime predictions [12].

2) DUALFOIL: For validation purposes, all our experiments have been verified by a low-level battery simulator DUALFOIL. DUALFOIL numerically simulates a set of partial differential equations governing the behavior of a rechargeable Li-ion cell [19]. For any input task profile, DUALFOIL gives very accurate output in terms of battery voltage as a function of the discharge time.

3) Battery Configurations Used: Two batteries have been considered throughout this paper.

- B1: This is a 2.2 Whr Li-ion battery used in Compaq's ITSY pocket computer. It has an open circuit voltage of 4.1 V and nominal discharge rate of 640 mA. α and β values for B1 have been estimated as 35 220 mA-min and 0.637 $\text{min}^{-1/2}$, respectively [20].
- B2: This is the Li-ion battery used in DUALFOIL. It has an open circuit voltage $V_{\rm oc} = 4.3$ V and the cutoff voltage $V_{\rm cut} = 3.2$ V. The parameters α and β for B2 have been estimated to be 40 375 mA-min and 0.273 $\text{min}^{-1/2}$, respectively [20].

The low value of β for the battery B2 suggests that B2 is highly nonlinear compared to B1. Note that while analytical (charge-based) models are available for both B1 and B2, DUALFOIL is only applicable to B2. Thus in evaluating the performance of the algorithms, we consider the residual charge for both B1 and B2, and the battery voltage or drop from the open-circuit voltage with respect to the full battery swing, defined as $(V_{\rm oc} - V_{\rm batt})/(V_{\rm oc} - V_{\rm cut})$, for B2. Higher the residual charge, better is the battery performance. In the same note, lower the voltage drop, better is the battery performance. Since the voltage range of B2 over its complete battery life is 1.1 V $(= 4.3 - 3.2)$, any improvement due to application of battery-aware task scheduling for a few minutes will be of the order of fraction of a volt.

C. Related Work

Battery-driven system design has been an area of interest in recent years. The work can be categorized into: 1) battery modeling and metrics to capture its nonlinear behavior [1]; 2) power management [7], [21], [22]; and 3) task scheduling of battery-operated devices [13]–[15].

There are two types of battery models: low-level and high-level. The high-level models include a PSPICE equivalent circuit [23], a discrete time VHDL model [7] and a stochastic model [9]. The model in [17] is based on Peukert's law and introduces the efficiency factor to account for charge nonlinearity. The stochastic model in [9] is represented as a Markov chain of battery states with forward transitions corresponding to conditions of discharge and backward transitions corresponding to recovery. The analytical model reported in [24] considers special cases of discharge process (diffusion limited, reaction-limited, and ohmically limited). The charge-based analytical model in [12] gives a very accurate estimate of the battery lifetime given the conditions of discharge.

The nonideal properties of a battery have resulted in a different set of metrics. Martin and Sewiorek [1] showed that peak power predicts the state of the battery better than the average power. Thus, decreasing a mobile computer's active power will increase the battery life more than decreasing its idle power, even if both reduce the average power by the same amount. Pedram and Wu [17] use Peukert's model to show that the battery efficiency decreases as the average discharge current from the battery increases. While the model is accurate for only constant discharge currents, the conclusion that more work can be extracted out of the battery if the load current is smaller, is still valid.

Benini *et al.* studied battery-aware dynamic power management policies in [7] and [21]. The policies were based on time-out open loop (battery voltage independent) and threshold-based closed loop (battery voltage dependent). Park and Srivastava in [25] introduced novel static and dynamic

battery state aware aproaches to improve battery utilization. The key distinction of this work is that is that they look at the total amount of work done (or service) as the main metric instead of just the lifetime. The stochastic battery model of [9] has been applied to bursty network traffic with the aim of reshaping the traffic to enhance battery performance [8]. The stochastic model has also been used for battery life estimation during hardware/software embedded system exploration [10].

There are very few approaches for battery-aware task scheduling. Luo and Jha [13] considered static scheduling of tasks with deadline constraints in a multiprocessor environment. The method is based on a combination of Peukert's model with those in [17] to design a schedule where the load profile has a reduced variance and average power consumption. Rakhmatov *et al.* considered aperiodic task scheduling for single-processor systems based on the analytical model of [12] in [14] and [15]. Our work is an extension of [14] and [15]; it is different in that both aperiodic and periodic tasks are considered, the computing platform could be single or multiprocessor, voltage scaling is used exclusively (instead of recovery insertion followed by voltage scaling) and time-varying loads are considered.

IV. Basis for Task-Scheduling Heuristics

A. Task Definition

A given task k is associated with four parameters: the current I_k, the duration Δ_k, the start time t_k and the deadline d_k. The current I_k can vary across the duration of a task as in most real time applications (e.g., MPEG). In that case, the current I_k is approximated by a staircase function with current $I_{k,l}$ being constant for duration $\Delta_{k,l}$ for $l \in [0, n_k]$. If n_k is the number of steps to represent task k, then $\Delta_k = \sum_{l=0}^{n_k} \Delta_{k,l}$. For the sake of simplicity, in many of the examples, the current loads have been assumed to be constant for the duration of the task. However, in the experiments described in Section VII, time-varying loads have been considered.

B. Cost Function

Assume that the discharge profile consists of n scheduled tasks. The profile length is $T = t_{n-1} + \Delta_{n-1}$, which is the finish time of the last task in the sequence. Let

$$\sigma = \sum_{k=0}^{n-1} \sum_{l=0}^{n_k-1} I_{k,l} F(T, t_{k,l}, t_{k,l} + \Delta_{k,l}, \beta) \tag{4}$$

where

$$F(T, t_{k,l}, t_{k,l} + \Delta_{k,l}, \beta) = \Delta_{k,l} + 2 \sum_{m=1}^{\infty} \frac{e^{-\beta^2 m^2 (T - t_{k,l} - \Delta_{k,l})} - e^{-\beta^2 m^2 (T - t_{k,l})}}{\beta^2 m^2}.$$

Intuitively, σ is the charge that the battery has lost in time T. The profile quality metric defined as $Q = \alpha - \sigma$ is the cost function to be *maximized*. Intuitively, Q is the residual charge (the amount of charge less than the capacity α). Note that a negative Q at the end of a profile indicates battery failure.

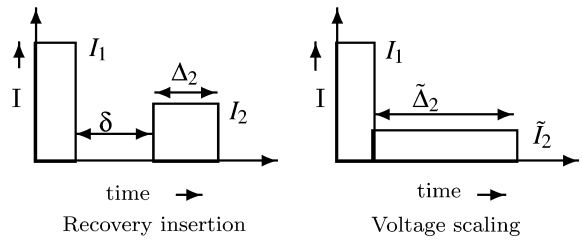


Fig. 4. Example 3: Recovery insertion versus voltage scaling.

C. Cost Function Properties

There are several important properties of the cost function σ that have been presented in [18], [20] and analytically derived in [20]. These properties are summarized below and illustrated using examples.

