PAPER

Static Task Scheduling Algorithms Based on Greedy Heuristics for Battery-Powered DVS Systems

Tetsuo YOKOYAMA[†a)], **Gang ZENG**[††], ***Nonmembers***, **Hiroyuki TOMIYAMA**[†††], ***and*** **Hiroaki TAKADA**[††], ***Members***

SUMMARY The principles for good design of battery-aware voltage scheduling algorithms for both aperiodic and periodic task sets on dynamic voltage scaling (DVS) systems are presented. The proposed algorithms are based on greedy heuristics suggested by several battery characteristics and Lagrange multipliers. To construct the proposed algorithms, we use the battery characteristics in the early stage of scheduling more properly. As a consequence, the proposed algorithms show superior results on synthetic examples of periodic and aperiodic tasks from the task sets which are excerpted from the comparative work, on uni- and multi-processor platforms, respectively. In particular, for some large task sets, the proposed algorithms enable previously unschedulable task sets due to battery exhaustion to be schedulable.

key words: *battery-aware voltage scheduling, dynamic voltage scaling, low power, real-time systems*

1. Introduction

Increases in battery capacity and improvements in battery utilization have proved to be indispensable due to higher demand for functionality resulting in a drastic increase of energy consumption. On the other hand, the energy density of the battery, though gradually improved, have already reached a half or a third of the theoretical limits [1]. It is therefore of great importance to reduce the energy consumption by controlling software in the battery-powered embedded systems. The challenge is to appropriately handle the non-trivial battery characteristics, such as *recovery* and *non-linearity* of battery capacity, which are dependent on the current load history [1]–[3]. A real battery recovers its charge when it is idle. The current load affecting the total capacity of batteries is not uniform but rather depends on the load history. Once the voltage of the battery reaches its threshold, a battery becomes *exhausted*. Then, its current is unavailable and never recovers without an external power supply. To use battery capacity to the greatest extent, it is impossible to apply existing energy optimization techniques as they are. Therefore, the aforementioned battery characteristics must be reflected.

Battery characteristics have been widely studied [4]. Although computationally expensive low-level electrochemical models (*e.g.*, Dualfoil [5]) are accurate, high-level battery models provide reasonably accurate approximation gained by lightweight computation [6]–[11]. Based on those models, a number of battery-aware voltage scheduling algorithms on dynamic voltage scaling (DVS) systems have been proposed [2], [3], [12].

Specifically, incorporating battery properties, such as recovery effect and non-linearity, Rakhmatov and Vrudhula [6], [7] developed a high-level analytical battery model with only two configuration parameters for each battery instance. Accuracy of their model was confirmed by the low-level electrochemical simulation Dualfoil [5] within approximately 5% error according to their report. Rakhmatov and Vrudhula [2] specified various important properties of their mathematically formulated cost function, and several efficient static battery-aware voltage scheduling algorithms. Chowdhury and Chakrabarti [3] identified several insightful battery properties and extended the work of Rakhmatov, Vrudhula, and Chakrabarti [2], [13]. They proposed battery-aware voltage scheduling algorithms not only for periodic tasks on uniprocessor platform but also for both aperiodic and periodic tasks on both uni- and multi-processor platforms.

Their improvement relies on heuristics derived from battery characteristics, such as the steepest profile and non-increasing ordering in the early stage of scheduling. In this paper, we investigate the implications caused by battery properties, which are identified in the aforementioned previous work [2], [3]. As the result of the proper investigation of those properties, we propose new static voltage scheduling algorithms for the battery-powered embedded systems, based on greedy heuristics suggested by several battery properties and Lagrange multipliers. We target both uni- and multi-processor systems. Our method shows better results in the aperiodic task sets of time varying load used in the previous works on multiprocessor systems, as well as in the periodic task sets on uniprocessor systems. To obtain optimal periodic task scheduling with respect to the battery load, we have found that a special care is necessary, while it is not necessarily considered in energy minimization.

One of the limitations of our approach is caused by the selection of the objective function to minimize. As mentioned above, it deviates from the low-level electrochemical simulation Dualfoil [5] by at most 5%. Moreover, Dualfoil does not perfectly reflect real battery behavior. Also, only

Manuscript received February 5, 2009.
Manuscript revised December 15, 2009.

[†]The author is with the Department of Software Engineering, Nanzan University, Seto-shi, 489–0863 Japan.
[††]The authors are with the Graduate School of Information Science, Nagoya University, Nagoya-shi, 464–8601 Japan.
[†††]The author is with College of Science and Engineering, Ritsumeikan University, Kusatsu-shi, 525–8577 Japan.
a) E-mail: tyokoyama@acm.org

DOI: 10.1587/transinf.E93.D.2737

Li-ion battery parameters are used when evaluating our result.

This paper is organized as follows. First, we present the non-ideal battery characteristic using a motivating example (Sect. 2). After representing the system and battery models we rely on (Sect. 3), we propose our voltage scheduling algorithms (Sect. 4). The proposed algorithms are evaluated through comprehensive experiments (Sect. 5). Finally, we conclude with a few remarks (Sect. 6).

2. Motivation

Figure 1 represents four profiles of two identical tasks (current 250 mA, duration 2 min in the highest voltage setting, deadline 6 min) on a battery-powered uniprocessor system, in which processor speed and power change continuously.

Figure 1 (a) shows the increase current profile. Since the idle time increases the nominal residual charge in batteries, the later we measure the nominal battery capacity, the higher value we obtain. The consumed capacity is measured by the objective function σ_B (see Sect. 3 in detail). The higher σ_B stands for the smaller nominal residual charge available in batteries at observation time B. Once σ_B reaches some threshold (denoted as α in this paper), the batteries become exhausted; The batteries are inactive and no longer recoverable without external power supply. The objective function at time 6 min (σ_6) is 1566 mA·min and it decreases to 1115 mA·min at time 12 min (σ_{12}). Each of two objective functions is the worst in the four cases.

For battery-unaware scheduling, it is well-known that only a single processor speed is sufficient to obtain the optimal profile [14]. The time between the end of task execution and deadline is called *slack time*. If we distribute the slack time equally among two identical tasks, we obtain level current profile (Fig. 1 (b), cf. [12]), which is an optimal profile in terms of the amount of energy dissipated from batteries. However, due to non-linearity of the objective function, it is not always an optimal profile as far as optimization of battery residual charge is concerned. As we will see later on, this profile is optimal in case of ideal battery.

