Battery-Conscious Task Sequencing for Portable Devices Including Voltage/Clock Scaling*

Daler Rakhmatov and Sarma Vrudhula
Center for Low Power Electronics
ECE Department, University of Arizona
Tucson, Arizona 85721
daler/sarma@ece.arizona.edu

Chaitali Chakrabarti
Center for Low Power Electronics
EE Department, Arizona State University
Tempe, Arizona 85287
chaitali@asu.edu

ABSTRACT

Operation of battery-powered portable systems can no longer be sustained once a battery becomes discharged. Maximization of the battery lifetime is a difficult task due to nonlinearity of battery behavior that depends on the characteristics of the system load profile. We address the problem of task sequencing without and with voltage/clock scaling that shapes the profile so that the battery lifetime is maximized. We developed an accurate analytical battery model and validated it with measurements taken on a real lithium-ion battery used in a pocket computer. We use the model as a basis for a unique battery-conscious cost function and utilize its properties to develop several novel algorithms, including insertion of recovery periods and voltage/clock scaling for delay slack distribution.

Categories and Subject Descriptors

J.6.2 [**Computer-Aided Engineering**]: Computer-Aided Design

General Terms

Algorithms, Performance, Design

Keywords

Battery, modeling, low-power design, scheduling, voltage scaling

1. INTRODUCTION

In portable battery-powered electronic systems maximizing the battery lifetime is an important problem. Consequently, a designer needs an accurate model of a battery. Previously, we have proposed a high-level analytical model that produces accurate lifetime predictions and permits a trade-off between the accuracy and the amount of computation performed [9]. The present work shows how this model can be utilized for single-processor load scheduling without and with voltage/clock scaling.

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

DAC 2002, June 10-14, 2002, New Orleans, Louisiana, USA.

Several published papers have considered the battery issues to improve system operation [2, 5, 6]. The authors of [2] used the VHDL-based simulation model from [1] to expose the impact of different dynamic power management policies on the battery lifetime. Both single-battery and dual-battery powered systems were considered. In [6] the authors addressed static scheduling of tasks with real-time constraints. Evaluation of the proposed method was based on the battery model combining Peukert's law [4] and empirical analysis due to [8]. Lifetime improvements reported in [2, 6] must be interpreted with care, since the results are heavily biased by the properties of a model used to represent the battery. Power-aware scheduling under timing constraints was described in [5]. The authors used a NASA/JPL Mars Pathfinder rover as a motivating application with the two power sources: a battery and a solar panel. The objective was to utilize the solar panel (the "free" energy source) as much as possible and minimize the energy drawn from the battery. However, the scheduler was only aware of the presence of an alternative energy source, not battery behavior itself.

In this paper, we propose an approach to battery load profile synthesis based on the high-level battery model from [9]. This model not only accurately predicts the battery lifetime under various discharge conditions, but also provides an analytical expression that can be used as a cost function to guide the battery lifetime optimization process. The specific problems that we address are (1) task sequencing without voltage/clock scaling and (2) task sequencing with voltage/clock scaling. While tackling these problems, we account for and take advantage of charge recovery effects. We also demonstrate that predictions of the model are in close correspondence with *measurements* performed on a real lithium-ion battery used in a pocket computer.

2. MOTIVATING EXAMPLE

In this section we demonstrate, by simple examples, the impact of task sequencing on a battery. Moreover, we also provide justification for the use of the battery model from [9] in our algorithms. All the results are obtained by performing experiments with a real battery.

The experimental setup consisted of a lithium-ion battery used in the ITSY pocket computer [10], the programmable electronic load Agilent 6060B, and the host computer recording experimental data. The open-circuit voltage of the battery was 4.2 V and the cutoff voltage was set to 3.0 V. The electronic load operated in the constant-current mode. Variable-current profiles were generated as a piece-wise constant-current profile (a staircase). The battery voltage was sampled every second, and once the voltage dropped below the cutoff level the load was disconnected from the battery. Recharging was performed in the constant-current mode at 800 mA, until the battery voltage recovered to its open-circuit value.

First, we conducted ten constant-current discharge experiments to estimate parameters for our battery model and to see how well the model predicts battery lifetime under constant loads. The discharge currents ranged from 1011 mA to 123 mA, and the lifetimes ranged from 30 minutes to over 300 minutes. The model fit is shown in Figure 1. The maximum error of lifetime predictions is 4%, with the average of 2%.