Property 1: For a fixed voltage assignment (only task start times can be changed), sequencing tasks in the nonincreasing order of their currents is optimal when the task loads are constant during the execution of the task.

The effectiveness of Property 1 has been illustrated in Example 2 in Section II. Sequencing tasks in nonincreasing order resulted in a 23.6% drop in the battery voltage compared to the 32.7% drop for the nondecreasing order.

For a load current that varies during the execution of a task, Property 1 has to be applied with caution. There are several parameters that can affect the conditions of discharge. These include average load current (I_{av}), load current times duration $(\sum i * t)$, variation of the load current with respect to I_{av}, peak current (I_{peak}), and peak current times its duration. Extensive experimentation revealed the following: 1) when I_{av} are equal, the load with higher $\sum i * t$ has higher priority; 2) when $\sum i * t$ are equal, the load with higher I_{av} has higher priority; 3) when both I_{av} and $\sum i * t$ are different, the load with higher I_{av} has higher priority; 4) when I_{peak} are same, the load with higher I_{av} has higher priority; 5) when I_{av} are same, the load with higher I_{peak} has higher priority; 6) when both I_{av} and I_{peak} are different, the load with higher I_{av} has higher priority; and 7) when both I_{peak} and the corresponding durations are different, the load with higher peak current times duration has higher priority. These priorities were used in our heuristics when handling time-varying loads.

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

Example 3 illustrates the fact that voltage downscaling always outperforms rest period insertion. Fig. 4 describes a 2-task profile once when a recovery period of δ is inserted between the two tasks, and once when time δ is used to scale the voltage of the second task. Here $I_1 = 200$ mA, $I_2 = 150$ mA, $\Delta_1 = 5$ min, $\Delta_2 = 2$ min and recovery $\delta = 1.9$ min. The residual charge for B1 and B2 and the voltage at the end of the profile for B2 are listed in Table II. It is clearly seen that voltage scaling does better than recovery insertion. Thus, anytime there is a battery failure, it is always better to downscale the voltage (and consequently, the current) of the failing load rather than insert a recovery period.

TABLE II
PERFORMANCE OF RECOVERY INSERTION AS OPPOSED TO VOLTAGE SCALING

Profile	Analytical Q (mA-min)		Voltage (B2)	% drop in voltage
	B1	B2		
Recovery Insertion	32933	35384	4.13V	15.4
Scaling	35384	36985	4.19V	10.0

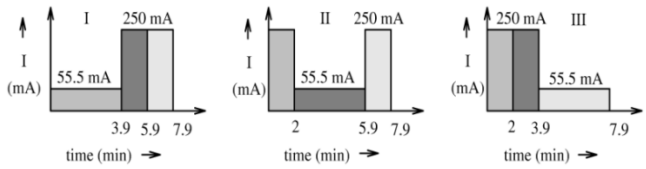


Fig. 5. Example 4: Effect of slack utilization on later slack.

TABLE III
EFFECT OF SLACK UTILIZATION ON LATER TASK

Case	Analytical Q (mA-min)		Voltage (B2)	% drop in voltage
	B1	B2		
I	32294	33937	4.095V	18.6
II	32492	34452	4.099V	18.2
III	33432	36380	4.174V	11.4

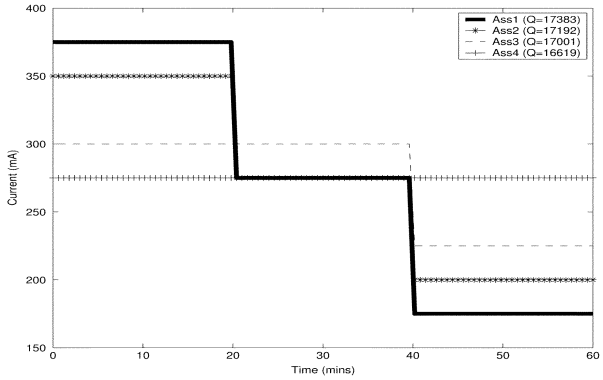


Fig. 6. Example 5: Effect of current slope on the quality factor.

TABLE IV
RESULTS FOR THE MULTIPROCESSOR LOAD ASSIGNMENTS

Assignment	Analytical Q (mA-min)		Voltage (B2)	% drop in voltage
	(B1)	(B2)		
1	17383	15970	3.75V	50.0
2	17192	15908	3.73V	51.8
3	17001	15787	3.71V	53.6
4	16619	12523	3.67V	57.2

Property 3: Given a pair of two identical tasks in the profile and a delay slack to be utilized by voltage downscaling, it is better to use the slack on the later task than on earlier task.

Example 4: Consider three identical tasks where each task has a load current of 150 mA, execution delay of 2 min and the slack of 1.9 min (see Fig. 5). This slack can be used either on Task 1 or Task 2 or on Task 3 as described by Case I, Case II, and Case III in Fig. 5. Table III gives the residual charge and voltage values for each case. Note that for both the battery configurations, Case III does best. This indicates that it is better to schedule tasks as early as possible and push all the recovery to the end.

Property 4: In a multiprocessor system, the assignment of loads among the processors should be such that the resulting load profile is the steepest.

In such a system, the charge that the battery has to deliver at any time instant is proportional to the sum of the current loads of all the active processors at that instant. Experiments show that the steeper the current slope (I_{batt} versus time), the better the battery performance. This is illustrated with the help of the following example.

Example 5: Consider a two processor system that has to execute six tasks with current values 200, 175, 150, 125, 100, and 75 mA each for a period of 20 min. Possible nonincreasing assignments are as follows.

Assignment 1: $200 + 175, 150 + 125, 100 + 75$.
Assignment 2: $200 + 150, 175 + 100, 125 + 75$.
Assignment 3: $200 + 125, 175 + 150, 100 + 75$.
Assignment 4: $200 + 75, 175 + 100, 150 + 125$.

These assignments are graphically demonstrated in Fig. 6. Distributing the load such that Processor 1 is assigned tasks with loads 200, 150, and 100 (mA) and Processor 2 is assigned with loads 175, 125, 75 (mA) result in the largest residual battery charge Q as seen in Fig. 6. The results for assignments 1–4 are listed in Table IV. It can be observed that the load with the steepest profile (Assignment 1) gives the best performance in terms of charge as well as voltage.

V. TASK SCHEDULING FOR A SINGLE PROCESSOR SYSTEM

A. Problem Definition

Given the battery parameters α and β, the task dependencies and the task specifications (arrival, execution and deadline times), and the set of supported discrete voltages, schedule the tasks such that the deadline and precedence constraints are met and the profile quality metric Q is maximized. We consider both aperiodic as well as periodic task sets.

1) Proposed Algorithm: The proposed scheduling algorithm operates in two phases.

Phase I: Obtain feasible schedule by: 1) using the earliest deadline first (EDF) algorithm; 2) trying to generate a nonincreasing order of loads; and 3) ensuring that there is no failure during the battery discharge. In case of failure, downscale the voltage of a failing task by the *minimum* amount.