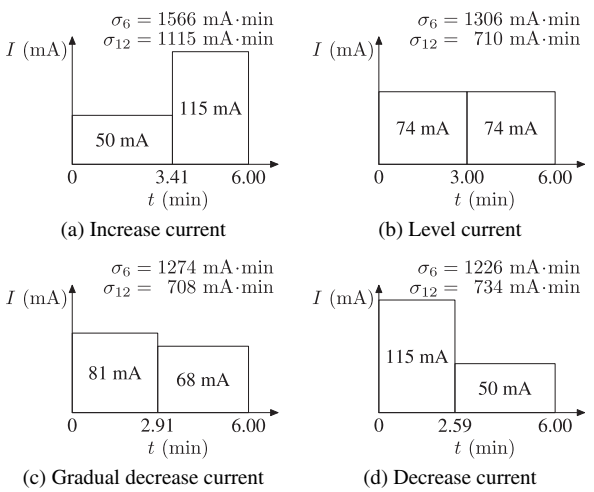


Fig. 1 Motivating example.

The decrease current profile results in better battery performance compared to increase current profile. Figure 1 (c) is an optimal profile minimizing the objective function σ_{12}. Despite higher energy consumption in the level current approach (Fig. 1 (b)), the battery loads σ_6, σ_{12} are reduced.

Figure 1 (d) is an optimal profile minimizing σ_6. This decrease current is obtained by swapping the two tasks in the increase current profile in Fig. 1 (a). Those two profiles consume exactly the same amount of energy, but the battery loads differ (34.2% improvement with respect to σ_{12}). This implies the importance of the task order (*i.e.*, the *history of load*) in the battery-aware voltage scheduling. The gradient of this task load is steeper than in the gradual decrease current profile (Fig. 1 (c)), and this results in the worse effect on σ_{12}. Namely, the steepest non-increasing load current profile is *not* always optimal. Hence, heuristics for choosing the steepest non-increasing load current profile [3] is not always the best choice.

This example becomes an intratask voltage scheduling problem [15], if we regard two identical tasks as one piece of task and have opportunity to change the speed when a half of the task is finished. In the battery-aware voltage scheduling, even when each task consumes power uniformly, there is still room to optimize the total available capacity in the battery by switching voltages during the task execution. The difference of σ_6 between Fig. 1 (b) and Fig. 1 (d) shows not negligible improvement (6.1%).

Energy minimization is *not* exactly equivalent to battery optimization. In summary, when considering the battery-aware voltage scheduling, we need to pay attention to the recovery effect and the history of load, and we should not rely too much on the steepest non-increasing profile heuristics to obtain an efficient profile.

The importance of the research on the better battery utilization is also reinforced by the fact that it is somewhat independent of the energy optimization of the other parts of the systems. For example, if a subsystem, which consumes 10% of the total energy, reduces the energy consumption by 50%, only 5% is reduced in total. However, if energy, which remained in a battery after battery failure, is used in the system it is exactly equal to the increase of the available energy for performing system.

3. Preliminaries

This section explains the assumptions used in the following sections, in order to make this paper self-contained.

3.1 System Configuration

We assume DVS-enabled uni- and multi-processors. For the sake of simplicity, the DC-DC conversion efficiency is assumed to be 100%. The ratio of the initial task duration Δ and the new task duration Δ^* after scaling the task voltage

V_{dd} down by factor s (*i.e.*, V_{dd}/s) is

$$\frac{\Delta^*}{\Delta} = s\left(1 + \frac{2(s-1)V_{\mathrm{th}}}{V_{\mathrm{dd}} - V_{\mathrm{th}}}\right) \tag{1}$$

with V_{th} a threshold voltage. The battery current I_{batt} scales by s^3, *i.e.*, $I^*_{\mathrm{batt}}/I_{\mathrm{batt}} = s^{-3}$ with I^*_{batt} a battery current after scaling the task voltage.

Our target processor is compatible with the StrongARM SA-1100 microprocessor [16]. The operating voltage ranges over set {3.3, 3.0, 2.7, 2.5, 2.0}. The threshold voltage V_{th} is 0.4 V. For the sake of simplicity, we assume no time and energy overhead due to the change of voltage setting.

In most modern embedded systems, dynamic energy consumption is dominant. Therefore, in this paper, static energy consumption is ignored, although it is one of major concerns in the near future. All the assumptions above are consistent with the work on battery-conscious voltage scheduling [3] for comparison purpose.

3.2 Battery Model

There has been no single model capturing the behavior of both charge and voltage at the same time. We focus on charge sensitive model and ignore voltage sensitive models [5], [17]. As a model for variable load, we use the high-level analytical model of Rakhmatov and Vrudhula's [6], [7]. The load on battery $i(t)$ and battery life time L are related by equation

$$\alpha = \int_0^L i(t)\mathrm{d}t + \int_0^L i(t)\left(2\sum_{m=1}^{\infty} e^{-\beta^2 m^2 (L-t)}\right)\mathrm{d}t \tag{2}$$

with α (mA·min) and β ($\mathrm{min}^{-1/2}$) being constants, uniquely determined for each battery. α corresponds to the battery's theoretical capacity and β represents the recovery rate.

The first term on the right hand side sums up the discharged capacity from the battery from time 0 to L. The second term represents the residual charge in the battery, unavailable at time L. It should be noted that once battery voltage becomes lower than some threshold, the residual charge in the battery cannot be used any more. If the second term on the right hand side is negligible (*e.g.*, $\beta \to \infty$), the battery behavior is nearly optimal. If α is significantly large, we do not have to consider the battery failures (exhaustion). By means of insertion of the period of no load, *i.e.*, $i = 0$, or low load, the battery life L increases; the battery at rest recovers its charge. Let τ_k (k in short) be tasks executed during the period of Δ_k starting at time t_k. For brevity, we estimate the load of each task k on the battery to be a constant I_k in the highest voltage setting. If the tasks are sequentially computed and the battery life ends at task u, the battery capacity equation computed with continuous current function (2) becomes the discrete equation [2], [6]

$$\alpha = \sum_{k \in S_u} I_k F(L, t_k, t_k + \Delta_k, \beta) + I_u F(L, t_u, L, \beta) \tag{3}$$

with a set S_u consisting of the tasks executed before task u and auxiliary function

$$F(T, s, f, \beta) = f - s + 2\sum_{m=1}^{m_{\max}} \frac{e^{-\beta^2 m^2 (T-f)} - e^{-\beta^2 m^2 (T-s)}}{\beta^2 m^2} \tag{4}$$

in which s represents the start time, f the finishing time, and T the observation time. If $t \notin [t_k, t_k + \Delta_k)$ for all $\tau_k \in S_u + \{\tau_u\}$ in the uniprocessor systems, then t is in the idle time. Since in theory $m_{\max}$ is infinite, the number of terms in the sum of infinite series provides a tradeoff between the accuracy and the amount of computation. In [6] and [7], a graph ranging $1 \le \beta^2 L \le 10^2$ shows that the sums of the first 10 and 100 000 terms create negligible difference. This fact implies that the first 10 terms are a good approximation.