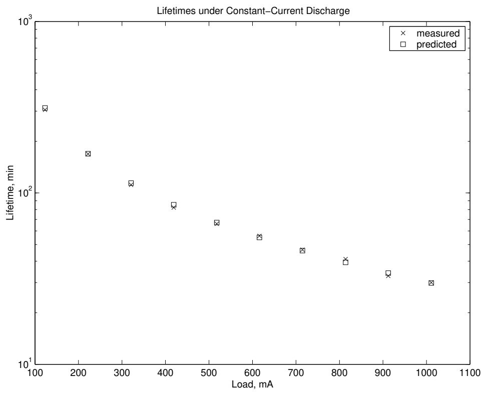


Figure 1: Lifetimes for Various Constant-Current Loads.

Next, we generated five variable-current profiles shown in Figure 2. We selected four currents of certain duration (1011 mA for 10 minutes, 814 mA for 15 minutes, 518 mA for 20 minutes, and 222 mA for 15 minutes) and sequenced them in different order to obtain profiles P1-P4. The length and total charge demand of each of the four profile is 60 minutes and 36010 mA-min, respectively. Note that in P1, after all the four currents were applied, the battery was discharged under the constant rate of 222 mA until the cutoff voltage was reached. The sole purpose of the last load 222 mA is to determine how much residual charge is left.

Profile	Measured		Predicted		Lifetime Error	Charge Error
	L_m min	C_m mA-min	L_p min	C_p mA-min	%	%
P1	64.9	37098	66.9	37542	3.1	1.2
P2	54.0	29944	54.4	30348	0.7	1.3
P3	55.8	32591	55.0	31940	1.4	2.0
P4	58.4	35181	57.5	34715	1.5	1.3
P5	67.5	34965	67.0	34706	0.7	0.7

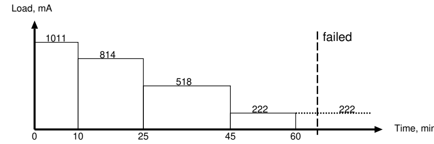
Table 1: ITSY Lifetime and Delivered Charge.

According to these experiments, profile P1 is the best, and profile P2 is the worst sequence for the battery. Note that load currents in P1 (P2) are decreasing (increasing). This important result is accurately predicted by the battery model. In P1 the battery survives all the four loads and remains operational for extra 4.9 minutes under 222 mA (residual 1088 mA-min charge). In P2, however, the battery fails to service the last 6.0 minutes under 1011 mA (undelivered 6066 mA-min charge). For these two profiles, the difference in the total delivered charge is as much as 20% of 36010 mA-min. The other alternatives P3 and P4 are neither better than P1 nor worse than P2, as predicted by the model and demonstrated by the measurements.

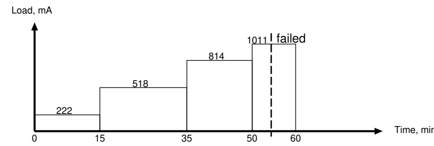
Voltage/clock scaling plays a critical role in minimizing energy consumption, thus improving the battery lifetime. To obtain profile P5, for example, we took P2 and changed the failing 10-minute load of 1011 mA to a 20-minute load of 518 mA to reflect a hypothetical change in the system voltage.[1] The profile length increased from 60 minutes to 70 minutes, and the battery failed after 67.5 minutes. The total delivered charge is 34965 mA-min, which is an improvement over P2 with 29944 mA-min.

Table 1 shows the measured/predicted lifetimes (L_m and L_p, respectively) and the measured/predicted delivered charges (C_m and C_p, respectively) for the profiles P1-P5. Note that the charge errors were within 2%, while the maximum lifetime error was 3%.

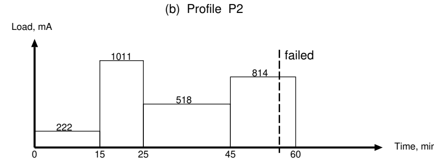
[1]This is a pessimistic scenario, since the charge consumption is usually reduced after the supply voltage is scaled down.