Phase II: Utilize the available slack by voltage down scaling as much as possible starting from the end of the profile.

The top-level view of the algorithm is shown in Fig. 7. We first describe the algorithm for aperiodic tasks and then point out the extensions for periodic tasks.

Phase I: Generating a Feasible Schedule

At the start of this phase, all the task voltages are assigned to V_0 (the highest voltage value). First, the tasks are arranged according to the earliest deadline first (EDF) policy. Then, a greedy approach is used to reschedule them so that a nonincreasing order of task currents is obtained, if possible. This step is justified by Property 1. After the task order is finalized, the algorithm checks whether $Q \geq 0$, i.e., the battery survives the

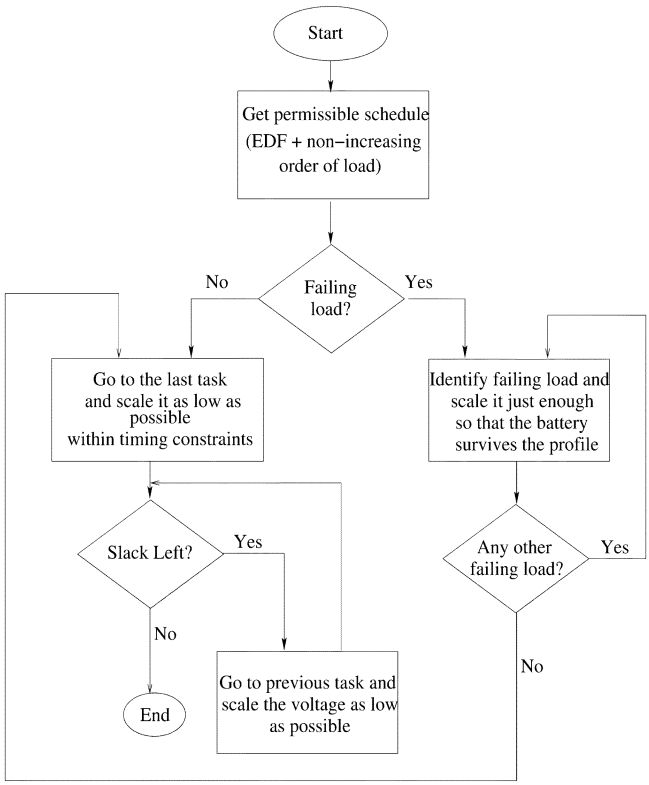


Fig. 7. Top-level view of the algorithm for aperiodic tasks for a single-processor system.

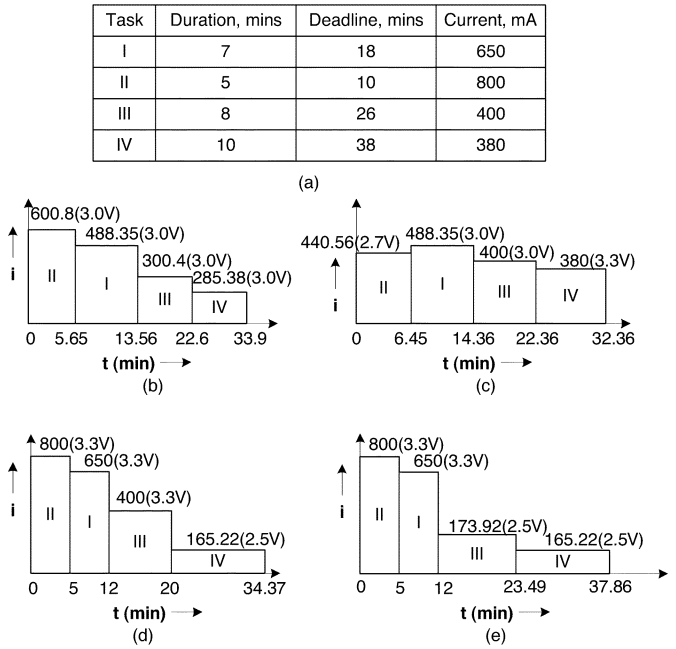


Fig. 8. Example 6: Load profiles after each step of the proposed aperiodic task algorithm. (a) Initial task specifications. (b) Load profile after greedy sequencing. (c) Load profile after failure recovery. (d) Load profile after slack utilization.

resulting discharge profile. If a failure is detected for some task q (i.e., Q is negative), it calls the failure recovery procedure.

Failure Recovery: The failure recovery procedure reshapes the profile so that the battery survives all the tasks. Task ordering is not changed; however, voltages for some tasks are scaled down. In each call, the procedure repairs the earliest failing task as follows. It downscales the task voltage by the *minimum* amount such that the following two conditions are satisfied: 1) the task no longer fails, and 2) the deadlines are met (voltage downscaling increases the task duration). If condition 2 is violated, the program control shifts to the previous task in the sequence, and the same procedure is repeated until condition 1 is satisfied for the failing task in question. If the program control moves all the way to the first task without successful repair of the battery failure, then the given the task set is infeasible in terms of the battery charge. The rationale behind downscaling the voltage as little as possible is that it leads to a delay slack (available at the end of this step) being as much as possible. Failure recovery process is repeated until all the failing tasks are repaired. Unlike [14] we do not consider rest period insertion as an alternative to the voltage downscaling, since according to Property 2, the latter always outperforms the former.

Phase II: Slack Distribution

This phase is based on Property 3, which suggests that the profile cost is reduced when the slack is used as much as possible by late tasks. The tasks are considered one by one, starting from the last task in the sequence. The last task is scaled to the lowest possible voltage 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.

Energy overhead: The energy overhead due to voltage scaling if the charge at the output capacitance of the dc–dc converter is pumped back into the input capacitance is given by $E = (1-\eta)C_{\text{bypass}}(V^2 - V_{\text{new}}^2)$ where V is the highest operating voltage, V_{new} is the voltage of the task after scaling, η is the efficiency of the dc–dc converter and C_{bypass} is the capacitance of the bypass capacitor at the output node of the converter. η values typically range from 0.8 to 0.9 and C_{bypass} values typically range from 10 to 200 μF. Note that if the processor sinks the charge, then the overhead is a lot larger and is given by $E = \eta C_{\text{bypass}}(V^2 - V_{\text{new}}^2)$.

2) Illustrative Example (Independent Task Set): To illustrate execution of the algorithm in its entirety, we use Example 6 shown in Fig. 8. Assume that the operating voltages are selected from the set $S_V = \{3.3, 3.0, 2.7, 2.5, 2.0\}$, consistent with the voltage range of the StrongArm SA1100 DVS processor. The time overhead due to voltage scaling is not considered since it is usually a few microseconds [26]. The task loads in Example 6 are chosen (unusually) high to demonstrate the failure recovery mechanism. We assume that all the tasks are ready at time $t = 0$ for this example. Fig. 8(b) displays the load profile after the initial sequencing. The battery fails during task III, and the quality factor Q is negative. Fig. 8(c) displays the load profile after repairing task III: its voltage is scaled from 3.3 V to 3.0 V. The battery no longer fails, and all the tasks can be completed. The length of this profile is 38.3 min; however, the last task IV is not due until $t = 45$ min. After utilization of the available slack, the profile length increases to 44.92 min [see Fig. 8(d)], and the quality factor Q increases from 336 mA-min to 4329 mA-min (i.e., more than 10X improvement). The energy overhead if the dc–dc converter sinks the charge is $[78.3 - 1566]$ μJ for $\eta = 0.9$ and $C_{\text{bypass}} = [10\,200$ μF$]$.