We justify this observation analytically. Since the formula (summands) under Σ notation in F is monotonically decreasing with m and T, by using the solution of Basel problem[†], we obtain, for any $m_{\max}$

$$\sum_{m=1}^{m_{\max}} \frac{e^{-\beta^2 m^2 (T-f)} - e^{-\beta^2 m^2 (T-s)}}{m^2} < e^{-\beta^2 (T-f)} \frac{\pi^2}{6}. \tag{5}$$

Given $m_{\max}$ and the upper bound of $f - s$, the upper bound of error is obtained. For example, assuming $f - s \ge 1$ min, $m_{\max} = 10$, $\beta = 0.637\,\mathrm{min}^{-1/2}$, $T - f = 10$ min, error is bounded by 1.03×10^{-3}. The sum of the first 10 terms has approximately at most 0.2% error. Therefore, we use $m_{\max} = 10$ in the formula of F (4).

We focus on the task sets of the middle duration range (0.5 min to 20 min). This is because firstly the load frequencies higher than 1 Hz can be filtered owing to the late response of the battery device [1] and battery-charge optimization is not effective for very fine-grained (< 10 ms) tasks [18]. Secondly, for very coarse-grained (> 30 min) tasks, battery-aware voltage scheduling is not much superior to energy optimal scheduling [18]; Battery optimization is almost equivalent to energy minimization in this range.

In this paper, we do not consider self-discharge mechanisms, aging caused by discharge/recharge repetition, and dependence on temperature, since this model does not take them into account.

3.3 Cost Function

A profile of n tasks consists of a set of the current I_k, the starting time t_k, and the duration Δ_k. For a given profile of n tasks and the observation time B, the battery-aware cost function [7]

$$\sigma_B = \sum_{k=0}^{n-1} I_k F(B, t_k, t_k + \Delta_k, \beta) \tag{6}$$

is to be minimized. The subscript B of σ is omitted if it

[†]Euler solved the Basel problem and obtained the formula $\sum_{m=1}^{\infty} 1/m^2 = \pi^2/6$.

is insignificant or obvious from its context. σ_B models a measure of the accumulated battery load until time B. When battery parameter β or the observing time B becomes large ($\beta \to \infty$, $B \to \infty$), the third term on the right hand side in the formula of F (4) disappears, and the battery becomes ideal ($\lim_{\beta\to\infty} \sigma_B = \sigma_\infty$).

To make the value σ meaningful, two conditions must be satisfied. Firstly, all tasks must terminate before observation time B:

$$\forall k.\ t_k + \Delta_k \le B\,. \tag{7}$$

Secondly, the battery must satisfy the *endurance constraint*; the battery must not be exhausted before B:

$$\forall t \le B.\ \alpha \ge \sum_{k=0}^{n-1} I_k F(t, \min\{t, t_k\}, \min\{t, t_k + \Delta_k\}, \beta)\,. \tag{8}$$

It should be noted that if the values of the second and third arguments of F are equal, F becomes zero.

For comparison purpose, we use the same battery parameters in all profiles (including motivating examples in Sect. 2) as in [2], [3], [7] ($\alpha = 40\,375\,\mathrm{mA{\cdot}min}$, $\beta = 0.273\,\mathrm{min}^{-1/2}$).

We will use the largest deadline of given tasks for the observation time B, instead of the end time of each profile, which was used in the previous work [2], [3]. As a result, even if all tasks finished earlier than the deadline, it is not always disadvantageous because of the usage of the remaining time for recovery.

3.4 Cost Function Properties

The properties derived by analysis of cost functions are described by proposers of the battery model [2] and independently identified by [3]. The cost function decreases when swapping two ascending tasks with no idle time in-between. Therefore, assuming no endurance constraint, no slack time interleaved between tasks, and no task dependency, σ is minimized if tasks are in non-increasing load order (Fig. 1 (d)), and is maximized if tasks are in non-decreasing load order (Fig. 1 (a)). When designing algorithms, we assume that the decrease of voltage results in increase in the duration of tasks as well as decrease in the amount of load:

$$\begin{aligned} &V \ge V' \\ &\quad \Rightarrow \Delta(V) \le \Delta(V') \text{ and } I(V)\Delta(V) \ge I(V')\Delta(V')\,. \end{aligned} \tag{9}$$

Under this assumption, several properties hold. When $B = \max_k\{t_k + \Delta_k\}$, the degradation of voltage of task k with the new observation time $B^* = \max_k\{t_k + \Delta_k^*\}$ does not make the cost function σ_{B^*} worse and the endurance constraint satisfied in the original profile is still satisfied. For two identical tasks, the earlier task is less expensive than the later task. When slack time is used for battery rest, it is optimal if it is used as late as possible. However, for the first task failing the endurance constraint, the voltage down scaling is better than the battery rest. We will take account of those properties in the next section.

Nevertheless, the existing battery-aware scheduling algorithm did not properly consider the implications of some battery properties, especially non-increasing profile. We make two remarks as follows. Firstly, a profile in the steepest decreasing order is not always optimal, if tasks are dependent on each other and/or their order is imposed by deadline constraints (cf. Sect. 2); We cannot exclude the possibility that downscaling the processing speed for the previous tasks is more optimal than downscaling the failing (battery exhausting) task. We need a quantitative measure to construct algorithms. Secondly, while task ordering does not affect the amount of energy consumption once voltages of tasks are fixed in terms of energy minimization, it does affect the nominal residual battery capacity, especially when large task sets are given and multiprocessor platforms are considered (see Sect. 5). We will construct static voltage scheduling algorithm by taking those two observations into consideration.

4. Battery-Aware Voltage Scheduling

A voltage scheduling problem for the battery-powered DVS system can be formulated as a non-linear optimization problem (Fig. 2). In this paper, we do not consider task preemption. The input consists of six ordered sets of voltages S_V on which a system operates, frequency S_ϕ, current S_I, release time S_r, duration S_Δ, and deadline S_d, task dependency graph G, observation time B, battery parameters α, β, specified in Sect. 3.3, and the number of processors p. The output consists of two ordered sets representing scheduled voltage S_{V^*} and start time S_t for each task. S_{V^*} is uniquely determined by scheduled current I^*s and/or scheduled duration Δ^*s.