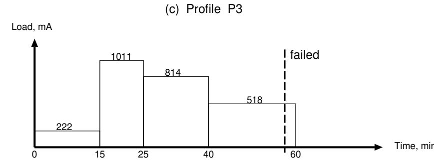
(a) Profile P1



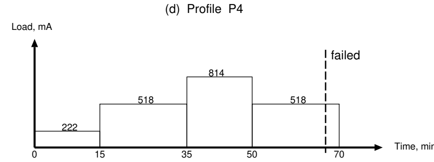
(b) Profile P2



(c) Profile P3



(d) Profile P4



(e) Profile P5

Figure 2: Experimental Load Profiles for ITSY Battery.

One can see that the model adequately captures the trend in battery behavior observed in the experiments, with very small prediction errors.

3. COST FUNCTION

In this section we describe the input and the output of the battery model and define a cost function for the task sequencing procedures. Given n tasks, we associate a 3-tuple with each task k: the current I_k drawn from the battery, the duration Δ_k, and the start time t_k.

Battery Model: The input to the model is the *load profile*, described by the following three sets: the set of currents $S_I = \{I_k \mid k = 0,1,...,n-1\}$, the set of durations $S_\Delta = \{\Delta_k \mid k = 0,1,...,n-1\}$, and the set of start times $S_t = \{t_k \mid k = 0,1,...,n-1\}$. The profile always starts at time 0. Let T denote the profile length, i.e. the latest finish time, $T = max_k\{t_k + \Delta_k\}$.

In addition to S_I, S_Δ, and S_t, the model requires specifications of two battery parameters α and β. These parameters are estimated based on the battery profiling data collected from several constant-current discharge experiments. For instance, the ITSY battery has $\alpha = 39668$ and $\beta = 0.57$. The units of α are mA-min, and the units of β^2 are 1/min. Intuitively, α represents the battery *capacity*, and β captures *nonlinearity* in battery behavior.

Let

$$F(x,y,z,\beta) = z - y + 2\sum_{m=1}^{\infty} \frac{e^{-\beta^2 m^2 (x-z)} - e^{-\beta^2 m^2 (x-y)}}{\beta^2 m^2}. \quad (1)$$

The output of the model is the battery lifetime L – the smallest positive number in the interval $[0,T]$ that satisfies the following equation:

$$\alpha = \sum_{k=0}^{n-1} I_k F(L, min\{L, t_k\}, min\{L, t_k + \Delta_k\}, \beta). \quad (2)$$

It is possible that such a number does not exist, i.e. the battery survives the profile (all tasks are completed). In such cases, $L =$ **NULL**. Procedure $ComputeLifetime(\cdot)$ is used for computing L. Further details are given in [9].

Cost Function: For a given profile of length T, let

$$\sigma = \sum_{k=0}^{n-1} I_k F(T, t_k, t_k + \Delta_k, \beta). \quad (3)$$

Intuitively, σ is the charge that the battery has lost by time T. If $\sigma < \alpha$, then the battery is still operational at time T. We use σ as our cost function to be *minimized*.

For a valid profile, the latency T must not exceed a given delay budget B. In addition the battery may not fail at any time $t \le T$, i.e.

$$\alpha \ge \sum_{k=0}^{n-1} I_k F(t, min\{t, t_k\}, min\{t, t_k + \Delta_k\}, \beta) \ , \ \forall \ t \le T. \quad (4)$$

Note that, if for some load k its start time $t_k \ge t$, then it does not contribute to the value of the sum. Condition (4) is necessary to account for relaxation effects that may mask a failure taking place before T. It requires that no sub-profile of length $t \le T$ is too extreme for the battery to survive.

The following property of the function $F(\cdot)$ is critical to many of the results in this paper.

THEOREM 1. *For $0 \le \Delta \le t + \Delta \le T$, function $F(T, t, t + \Delta, \beta)$ is (a) monotonically increasing in t and (b) monotonically decreasing in T.*

Remark: Note that our battery model does not involve battery voltage. The cutoff voltage (i.e. the voltage at which the battery is declared to be discharged) is abstracted to α, and a user is expected to specify the load profile as a discrete set of current values drawn from the battery during application execution. Sampling the battery current at the frequency of less than 1Hz will suffice [7]. If such information is not available, a user may assume the average battery voltage V_{ave}, and let the load profile $i(t) = P(t)/(\varepsilon V_{ave})$, where ε is the efficiency of a DC-DC converter and $P(t)$ is the power consumption of an application. One can assume that ε is constant and $P(t) \propto V^2(t) f(t)$ [3], where $V(t)$ is the converter output voltage and $f(t)$ is the clock rate. Then $i(t) \propto V^2(t) f(t)$.