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

(a)

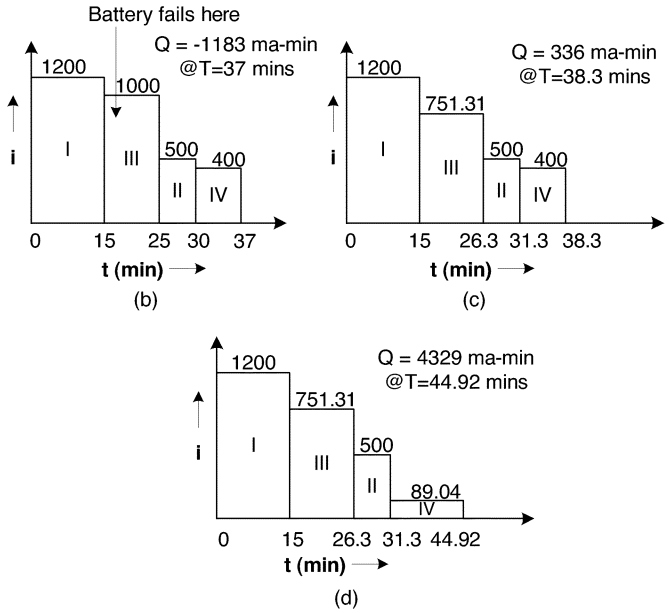


Fig. 9. Example 7: Different approaches to slack utilization for aperiodic tasks. (a) Initial task specifications. (b) Load profile after leveling out voltages. (c) Load profile after flattening current load. (d) Load profile using battery-aware algorithm. (e) Load profile using our algorithm.

The algorithm proposed in [14] does not fare well for this example. The battery repair using recovery insertion is unsuccessful for this task set as the amount of recovery period needed is greater than 45 min which is the deadline for the last task in the sequence.

3) Comparative Example: Fig. 9 describes Example 7 illustrating the advantage of the proposed slack utilization approach. The initial task specification is given in Fig. 9(a). Our algorithm generates the final profile as shown in Fig. 9(d). For comparison purposes, three alternative profiles are presented. The first alternative, shown in Fig. 9(b), is generated with the goal of levelling out the task voltages (i.e., all the tasks are assigned to the same voltage). This is a strategy used for minimizing energy consumption of a system powered by an ideal power source [4]. The second alternative, shown in Fig. 9(c), is generated with the goal of minimizing the peak current of the profile (i.e., the profile is flattened) [13]. The third alternative, referred to as battery-aware, first assigns tasks to the lowest available voltage and then upscales the voltage starting from the first task (so that the profile is a decreasing one) to meet the timing constraints [14].

Table V describes the results for this example using the analytical model as well as DUALFOIL. The proposed approach is superior both in terms of residual charge as well as voltage. For instance the drop in voltage is only 41.8% compared to 50.9% for the level voltage method, 59.1% for the level current method and 46.4% for the battery aware approach. If the dc–dc converter

TABLE V
PERFORMANCE COMPARISON OF THE VARIOUS SLACK UTILIZATION SCHEMES FOR APERIODIC TASKS

Algorithm	Analytical Q (mA-min)		Voltage	% drop
	B1	B2	B2	in voltage
Level Voltage	19841	14942	3.74V	50.9
Level Current	18611	11229	3.65V	59.1
Battery-aware	19826	17101	3.79V	46.4
Our Approach	**21036**	**19569**	**3.84V**	**41.8**

sinks the charge and $\eta = 0.9, C_{\text{bypass}} = [10\text{–}200]\ \mu$F, the energy overhead for the level voltage approach is [68–1360.8] μJ, for the level current approach is [49.41–988.3] μJ, for the battery-aware approach is [41.76–835.2] μJ, and for our approach is [83.52–1764] μJ.

B. Scheduling for Periodic Tasks

A feasible schedule for periodic tasks exists if and only if there exists a feasible schedule for its *hyperperiod* [27], where the hyperperiod is defined as the *lowest common multiple* of the periods of all the task graphs in the task set. The scheduling algorithm for periodic tasks is based on application of battery-aware scheduling techniques (developed for aperiodic tasks) within the hyperperiod.

Phase I: Generating a feasible schedule

A best nonincreasing schedule within a hyperperiod that meets all timing constraints is created. Failure detection and recovery is done at this stage in the same way as is done for the hard aperiodic tasks. However, the failing tasks need to identified and repaired within a hyperperiod.

Phase II: Slack distribution

The available slack is distributed starting from the last task in the hyperperiod as described in Section V-A.1.

1) Comparative Example: Fig. 10 describes an example illustrating the advantage of our approach for a periodic task set. There are three tasks with periods 2, 4 and 6 min, respectively; the hyperperiod is 12 min (L.C.M of 2, 4 and 6). The initial task specification is described in Fig. 10(a). The slack utilization algorithm generates the final profile as shown in Fig. 10(b). For comparison purposes, an alternative profile generated with the intention of levelling out the current and reducing the peak current [13] is shown in Fig. 10(c). For each of the two cases, the corresponding quality factor Q using the analytical model and the battery voltage at the end of the discharge using DUALFOIL are displayed in Table VI. One can see that our approach is superior in terms of charge as well as voltage.

VI. TASK SCHEDULING FOR A MULTIPROCESSOR SYSTEM

A. Multiprocessor System Configuration

The system configuration for the DVS based multiprocessor system under consideration is described in Fig. 11. A single battery drives multiple processors; each processor has a dedicated dc–dc converter connected to it. The terms $V_{\text{batt}}, I_{\text{batt}}$ in Fig. 11 represent the battery voltage and battery current. $V_{\text{proc}}(i)$ and $I_{\text{proc}}(i)$ represent the operating voltage and current of processor (i); $I_{\text{batt}}(i)$ refers to the current drawn from the battery by processor (i). Therefore, $I_{\text{batt}} = \sum_{i=1}^{i=n} I_{\text{batt}}(i)$.