The objective function to be minimized is σ_B in the battery-aware cost function (6). Five constraints are imposed: 1) release time, 2) number of processor, 3) dependency, 4) deadline, and 5) current load endurance. First, release time of each task must be smaller or equal to each starting time. Second, the number of processes running simultaneously must be smaller or equal to the number of pro-

INPUT: $S_V,\ S_\phi,\ S_I, S_r, S_\Delta, S_d, G, B, \alpha, \beta, p$
OUTPUT: S_{V^*}, S_t
OBJECTIVE:

$$\text{minimize } \sigma_B \ \text{ s.t. } \ \sigma_B = \sum_{k=0}^{n-1} I_k^* F(B, t_k, t_k + \Delta_k^*, \beta)$$

CONSTRAINTS:

1. $r_k \le t_k$
2. $\forall t.\ \#(t \in [t_k, t_k + \Delta_k^*)\ ) \le p$
3. k' depends on k in $G \ \Rightarrow\ t_k + \Delta_k^* \le t_{k'}$
4. $\forall k.\ t_k + \Delta_k^* \le d_k$ and $\forall k.\ t_k + \Delta_k^* \le B$
5. $\forall t \le B.\ \sum_{k=0}^{n-1} I_k^* F(t, \min\{t, t_k\}, \min\{t, t_k + \Delta_k^*\}, \beta) \le \alpha$

Fig. 2 Battery-aware voltage scheduling problem.

cessors. Third, the task dependency represented by G must be preserved. Fourth, the deadline d_k for each task k should be met, and the length of profile must be smaller than observation time B. Fifth, the battery must survive all the tasks without battery exhaustion until observation time B.

A voltage scheduling problem is NP-hard, even if tasks have the fixed-priorities [19]. Therefore, the efficient and effective heuristics is needed. The energy minimization problem is in general an instance of the battery-aware energy optimization problem, since the latter becomes the former when $\alpha, \beta \to \infty$.

We will add the following assumptions and make the problem simpler. For simplicity we do not consider intra-task voltage scheduling [15], *i.e.*, in our intertask voltage scheduling the power and performance are approximated to be uniform within a single task, as is often assumed. The arrival time, the deadline, the current, and the dependence relation are known in advance before execution.

Our major idea is simple greedy heuristics. The objective function σ reaches its minimum if the equation

$$\frac{\partial\sigma}{\partial\Delta_1} = \frac{\partial\sigma}{\partial\Delta_2} = \cdots = \frac{\partial\sigma}{\partial\Delta_n} \tag{10}$$

is satisfied. This equation is obtained by the Lagrange multiplier method. Here, we assume there is no slack time between tasks. The performance of the task will be decreased, so that the energy increase is the most effective. The effectiveness of the energy increase is measured by the decreased cost per the decreased duration $\partial\sigma/\partial\Delta_k$. Since we focus on discrete DVS and configurations, the effectiveness is measured by

$$\frac{\sigma^* - \sigma}{\Delta_k^* - \Delta_k}. \tag{11}$$

It should be noted that this expression does not assume any specific function σ.

4.1 Voltage Scheduling for Uniprocessor Systems

The proposed algorithm consists of two phases (Fig. 3): I) obtain a feasible solution and II) distribute slack time.

In Phase I, battery-unaware scheduling algorithm without voltage scaling is used. In this paper, we use the earliest deadline first (EDF) algorithm, but this choice is not essential. The power is scaled down starting from the highest power initial solution, which hopefully satisfies the deadline constraints. When it does not satisfy the endurance constraint, our scheduling algorithm returns "Failure".

To repair the battery failure, we repeatedly scale down the speed of the failed tasks or the tasks appeared before them, in such a way that reduction of their speed by one level results in the greatest value using the discrete voltage downscaling effectiveness measure (11) within timing constraints (greedy choice). It should be noted that the downscaled task is not always the failed task unlike in case of the scheduling algorithms in [3], where even the failed task

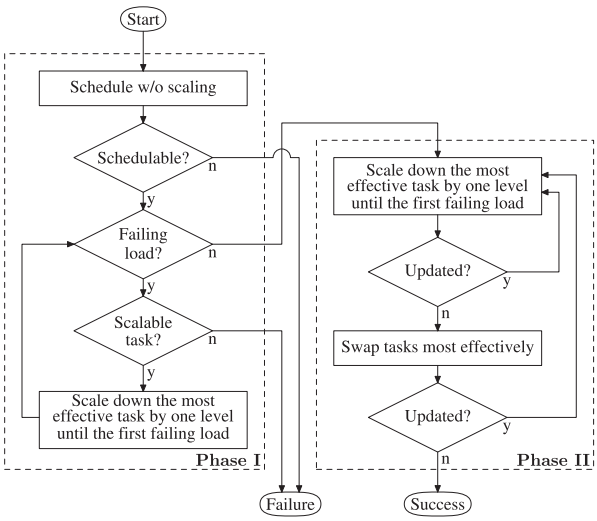


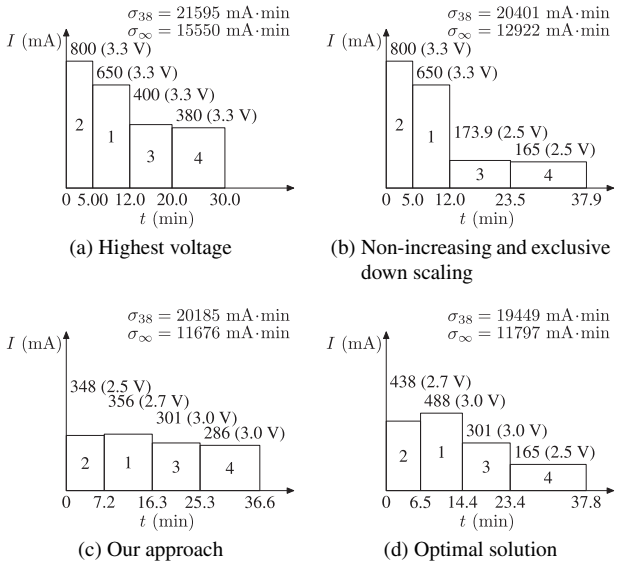
Fig. 3 Battery-aware uniprocessor voltage scheduling algorithm.

is not assigned the lowest voltage. If the task is not a failing task, the voltage downscaling is not guaranteed to be superior to the insertion of the battery idle time. But such slack time tends to be relatively larger compared to our time range. Therefore, as in [3] and unlike in [2], [13], we do not consider the insertion of the idle time. If the effectiveness of DVS evaluated by the discrete voltage downscaling effectiveness measure (11) is negative, the voltage of the task is not scaled down. This case never occurs if assumption (9) holds at any moment. Strictly speaking in our assumption, due to duration scaling equation (1), it does not always hold, but the effect is limited and can be ignored when designing scheduling algorithm.