4. TASK SEQUENCING (NO SCALING)

The formulation of the battery-aware task sequencing problem without voltage/clock scaling is shown in Figure 3. The input consists of sets S_I and S_Δ, a directed acyclic task graph $\vec{G}$, the delay budget B, and the battery parameters α and β. The output is the set S_t of start times of tasks. The objective is to minimize σ given by Equation (3), subject to the following three constraints:
(1) **dependency constraint:** task dependencies are preserved,
(2) **delay constraint:** the latency does not exceed the delay budget,
(3) **endurance constraint:** at any time within a schedule, the battery is alive.

We tackle the sequencing problem in three steps: (i) greedy sequencing, (ii) incremental recovering, and (iii) local compressing. The first step is *greedy sequencing*, subject to dependencies, with no idling allowed. The length of the resulting profile is $T = \sum_{k=0}^{n-1} \Delta_k$. In this case, the delay constraint is already satisfied, under the assumption that B is never less than the sum of all task durations. However, the endurance constraint may be violated, which calls for the second step – *incremental recovering*. In the second step, we insert idle periods so that the profile no longer fails. As much recovery as necessary is exercised, and the load ordering is not changed. Thus, the endurance constraint and dependencies are not violated; however, the delay budget may be exceeded due to recovery delay penalty. In the third step, *local compressing*, we attempt to place light loads inside idle periods, subject to dependencies, so that load relaxation is still present but recovery delay penalty is reduced. Thus, the purpose of the third step is to reduce the profile length T to the budget B. Details of these three procedures are shown in Figure 3.

Greedy Sequencing: The input is the task graph $\vec{G}$ as well as S_I and S_Δ, and the output is S_t. Dependencies are preserved by construction: no task is scheduled unless its predecessors have already been scheduled. If there are several candidates competing for the current place in a sequence, then the procedure selects the one with the heaviest weight among them. Once a task is scheduled, it is removed from the task graph.

The weight of a node is determined as follows. Let a subgraph induced by some node p be denoted by $\vec{G}_p$, and let the function $Mean(\vec{G}_p)$ return the mean current of the nodes in $\vec{G}_p$. The weight $w(p)$ is defined as the maximum of I_p and $Mean(\vec{G}_p)$. Such weighing is preferred over $w(p) = I_p$ for each task p (e.g. a task with very low current may have a successor with very large current, whose execution will be delayed until its predecessor is scheduled).

If there are no dependencies, then $w(p)$ equals to I_p for each node p. In this case, the procedure sequences tasks in nonincreasing order of their currents. Such a profile has the minimum cost σ (for $T = \sum_{k=0}^{n-1} \Delta_k$), in accordance with Theorem 2.

THEOREM 2. *For a task sequence with no idle periods allowed, $\sum_{k=0}^{n-1} I_k F(T, t_k, t_k + \Delta_k, \beta)$ is minimized, if $I_i \ge I_j \Rightarrow t_i \le t_j$ for all $0 \le i, j \le n-1$.*

Incremental Recovering: The input is an initial profile specified by S_I, S_Δ, and S_t (the output of the greedy sequencing step) as well as the battery parameters α and β. The output is the task sequence with inserted idle periods and a flag indicating whether recovering failed or succeeded. Idle periods are inserted only if the battery fails before time T; otherwise, the load profile is returned as it is.

Let δ denote the length of an idle period. Before computing δ, the procedure checks whether δ will be less than the delay budget B. This check is performed as follows. Let q denote the failing task, i.e. $L \in [t_q, t_q + \Delta_q]$. First, the procedure assumes that $\delta = B$ and creates temporary set $\hat{S}_t$, where start times $\hat{t}_k$ of each task k following and including q are increased by B (e.g. $\hat{t}_q = t_q + B$). Then, it checks whether task q is recovered, i.e. $L \notin [\hat{t}_q, \hat{t}_q + \Delta_q]$. If $\delta \ge B$, then the procedure terminates with **FAILURE**, since it will be impossible to compress the profile so that the delay constraint is met. If δ is guaranteed to be less than B, then it is computed by $MinRecoveryLength(\cdot)$.[2] Finally, $InsertRecovery(\cdot)$ places the idle period of length δ at t_q and adds δ to t_q and to the start times of tasks following q in the sequence. This process is repeated until either all failures are repaired or recovery is no longer feasible.