Task	Duration, mins	Period, mins	Current, mA
1	0.5	2	250
2	0.2	4	100
3	1.0	6	500

(a)

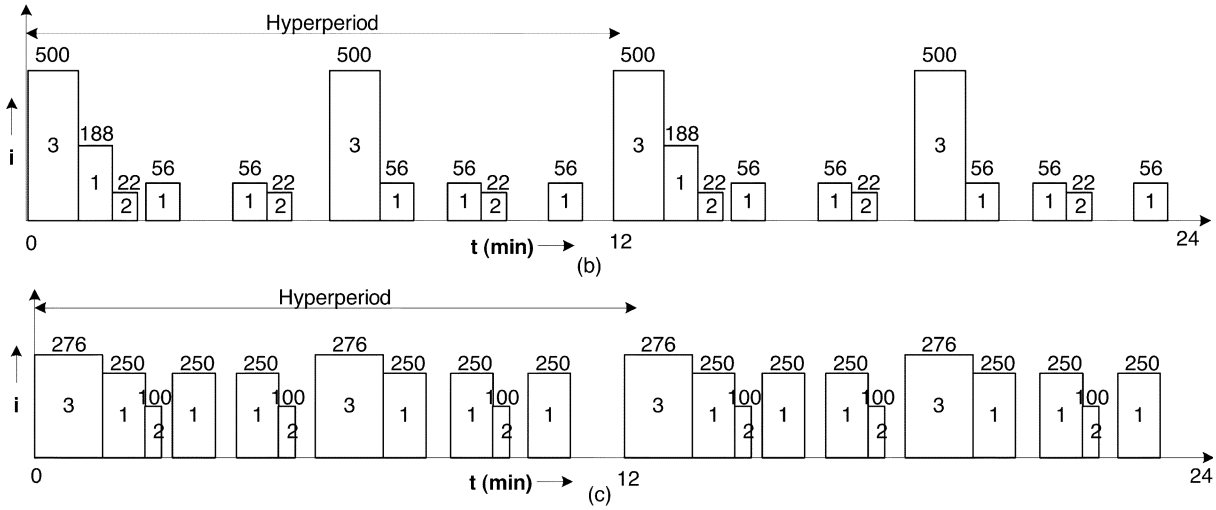


Fig. 10. Example 9: Different approaches to slack utilization for periodic tasks. (a) Initial task specification. (b) Task profile using the proposed algorithm. (c) Task profile using the levelling current algorithm.

TABLE VI
PERFORMANCE COMPARISON OF THE VARIOUS SLACK UTILIZATION SCHEMES FOR PERIODIC TASKS

Algorithm	Analytical Q (mA-min)		Voltage	% drop in voltage
	B1	B2	B2	
Level Current	31019	32337	4.03V	24.5
Our Approach	**31905**	**34155**	**4.10V**	**18.1**

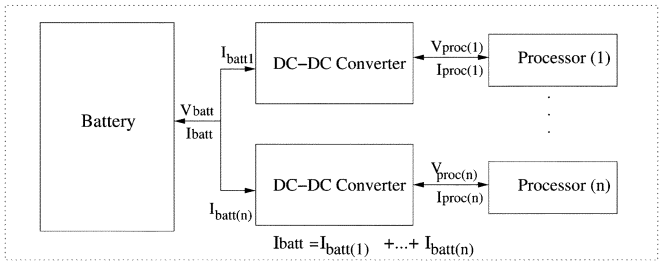


Fig. 11. System-level configuration for a multiprocessor system.

B. Problem Definition

Given the battery parameters α and β, the task graph G describing the task profile and task specifications (arrival times, deadlines and durations), schedule the tasks among the processes such that all the deadline and precedence constraints are met and the profile quality metric Q at the end of the task set is maximized.

C. Proposed Approach

The scheduling algorithm for the multiprocessor 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 processes. The top-level view of the algorithm is described in Fig. 12.

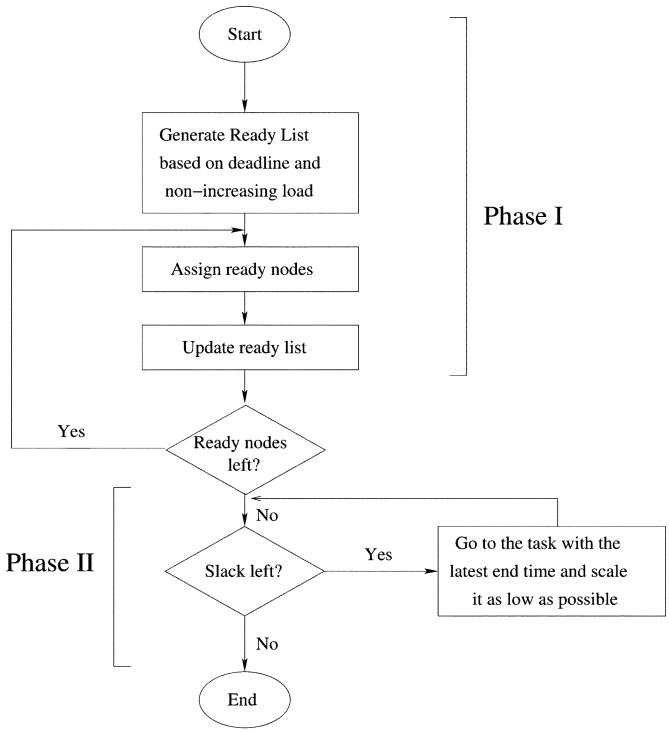


Fig. 12. Top-level view of the multiprocessor task-scheduling algorithm.

1) Phase I: 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 priority function includes deadline (first priority) and nonincreasing order of currents (second priority). Since

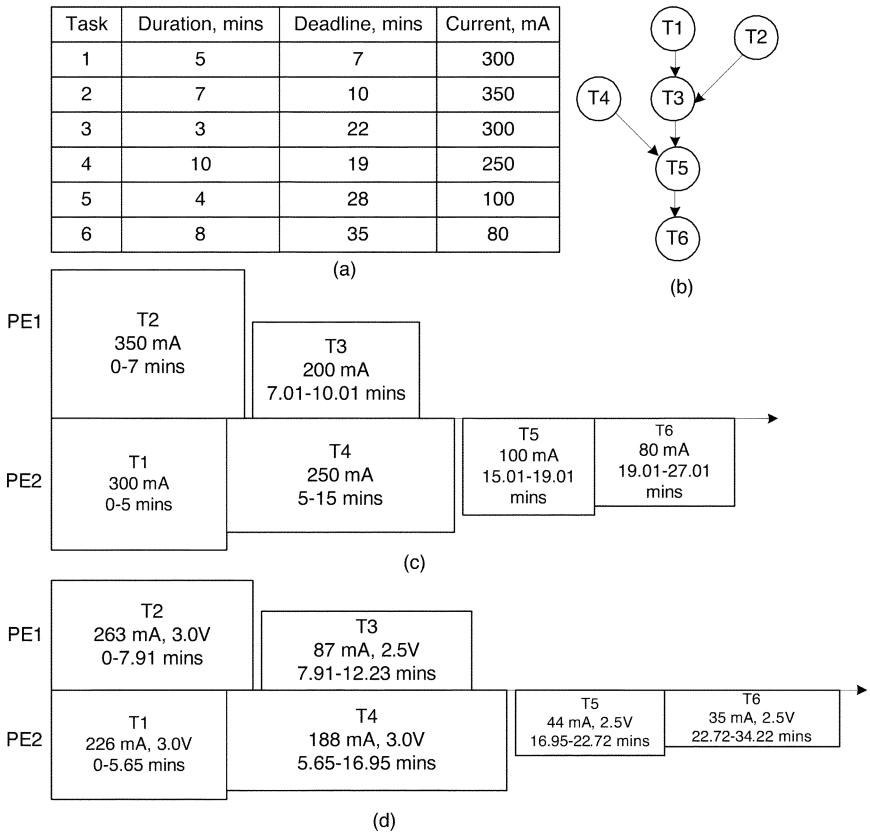


Fig. 13. Example 10: Task scheduling in a multiprocessor system using our approach. (a) Initial task specifications. (b) Input task graph. (c) Processor assignment after Phase one. (d) Processor assignment after Phase two.