In Phase II, we repeatedly use the available slack time by scaling down speeds of tasks to achieve the most effective decrease of the cost with respect to the discrete voltage downscaling effectiveness measure (11). Next, we swap the task order if the result has lower cost without violating the deadline constraints. We repeat this phase until no tasks can be swapped. Consideration of batteries is a necessary condition for this new method of swapping tasks to optimize the energy efficiency. It should be noted that, if all tasks have the same release time and deadline, the scheduled subprofiles are always placed in the non-increasing order [2], [3].

Generally, the energy efficiency is achieved only if σ has an asymptotic lower bound being the function proportional to an exponential function of Δ, in which exponent x is greater than one, *i.e.*, $\sigma > \Delta^x$ $(x > 1)$. If σ is monotonic with respect to Δ, scaling down the voltage to the next level is more effective than to any other levels. This observation justifies the reduction of the speed of the most effective task by one level at each time.

The proposed algorithm depends on heuristics and thus it is not guaranteed to return the optimal solution, as we will see in the next example (Fig. 4). Nevertheless experiments in Sect. 5 show that for the same task set proposed algorithm results in better schedules when comparing to previ-



(a) Highest voltage (b) Non-increasing and exclusive down scaling

(c) Our approach (d) Optimal solution

Fig. 4 Load profiles for tasks in Table 1. The numbers displayed in the bars are the task numbers and the numbers above them the current in miliampair.

Table 1 Initial task specifications.

Task #	Duration (min)	Deadline (min)	Current (mA)
1	7	18	650
2	5	10	800
3	8	26	400
4	10	38	380

ous works [3].

We do not assume continuity and differentiability of the objective function, and we can apply our method to the practical objective functions, most of which are neither continuous nor differentiable, especially when we consider memory accesses and peripherals.

Table 1 describes an initial task set of aperiodic tasks. Figure 4 (a) shows the load profile given by EDF scheduling algorithm in the highest voltage setting. The value of objective function at infinite time σ_∞ is 28.0% smaller than its value at time 38 min, *i.e.*, σ_{38}. The difference shows the theoretical maximum bound of recovery. This profile returns the worst result among four profiles given in Fig. 4. It should be noted that to schedule the tasks we used the objective function at time 38 min.

Figure 4 (b) shows the load profile achieved by two existing approaches, *i.e.*, non-increasing [3] and exclusive down scaling [13]. These two accidentally result in the same profile. Both algorithms assign all tasks to the highest available voltage, schedule tasks by the battery-unaware algorithm, recover the failure if any occurs by downscaling the task, and repeatedly distribute the remaining slack to preceeding tasks whose voltage can be lowered (as the result, the profile becomes steep) without violating deadline constraints.

Figure 4 (c) shows the load profile achieved by the proposed algorithm. Unlike two previously presented algorithms, our algorithm does not always return the non-increasing profile, as this figure shows. However, both objective functions σ_{38} and σ_∞ show the improved values.

Table 2 Initial task specifications of periodic tasks.

Task #	Duration (min)	Deadline (min)	Current (mA)
1	0.5	2	250
2	0.2	4	100
3	1.0	6	500

Figure 4 (d) shows the load profile by the exhaustive search, resulting in the optimal solution at time 38 min. While the proposed algorithm is not optimal, even when there are no task dependencies, objective function obtained in our approach differs from the optimal solution by at most 4%. Note that the optimal solution was obtained by brute-force search. The optimal solution does not always result in a non-increasing order. The simultaneous reduction of the objective functions at different observation times is by no means inevitable. In fact, the value of σ_∞ in Fig. 4 (c) is more optimal than one in Fig. 4 (d).

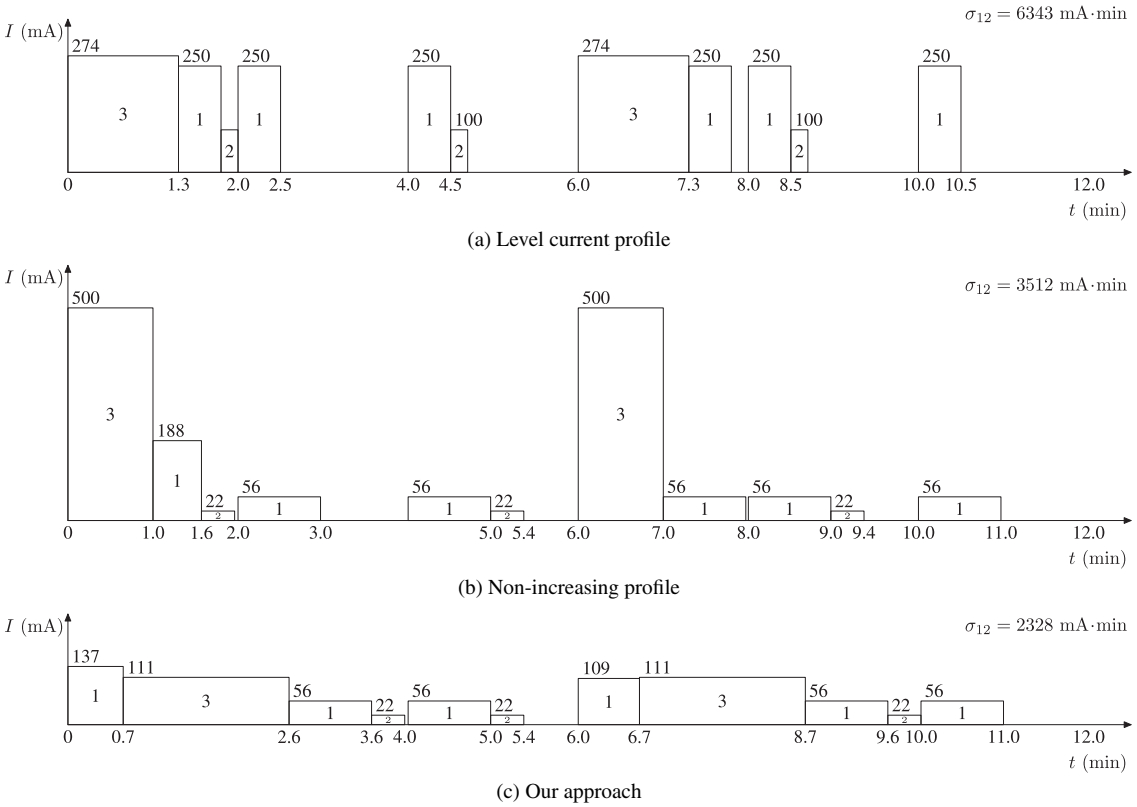
It should be noted that, for the task set in Table 1, proposed algorithm returns more efficient profile than the previously proposed algorithms. This result implies that, on the contrary to the intuition described in [3], the non-increasing scheduling algorithm is not optimal for the case when there is no task dependency.