At the absence of the delay constraint, the proposed approach provides an optimal failure recovering solution for a given load profile, in accordance with Theorem 3.

THEOREM 3. *Let q be the failing task, and let σ_q denote the cost of the subprofile of length $t_q + \Delta_q$. Let δ denote the length of an idle period inserted within the subprofile in question (the length of the resulting subprofile becomes $t_q + \Delta_q + \delta$). (1) The cost of the new subprofile is minimized, if an idle period is inserted at t_q. (2) If an idle period is placed earlier than t_q, its length must be greater than δ to achieve the same cost as in (1).*

Local Compressing: The input is the task graph $\vec{G}$, the recovered load profile (specified by S_I, S_Δ, and S_t), the delay budget B,

[2]Note that we set the upper bound on δ to be B. One (naive) implementation of $MinRecoveryLength(\cdot)$ would be to increase δ from 0 to B with small time increments, say τ, and check whether L is still in $[t_q + \delta, t_q + \Delta_q + \delta]$. (After inserting an idle period the start time of q is shifted by δ). In other words, one needs to increase δ until the cost of the subprofile of length $t_q + \Delta_q + \delta$ becomes less than α.

PROBLEM: Battery-Conscious Task Sequencing
INPUT: S_I, S_Δ, $\vec{G}$, B, α, β
OUTPUT: S_t
OBJECTIVE: $min\{\sigma = \sum_{k=0}^{n-1} I_k F(T, t_k, t_k + \Delta_k, \beta)\}$
CONSTRAINTS:
(1) if q depends on p in $\vec{G}$, then $t_q \geq t_p + \Delta_p$
(2) $T \leq B$
(3) $\forall\ t \leq T,\ \sum_{k=0}^{n-1} I_k F(t, min\{t, t_k\}, min\{t, t_k + \Delta_k\}, \beta) \leq \alpha$

TaskSequenceConstruction $(\vec{G}, S_I, S_\Delta)$
$S_t = GreedySequencing(\vec{G}, S_I, S_\Delta)$
$\{S_t, flag\} = IncrementalRecovering(S_I, S_\Delta, S_t, \alpha, \beta)$
IF $flag =$ **SUCCESS**
$\{S_t, flag\} = LocalCompressing(\vec{G}, S_I, S_\Delta, S_t, \alpha, \beta)$
RETURN $\{S_t, flag\}$

GreedySequencing $(\vec{G}, S_I, S_\Delta)$
FOR all $p \in \vec{G}$
$w(p) = max\{I_p,\ Mean(\vec{G}_p)\}$
$now = 0$
WHILE $\vec{G}$ is not empty
$S = \{p \mid Parents(p, \vec{G}) = \emptyset\}$
Select $q \mid w(q) = max\{w(p) \mid p \in S\}$
$t_q = now,\ \ now = t_q + \Delta_q$
Remove q from $\vec{G}$
RETURN S_t

IncrementalRecovering $(S_I, S_\Delta, S_t, \alpha, \beta)$
$L = ComputeLifetime(S_I, S_\Delta, S_t, \alpha, \beta)$
WHILE $L \neq$ **NULL**
Find $q \mid L \in [t_q, t_q + \Delta_q]$
$\hat{S}_t = InsertRecovery(B, t_q, S_t)$
$L = ComputeLifetime(S_I, S_\Delta, \hat{S}_t, \alpha, \beta)$
IF $L \in [\hat{t}_q, \hat{t}_q + \Delta_q]$
RETURN $\{S_t$, **FAILURE**$\}$
$\delta = MinRecoveryLength(\alpha, \beta, q, S_I, S_\Delta, S_t)$
$S_t = InsertRecovery(\delta, t_q, S_t)$
$L = ComputeLifetime(S_I, S_\Delta, S_t, \alpha, \beta)$
RETURN $\{S_t$, **SUCCESS**$\}$