I_{batt} is the sum of the loads assigned to each processor, and since each processor also has the current loads arranged in nonincreasing order, this assignment results in the slope of the battery load current being the sharpest, thus conforming to battery Property 4. 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. Note that in the processor assignment, inter-processor communication is avoided as much as possible. This might lead to some processors being active most of the time and others being relatively idle.

2) Phase II: Slack Utilization: This phase is based on Property 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 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 V-A2.

D. Illustrative Example

Example 9 in Fig. 13 illustrates the algorithm in its entirety. Fig. 13(a) describes the initial task specification. The task graph in Fig. 13(b) defines the precedence relation amongst the tasks. Fig. 13(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.

TABLE VII
PERFORMANCE OF THE ALGORITHM FOR THE MULTIPROCESSOR SYSTEM

Algorithm	Analytical Q (mA-min)		Voltage (B2)	% drop in voltage
	B1	B2		
EDF	25607	22019	3.93V	33.6
Level Current	27808	28572	3.99V	28.2
Minimum Energy	27953	28627	4.02V	25.5
Our Approach	**28462**	**31042**	**4.04V**	**23.6**

Fig. 13(d) shows the new schedule after slack utilization. E1 and E2 in 13(c) and (d) refer to the inter-processor communication delay and cost which are assumed to be 0.01 min and 1 mA, respectively. The Q value for Fig. 13(c) is 26 508 mA-min and is 28 462 mA-min for Fig. 13(d) after slack adjustment. For comparison purposes, the task set was scheduled using the popular EDF, level current [13] and the minimum energy technique proposed in [28]. The results have been listed in Table VII. All the competing approaches give lesser residual charges and higher drop in battery voltages compared to our approach.

VII. EXPERIMENTAL RESULTS

This section shows the performance of our algorithm on synthetic examples created with tasks whose profiles were generated by sampling an application running on ITSY every 30 s. Since all these applications are time-varying, each task can be considered to be composed of sub-tasks, where no local scheduling within the sub-tasks can be carried out. Global scheduling amongst these tasks can however be carried out subject to timing and dependency constraints. The load profiles for these

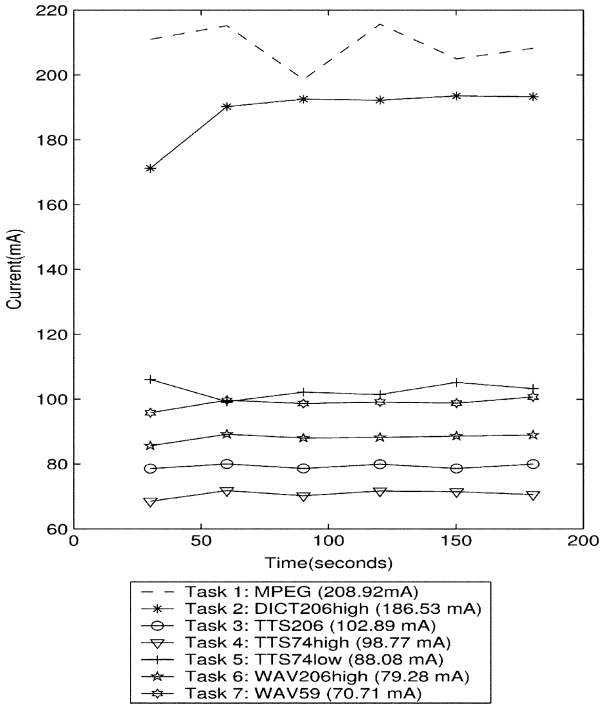


Fig. 14. Current profiles for the real-time applications running on ITSY.

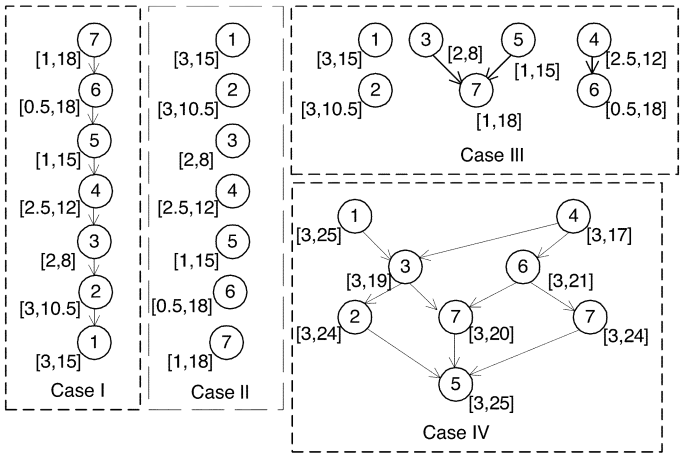


Fig. 15. Synthetic aperiodic task graphs created with applications described in Fig. 14. Node [A, B]: A is the execution time and B is the deadline in minutes.

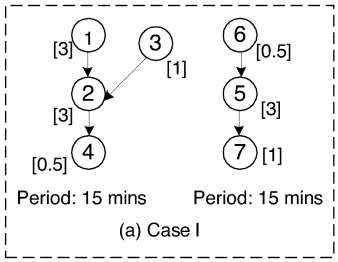
applications for a 3-min interval are shown in Fig. 14 [29]. The time-varying profiles for tasks 3–7 look relatively flat due to the current scale used. For maximum flexibility of scheduling, the arrival times of the tasks in all the experiments are assumed to be $t = 0$ though this is not a limitation of the algorithm.