4.2 Scheduling Periodic Tasks

The least common multiple (LCM) of the periods of all periodic tasks is called a *hyperperiod*. In a conventional voltage scheduling problem, the optimal scheduling for each hyperperiod is identical and the simple repetition of the optimal profile becomes globally optimal scheduling in terms of energy consumption. However, in a battery-aware voltage scheduling, it is *not* always optimal in terms of the nominal residual charge due to the non-linearity of the objective function (see the experimental result in Sect. 5). The optimal profile for each hyperperiod is not always the same. Therefore, when the same scheduling is performed periodically, the superiority of the battery-aware scheduling to battery-non-aware scheduling is not obvious.

The latter the executed tasks are at a certain observation time, the more temporal decrease of the battery charge is observed. On the other hand, such temporal effects of former tasks are smaller. The most efficient scheduling in terms of nominal residual battery charge at certain observation time is not most efficient when the observation time changes. Therefore, if we can change scheduling online, to optimize the nominal battery residual charge at the observation time, at the former periods we would start with non-battery scheduling. As the time goes by we needed to be more and more taking battery effects into consideration.

Figure 5 illustrates the comparison of slack utilization by three algorithms, for the periodic tasks specified in Table 2. Profile shown in Fig. 5 (a) is generated by the level current algorithm [12], which aims to reduce the peak



(a) Level current profile

(b) Non-increasing profile

(c) Our approach

Fig. 5 Load profiles of the first hyperperiod. The numbers displayed in the bars are the task numbers and the numbers on them the current in mili-ampair.

power. Figure 5 (b) is substantially improved due to the non-increasing task ordering achieved by battery-aware periodic task voltage scheduling algorithm in [3], even though the peak current is significantly greater than in case of the level current profile. The objective function σ_{12}, the value after execution of the first hyperperiod, shows significant improvement (45.6%). All sequential subprofiles are non-increasing, but the profile in Fig. 5 (b) still has a lot of unused slack time.

The profile produced by the proposed algorithm (Fig. 5 (c)), in which swapping task ordering plays an important role, further distributes the slack time. The objective function σ_{12} shows the further considerable improvement (33.7% from the non-increasing profile, 63.3% from the level current profile). The resulting profile of the proposed algorithm does not always have a non-increasing task ordering. In fact Fig. 5 (c) contains a non-increasing subprofile, while the overall tendency is non-increasing. Better slack time utilization, however, shows better battery efficiency.

When the same scheduling is repeated for all hyperperiods, schedule A is superior to schedule B at any observation time if both the nominal and actual loads of schedule A is smaller than that of schedule B at the end of the hyperperiod [2]. In Fig. 5, since the profile of our approach is superior to the level current profile and non-increasing profile in both nominal and actual loads, our profile is superior to them for any number of hyperperiods at any observation time.

4.3 Voltage Scheduling for Multiprocessor Systems

Our multiprocessor voltage scheduling algorithm illustrated in Fig. 6 is based on uniprocessor voltage scheduling algorithm from Sect. 4.1. We followed the basic idea of multiprocessor scheduling algorithm in [3]. It consists of two phases. In Phase I, we use the list based scheduling in which the first key is deadline in non-decreasing order and the second key is current in non-increasing order. Since any task schedule must not violate deadline, the deadline is the primary key. In addition to the usual deadline order, the introduction of the non-increasing current order is due to battery effects that non-increasing order is superior to non-decreasing order, as we have seen in Sect. 2. The energy optimized scheduling is achieved when the task load is equally distributed to each processor. Therefore, we use worst fit scheduling. Scaling down of voltage is applied to the most effective task according to the discrete voltage downscaling effectiveness measure (11), and the tasks following the downscaled task are replaced in the ready list. Phase II is the same as in the uniprocessor voltage scheduling algorithm (Fig. 3). It should be noted that tasks assigned to the different processors are also swapped. It should be noted that as in uniprocessor algorithms the sharpest load profiles does not always result in the optimal scheduling algorithm.

For a given taskset presented in Fig. 7, Fig. 8 illustrates

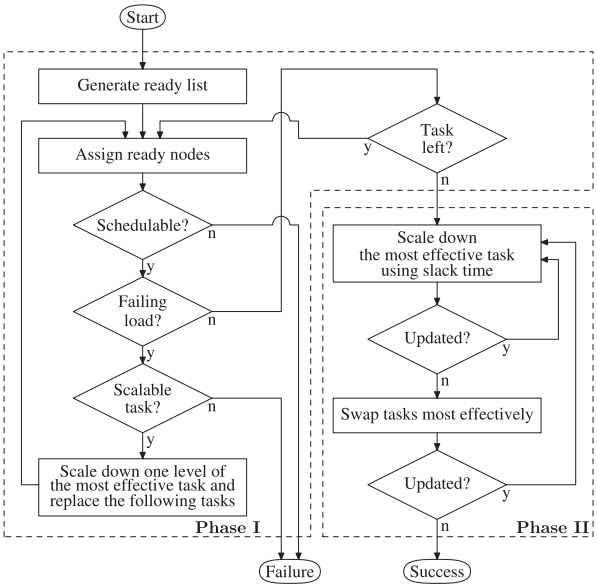


Fig. 6 Battery-aware multiprocessor voltage scheduling algorithm.

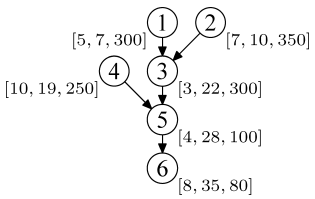


Fig. 7 Input task graph for a multiprocessor system. Each node is equipped with $[A, B, C]$: A is the execution time and B is the deadline in minutes; C is the current in mili-ampair.

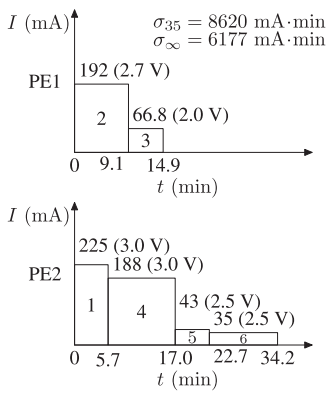


Fig. 8 Load profiles of two processors for tasks in Fig. 7. The numbers displayed in the bars are the task numbers and the numbers above them the current in mili-ampair.

the result of the proposed algorithm with the inter-processor communication delay and cost ignored. Overall, the result shows the non-increasing tendency. For this example, the proposed algorithm is as efficient as the profile produced by non-increasing algorithm [3, Fig. 13 (d)].

Table 3 Task specifications.

Task #	Name	Current (mA)
1	MPEG	208.92
2	DICT206high	186.53
3	TTS206	102.89
4	TTS74high	98.77
5	TTS74low	98.77
6	WAV206high	79.28
7	WAV59	70.71

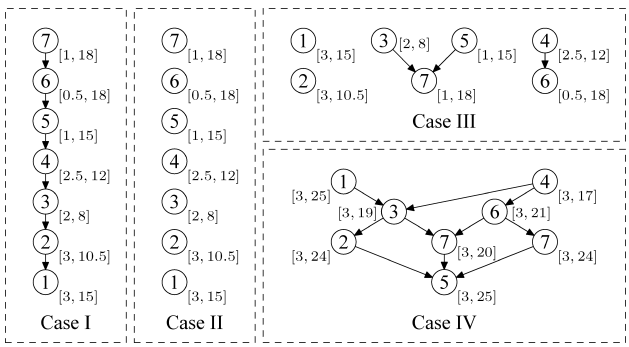


Fig. 9 Aperiodic task graphs. Each node is equipped with $[A, B]$: A is the execution time and B is the deadline in minutes.

Table 4 Loads given by the aperiodic task voltage scheduling for uniprocessor systems for tasks in Fig. 9.

Algorithm\task sets	I	II	III	IV
Exclusive down scaling	3258	3039	3039	5265
Non-increasing	3258	2834	2928	5807
Proposed algorithm	3254	2799	2800	5265

5. Experimental Results

For comparison purposes, throughout this paper, we used the same task sets as in the existing work [3]. Table 3 shows the current of the real-time applications running on ITSY [3]. The arrival time of all tasks was assumed to be time 0 to use the greatest freedom of schedulability, though our algorithms can be used under the restriction.

Aperiodic tasks on uniprocessor systems. Four task graphs are presented in Fig. 9: the strictly increasing profile (Case I), the independent tasks (Case II), and randomly generated task sets (Case III and IV). The number in the circle denotes the number of tasks specified in Table 3. Each task has a pair of duration (min) and deadline (min).

For comparison purpose, we used exclusive down scaling algorithm (*ExclusiveDownScaling*2(·) in [2, p.304]), which showed the best result among algorithms described in [2] for their example task sets, and the profile produced by non-increasing algorithm [3]. Table 4 shows that all the resulting profiles of the proposed algorithm were not worse than exclusive down scaling profiles and non-increasing profiles. The proposed algorithm was approximately 2% superior on average and 3% at the maximum.

Non-increasing algorithm did not return the result for the long profiles, *e.g.*, 160 identical tasks of task #1 (dura-

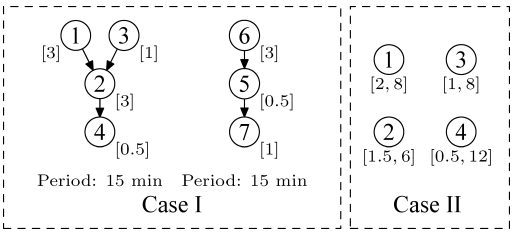


Fig. 10 Periodic task graphs. In Case I and Case II, each node is equipped with $[A]$ and $[A, B]$, respectively; A is the execution time and B is the period in minutes.

Table 5 Loads given by the periodic task voltage scheduling at the start of specified hyperperiod after the targeting hyperperiod for uniprocessor systems for tasks in Fig. 10.

Algorithm\task sets	I		II	
Observation time	1st	20th	1st	20th
Non-increasing	2013	1534	2198	1939
Proposed algorithm	1705	1234	1918	1663

Table 6 Loads given by the aperiodic task voltage scheduling for multiprocessor systems for tasks in Fig. 11.

Algorithm\task sets	I	II	III	I ×16
# of PE	2	2	2	4
Non-increasing	3764	4924	4168	33160
Proposed algorithm	3760	4603	3195	28378

tion 1 min in the highest voltage, deadline 240 min). Non-increasing algorithm cannot recover the battery exhaustion, since it does not take into account downscaling of tasks preceding failing task. The proposed algorithm, however, successfully returned the feasible profile.

Periodic tasks on uniprocessor systems. Figure 10 presents task sets of the same period dependent tasks (Case I) and the different period independent tasks (Case II). The results of scheduling were presented in Table 5. The first column of each case describes the nominal battery load of a hyperperiod at the end of this hyperperiod (at the start of the next hyperperiod) and the second column after 20 period. In all cases, our profiles are superior to non-increasing profiles, in the way that the improvement is up to 19.0%, with an average of 15.5%. Interestingly, in our result, targeting the different periods had the different voltage profiles, while in non-increasing profiles those were the same. The actual loads of our profiles were also superior to those of non-increasing profiles. Those imply that in those cases our profiles outperformed non-increasing profiles in terms of the residual nominal battery charge at any observation time when the same profiles are repeated.

Aperiodic tasks on multiprocessor systems. We used voltage scheduling algorithm described in Sect. 4.3. We assumed no time and energy overhead by interprocessor communication. Table 6 shows the objective functions of the three task sets presented in Fig. 11[†]. Since non-increasing

[†]We used the different deadline of tasks to [3], since otherwise all the tasks can be downscaled within the deadline constraints. We regarded task 8 of Case I in [3] as task 7.

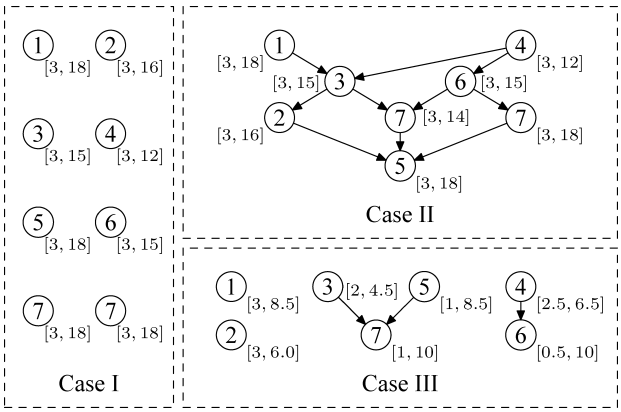


Fig. 11 Task graphs for multiprocessor systems. Each node is equipped with $[A, B]$: A is the execution time and B is the deadline in minutes.

profile in Case I is almost optimal, the difference between their and our results is limited. However, the improvement of the proposed algorithm in the independent task sets appears especially in the large task sets such as in Case IV, where set consists of 16 times of tasks from Case I with multiplying each deadline 8 times, and is processed by 4 processors. Our profiles showed 23.3% improvement at the maximum.