A. Evaluation Results for Aperiodic Tasks

Fig. 15 describes the dependency and timing constraints for four synthetic test cases generated using tasks 1–7 of Fig. 14. Case I corresponds to the case when the dependencies lead to a strictly increasing profile, Case II corresponds to the case when there are no dependencies and thus has the maximum flexibility for scheduling and slack utilization, and Cases III and IV are randomly generated, with Case III having fewer dependencies than Case IV. The performance of our algorithm (in terms of

TABLE VIII
PERFORMANCE OF THE APERIODIC TASK SCHEDULING ALGORITHM FOR THE SYNTHETIC EXAMPLES IN FIG. 15

Case	Algorithm	Analytical Q (mA-min)		Voltage	% drop in voltage
		B1	B2	B2	
I	Level Voltage	31860	33060	4.074V	20.5
	Level Current	32274	33976	4.100V	18.1
	Our Approach	**32973**	**35602**	**4.140V**	**14.5**
II	Level Voltage	32624	34786	4.129V	15.5
	Level Current	32819	35256	4.136V	14.9
	Our Approach	**33429**	**36997**	**4.172V**	**11.6**
III	Level Voltage	32531	34529	4.127V	15.7
	Level Current	33013	35783	4.146V	14.0
	Our Approach	**33423**	**36985**	**4.171V**	**11.7**
IV	Level Voltage	31261	32654	4.071V	20.8
	Level Current	31503	33187	4.079V	20.0
	Our Approach	**31669**	**33834**	**4.092V**	**18.9**



Task	Period (mins)	Execution Time (mins)
1	8	2
2	6	1.5
3	8	1
4	12	0.5

(b) Case II

Fig. 16. Synthetic periodic task graphs created with applications described in Fig. 14. Node [A]: A is the execution time in minutes. (a) Case I. (b) Case II.

residual charge) is tested against both B1 and B2. The residual charge Q for both the batteries (using the analytical model) and the voltage at the end of the task profile for B2 have been listed in Table VIII. It is clear that application of our heuristic (non-increasing+slack utilization) gives the highest value of Q as well as higher voltage. The drops in voltage due to the various schemes is not an eyecatcher, since the battery was subjected to a load for only approximately 20 min. However, even for such a short duration, our algorithm did significantly better than (say) level current; 14.5% reduction versus 18.1% reduction for Case I, 11.6% reduction versus 14.9% reduction in Case II, 11.7% reduction versus 14.0% reduction in Case III, and 20.0% versus 18.9% in Case IV. Longer task profiles result in a higher drop in the battery voltage but the trend remains the same. For instance, if Case I is repeated 3 times, then the voltage in B2 is 4.0 V (27.2% drop) for level current and 4.036 V (24.0% drop) using our approach. Thus the difference in the drop in voltage does not change much across iterations, $3.6\%(=18.1-14.5)$ for one iteration versus $3.2\%(=27.2-24)$ for three iterations.

B. Evaluation Results for Periodic Tasks

Fig. 16 describes two synthetically generated periodic task graphs: Case I consists of a set of dependent tasks that have the same period and Case II consists of a set of independent tasks with different periods. The node label corresponds to the task

TABLE IX
PERFORMANCE OF THE ALGORITHM AT THE END OF ONE HYPERPERIOD FOR THE PERIODIC TASK GRAPHS DESCRIBED IN FIG. 16

Case	Algorithm	Analytical Q(mA-min)		Voltage	% drop
		B1	B2	B2	in voltage
I	Level Current	32978	35616	4.14V	14.5
	Our Approach	**33243**	**36287**	**4.20V**	**9.0**
II	Level Current	32480	35028	4.11V	17.3
	Our Approach	**32986**	**36430**	**4.15V**	**13.6**

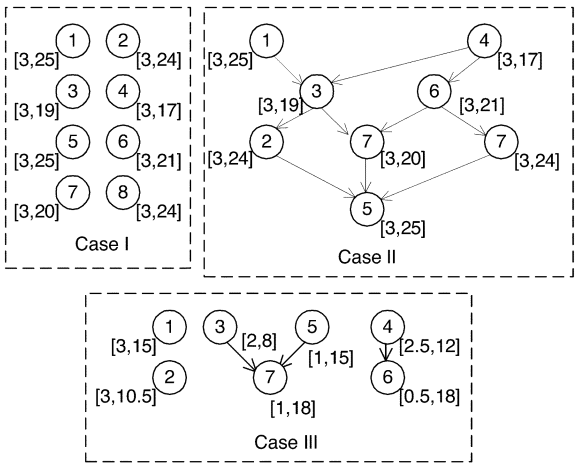


Fig. 17. Synthetic aperiodic task graphs created with applications described in Fig. 14 for the multiprocessor system. Node [A, B]: A is the execution time and B is the deadline in minutes.

TABLE X
PERFORMANCE OF THE ALGORITHMS FOR THE SYNTHETIC EXAMPLES IN FIG. 17 FOR THE MULTIPROCESSOR SYSTEM

Case	Algorithm	Analytical Q (mA-min)		Voltage	% drop
		B1	B2	(B2)	in voltage
I	Level Current	29991	29001	3.99V	28.1
	Our Approach	**37300**	**33417**	**4.18V**	**10.9**
II	Level Current	31764	33625	4.08V	20.0
	Our Approach	**32410**	**34968**	**4.12V**	**16.3**
III	Level Current	32633	34596	4.133V	15.2
	Our Approach	**33954**	**37691**	**4.210V**	**8.1**

described in Fig. 14. The results of the task set after one hyperperiod using our approach and the competing levelling current approach are listed in Table IX. For Case I (II), the drop in voltage using our approach is 9.0% (13.6%) as opposed to 14.5% (17.3%) when level current was used. Thus application of our algorithm resulted in superior performance, in terms of both residual charge Q and battery voltage.

C. Evaluation Results for Aperiodic Tasks on Multiprocessor Systems

Fig. 17 describes the dependency and timing constraints for three three synthetically generated test cases. Case I corresponds to the case when there are no dependencies and thus has the maximum flexibility for scheduling and slack utilization and hence the steepest profile after processor allocation, and Cases II and III are randomly generated. Interprocessor communication has been assumed to cause a delay of 0.01 min and a current overhead of 0.1 mA. The performance of our algorithm is tested against both B1 and B2 and listed in Table X. The performance of our approach is compared against the competing level current approach. It is clear that application of our heuristic does better in terms of both residual charge and voltage. For example, in Case I, the drop in voltage for our approach is only 10.9% as opposed to a drop of 28.1% if level current was used.

VIII. CONCLUSION

In this paper we addressed the problem of aperiodic and periodic static task scheduling for battery-operated systems that increase battery lifetime by maximizing the residual charge or the battery voltage. The scheduling algorithm has been applied to both single-processor as well as multiprocessor systems. The heuristic algorithm is based on nonincreasing ordering of loads, voltage scaling to distribute the available slack and scaling tasks starting from the end of the profile. These properties have been analytically proven to yield better battery performance. While we could not provide a formal proof, the proposed algorithm for the case when there are no task dependencies, is likely to be optimal. Even for the case where there are task dependencies, the algorithm consistently outperforms the competing energy-aware and battery-aware approaches.

Finally, the proposed static scheduling algorithm can be combined with a greedy run-time algorithm to generate battery-aware dynamic task-scheduling algorithms. This is the object of our current research.

ACKNOWLEDGMENT

The authors would also like to acknowledge D. Rakhmatov and S. Vrudhula of the University of Arizona for many useful discussions and for sharing the ITSY data.