6. Conclusions

We have proposed static voltage scheduling algorithms for battery-powered DVS systems based on the studied battery characteristics and our analysis. The proposed algorithms are extensions to the work of Chowdhury and Chakrabarti [3] and are designed by using greedy heuristics. Periodic and aperiodic voltage scheduling on uni- and multiprocessor platforms, respectively, outperformed those in the comparative work [6], [7].

Acknowledgment

The authors wish to thank Takuya Azumi, Krzysztof Jozwik, Seiya Shibata, and Hideki Takase for providing useful comments on a draft of this paper. Part of this work was supported by CREST, JST.

References

[1] T.L. Martin, Balancing batteries, power, and performance: System issues in CPU speed-setting for mobile computing, Ph.D. Thesis, Carnegie Mellon University, 1999.
[2] D. Rakhmatov and S. Vrudhula, "Energy management for battery-powered embedded systems," ACM Trans. Embedded Computing Systems, vol.2, no.3, pp.277–324, 2003.
[3] P. Chowdhury and C. Chakrabarti, "Static task-scheduling algorithms for battery-powered DVS systems," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.13, no.2, pp.226–237, 2005.
[4] R. Rao, S. Vrudhula, and D.N. Rakhmatov, "Battery modeling for energy-aware system design," Computer, vol.36, no.12, pp.77–87, 2003.
[5] T.F. Fuller, M. Doyle, and J. Newman, "Simulation and optimization of the dual lithium ion insertion cell," J. Electrochemical Society,

vol.141, pp.1–10, 1994.
[6] D.N. Rakhmatov and S.B.K. Vrudhula, "An analytical high-level battery model for use in energy management of portable electronic systems," International Conference on Computer-Aided Design, pp.488–493, 2001.
[7] D. Rakhmatov, S. Vrudhula, and D. Wallach, "A model for battery lifetime analysis for organizing applications on a pocket computer," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.11, no.6, pp.1019–1030, 2003.
[8] P. Rong and M. Pedram, "Battery-aware power management based on Markovian decision processes," International Conference on Computer-Aided Design, pp.707–713, 2002.
[9] L. Benini, G. Castelli, A. Macii, E. Macii, M. Poncino, and R. Scarsi, "Discrete-time battery models for system-level low-power design," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.9, no.5, pp.630–640, 2001.
[10] C.F. Chiasserini and R.R. Rao, "Energy efficient battery management," IEEE J. Sel. Areas Commun., vol.19, no.7, pp.1235–1245, 2001.
[11] T.L. Martin and D.P. Siewiorek, "A power metric for mobile systems," Proc. International Symposium on Low Power Electronics and Design, pp.37–42, 1996.
[12] J. Luo and N.K. Jha, "Battery-aware static scheduling for distributed real-time embedded systems," Proc. Conference on Design Automation, pp.444–449, 2001.
[13] D. Rakhmatov, S. Vrudhula, and C. Chakrabarti, "Battery-conscious task sequencing for portable devices including voltage/clock scaling," Proc. ACM Conference on Design Automation, pp.189–194, 2002.
[14] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," International Symposium on Low Power Electronics and Design, pp.197–202, 1998.
[15] D. Shin and J. Kim, "Optimizing intratask voltage scheduling using profile and data-flow information," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.26, no.2, pp.369–385, 2007.
[16] Intel StrongARM SA-1100 Microprocessor Developer's Manual, 1999.
[17] D.N. Rakhmatov, "Battery voltage prediction for portable systems," IEEE International Symposium on Circuits and Systems, pp.4098–4101, IEEE, 2005.
[18] R. Rao and S. Vrudhula, "Battery optimization vs energy optimization: Which to choose and when?," International Conference on Computer-Aided Design, pp.439–445, 2005.
[19] H.S. Yun and J. Kim, "On energy-optimal voltage scheduling for fixed-priority hard real-time systems," ACM Trans. Embedded Computing Systems, vol.2, no.3, pp.393–430, 2003.

Tetsuo Yokoyama is an assistant professor at the Department of Software Engineering, the Faculty of Information Science and Technology, Nanzan University. He received his Ph.D. degree in Information Science and Technology from the University of Tokyo in 2006. He visited the University of Copenhagen, Department of Computer Science, supported by JSPS Research Fellowships for Young Scientists, from 2006 to 2007. From 2007 to 2009, he was a Researcher at the Center for Embedded Computing Systems, the Graduate School of Information Science, Nagoya University. His research interests include energy optimization, reversible computing, and programming languages. He is a member of ACM and JSSST.

Gang Zeng graduated from Hunan University, China, with B.E., M.E. degrees in 1993, 2001, respectively. From 1993 to 1998, he joined Hunan University where he was a lecturer in the Institute of Electric and Information Engineering. From 2001 to 2002, he joined ZTE Corporation where he was a senior engineer. He received his Ph.D. degree in information science from Chiba University in 2006. He is currently an assistant professor in the Graduate School of Information Science of Nagoya University. His research interests include power-aware computing, embedded system design, design for testability, system-on-a-chip testing. He is a member of IEEE.

Hiroyuki Tomiyama received his B.E., M.E. and D.E. degrees in computer science from Kyushu University in 1994, 1996 and 1999, respectively. From 1999 to 2001, he was a visiting postdoctoral researcher with the Center of Embedded Computer Systems, University of California, Irvine. From 2001 to 2003, he was a researcher at Institute of Systems & Information Technologies/KYUSHU. In 2003, he joined the Graduate School of Information Science, Nagoya University as an assistant professor, and became an associate professor in 2004. In 2010, he joined the College of Science and Engineering, Ritsumeikan University as a full professor. His research interests include system-level design automation, architectures and compilers for embedded systems and systems-onchip. He currently serves as an associate editor-in-chief of IPSJ Transactions on System LSI Design Methodology, an associate editor of IEEE Embedded Systems Letters, and an editorial board member of International Journal on Embedded Systems. He has also served on the organizing and program committees of several premier conferences including ICCAD, ASP-DAC, DATE, CODES+ISSS, and so on. He is a member of ACM, IEEE and IPSJ.

Hiroaki Takada is a Professor at the Department of Information Engineering, the Graduate School of Information Science, Nagoya University. He received his Ph.D. degree in Information Science from University of Tokyo in 1996. He was a Research Associate at University of Tokyo from 1989 to 1997, and was an Assistant Professor and then an Associate Professor at Toyohashi University of Technology from 1997 to 2003. His research interests include real-time operating systems, real-time scheduling theory, and embedded system design. He is a member of ACM, IEEE, IPSJ, and JSSST.