REFERENCES

[1] T. Martin, "Balancing batteries, power, and performance: System issues in CPU speed-setting for mobile computing," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, PA, Aug. 1999.
[2] F. Yao, A. Demers, and S. Shankar, "A scheduling model for reduced CPU energy," *IEEE Annu. Found. Comput. Sci.*, no. 11, pp. 374–382, 1995.
[3] Y. Shin, K. Choi, and T. Sakurai, "Power optimization of real-time embedded systems on variable speed processors," in *Proc. ICCAD*, 2000, pp. 365–368.
[4] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling algorithms for minimizing energy," *IEEE Trans. VLSI Syst.*, vol. 11, no. 2, pp. 270–276, Apr. 2003.
[5] I. Hong, D. Kirovski, G. Hu, M. Potkonjak, and M. Srivastava, "Power optimization of variable-voltage core-based systems," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 18, no. 12, pp. 1702–1714, Dec. 1999.
[6] A. Sinha and A. Chandrakasan, "Energy efficient real-time scheduling," in *Proc. ICCAD*, 2001, pp. 458–470.
[7] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "A discrete-time battery model for high-level power estimation," in *Proc. DATE*, 2000, pp. 35–39.
[8] C. Chiassereni and R. R. Rao, "Improving battery performance by using traffic shaping techniques," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 7, pp. 1385–1394, Jul. 2001.
[9] D. Panigrahi, C. Chiasserini, S. Dey, R. Rao, A. Raghunathan, and K. Lahiri, "Battery life estimation of mobile embedded systems," in *14th. Int. Conf. VLSI Design*, Jan. 2001, pp. 57–63.
[10] K. Lahiri, A. Raghunathan, S. Dey, and D. Panigrahi, "Battery-driven system design: A new frontier in low power design," in *Proc. DAC*, Jan. 2002, pp. 261–267.
[11] S. Park, A. Savvides, and M. B. Srivastava, "Battery capacity measurement and analysis using lithium coin cell battery," in *Proc. ISLPED*, Aug. 2001, pp. 382–387.

[12] D. Rakhmatov, S. Vrudhula, and D. Wallach, "A model for battery lifetime analysis for organizing applications on a pocket computer," *IEEE Trans. VLSI Syst.*, vol. 11, no. 6, pp. 1019–1030, Dec. 2003.
[13] J. Luo and N. Jha, "Battery-aware static scheduling for distributed real time embedded systems," in *Proc. DAC*, 2001, pp. 444–449.
[14] D. Rakhmatov, S. Vrudhula, and C. Chakrabarti, "Battery-conscious task sequencing for portable devices including voltage/clock scaling," in *Proc. DAC*, 2002, pp. 189–194.
[15] D. Rakhmatov and S. Vrudhula, "Energy management for battery-powered embedded systems," *ACM Trans. Embedded Computing Syst.*, vol. 2, no. 3, pp. 277–324, Aug. 2003.
[16] D. Linden, *Handbook of Batteries*. New York: McGraw-Hill, 1995.
[17] M. Pedram and Q. Wu, "Design considerations for battery-powered electronics," in *Proc. DAC*, 1999, pp. 861–866.
[18] P. Chowdhury and C. Chakrabarti, "Battery-aware task scheduling for a system-on-a-chip using voltage/clock scaling," in *Proc. SiPS*, 2002, pp. 201–206.
[19] T. Fuller, M. Doyle, and J. Newman, "Simulation and optimization of the dual lithium ion insertion cell," *J. Electrochemical Society*, vol. 141, no. 1, pp. 1–10, 1994.
[20] D. Rakhmatov, "Modeling and optimization of energy supply and demand for portable reconfigurable electronic systems," Ph.D. thesis, Dept. Electr. Comput. Eng., Univ. Arizona, Tucson, May 2002.
[21] L. Benini, G. Castelli, A. Macii, and R. Scarsi, "Battery-driven dynamic power management," in *Proc. IEEE Design and Test*, vol. 18, Mar./Apr. 2001, pp. 53–60.
[22] T. Simunic, L. Benini, and G. De Micheli, "Energy-efficient design of battery-powered embedded systems," *IEEE Trans. VLSI Syst.*, vol. 9, no. 1, pp. 15–28, Feb. 2001.
[23] S. Gold, "A PSPICE macromodel for lithium-ion batteries," in *Proc. Battery Conf.*, 1997, pp. 9–15.
[24] M. Doyle and J. Newman, "Analysis of capacity-rate data for lithium batteries using simplified models of the discharge process," *J. Appl. Electrochem.*, vol. 27, no. 7, pp. 846–856, 1997.
[25] S. Park and M. B. Srivastava, "Dynamic battery state aware approaches for improving battery utilization," in *Proc. ISLPED*, 2001, pp. 382–387.
[26] T. Pering, T. Burd, and R. Broderson, "Voltage scheduling in the lpARM microprocessor system," in *Proc. ISLPED*, 2000, pp. 96–101.
[27] E. L. Lawler and C. U. Martel, "Scheduling periodically occuring tasks on multiple processors," *Inform. Process. Lett.*, vol. 7, pp. 9–12, Feb. 1981.
[28] J. Luo and N. Jha, "Static and dynamic variable voltage scheduling algorithms for real-time hetrogeneous distributed embedded systems," in *Proc. Int. Conf. VLSI Design*, 2002, p. 719.
[29] D. A. Wallach, "Interpreting the battery lifetime of the Itsy version 2.4,", Compaq WRL Tech. Note TN-61, Dec. 2001.

Princey Chowdhury received the B.Tech. degree in electronics and communication engineering from Karnataka Regional Engineering College, Surathkal, India, in 2000 and the M.S. degree in electrical engineering from Arizona State University (ASU), Tempe, in 2003.

Since May 2003, she has been with the Signal Processing and Conversion Group at Maxim Integrated Products, Sunnyvale, CA, as a Circuit Design Engineer. She is currently an Associate Member of Technical Staff and her primary responsibilities are design of high-performance digital and mixed-signal integrated circuits and behavioral modeling of analog circuits. During her Master's work, she was a Research Assistant in the Center for Low Power Electronics Lab at ASU. Her Master's thesis involved developing several static task-scheduling algorithms for maximing battery lifetime in single-processor and multiprocessor DVS-based systems.

Chaitali Chakrabarti received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, India, in 1984, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park in 1986 and 1990 respectively.

Since August 1990, she has been with the Department of Electrical Engineering, Arizona State University, Tempe, where she is now a Professor. Her research interests are in the areas of low-power systems design including memory optimization, high-level synthesis and compilation, and VLSI architectures and algorithms for signal processing, image processing, and communications.

Dr. Chakrabarti is a member of the Center for Low Power Electronics, the Consortium for Embedded and Inter-Networking Technologies and Connection One. She received the Research Initiation Award from the National Science Foundation in 1993, a Best Teacher Award from the College of Engineering and Applied Sciences, ASU, in 1994, and the Outstanding Educator Award from the IEEE Phoenix section in 2001. She has served on the program committees of ICASSP, ISCAS, SIPS, ISLPED, and DAC. She is currently an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the *Journal of VLSI Signal Processing Systems*.