LocalCompressing $(\vec{G}, S_I, S_\Delta, S_t, B, \alpha, \beta)$
$T = max_k\{t_k + \Delta_k\}$
IF $B \geq T$
RETURN $\{S_t$, **SUCCESS** $\}$
$T_{new} = T,\ \ change =$ **TRUE**
WHILE $change =$ **TRUE**
$\{R_s, R_f\} = IdentifyIdlePeriods(S_I, S_\Delta, S_t)$
$M = |R_s|,\ \ \hat{S}_t = S_t,\ \ T_{old} = T_{new},\ \ change =$ **FALSE**
WHILE $M > 0$
$\hat{S} = \{p \mid t_p \geq R_f[M]\}$
Find $q \mid I_q = min\{I_p \mid p \in \hat{S}$ **AND** $Parents(p, \vec{G}) \notin \hat{S}\}$
Remove q from $\hat{S}$
$t_q = R_s[M]$
$S_t = UpdateStartTimes(S_t)$
$\{S_t, flag\} = IncrementalRecovery(S_I, S_\Delta, S_t, \alpha, \beta)$
IF $flag =$ **FAILURE**
$S_t = \hat{S}_t$
ELSE
$T_{new} = max_k\{t_k + \Delta_k\}$
IF $B \geq T_{new}$
RETURN $\{S_t$, **SUCCESS** $\}$
IF $T_{old} \leq T_{new}$
$S_t = \hat{S}_t$
ELSE
$change =$ **TRUE**
BREAK
$M = M - 1$
RETURN $\{S_t$, **FAILURE** $\}$

Figure 3: Task Sequencing without Voltage/Clock Scaling.

and the battery parameters α and β. The output is the new profile specification and the flag indicating whether compression succeeded or failed.

First, the length T of the original profile is compared with the budget B. If the profile length is within the budget, then the profile is not changed. Otherwise, we identify idle periods and construct two sets R_s (idle start times) and R_f (idle finish times). For example, the start time of the k-th idle period (counting from the beginning of the profile) is recorded as $R_s[k]$, and its finish time is recorded as $R_f[k]$. Next, we select the latest unvisited idle period M and collect the tasks scheduled after M into the set $\hat{S}$. In $\hat{S}$, we find task q such that I_q is as low as possible provided that starting q at $R_s[M]$ will not violate dependencies. Once q is placed at $R_s[M]$, the start times of the remaining tasks in $\hat{S}$ are updated by $UpdateStartTimes(\cdot)$, so that their initial ordering is preserved, but initial idle periods among them are omitted. Then, new idle periods are inserted, if necessary. If the length of this new profile is decreased, then it becomes the current solution, and the outer **WHILE**-loop is repeated. Otherwise, profile changes are ignored, and the next latest unvisited idle period is considered (the inner **WHILE**-loop is repeated).

Local compressing is terminated, if (a) changes can no longer reduce the length of the current profile, or (b) the current profile meets the delay budget. Note that modifications are applied starting from the end of the load profile, since the lightest loads are normally scheduled late. Light load may reduce the total recovery time if placed inside an idle period of interest. We consider idle periods starting from the latest so that the minimal portion of the profile is changed. In other words, the procedure places lighter tasks into later idle periods, thus attempting to approach the deadline at a minimal cost penalty.

5. TASK SEQUENCING WITH SCALING

Simultaneous voltage/clock scaling is a very effective way to reduce energy consumption [3]. From the battery lifetime maximization perspective, scaling techniques are critical in shaping the load profile appropriately. In this section we state the problem of task sequencing with voltage/clock scaling and provide two procedures for improving delay-constrained load profiles. Note that the battery model involves only task durations and currents drawn from the energy source by a DC-DC converter. Since voltage/clock scaling is applied to the energy consumer (the output side of a DC-DC converter), changes in voltage levels and clock frequencies must be translated into changes in the task currents and durations before the model can be used.

We assume that the system can operate at any voltage V_i from the ordered set $S_V = (V_0, V_1, ..., V_{K-1})$ at the corresponding maximum frequency ϕ_i from the ordered set $S_\phi = (\phi_0, \phi_1, ..., \phi_{K-1})$. The elements of S_V and S_ϕ are in ascending order. Let I_{ik} and Δ_{ik} respectively denote the current and the duration of task k, when the system is operating at V_i with the clock frequency ϕ_i. Figure 4 shows the formulation of the battery-conscious voltage/clock scaling problem. The objective is to assign each task k to a specific voltage V_i at the frequency ϕ_i (thus determining the task current I_{ik} and the task duration Δ_{ik}) and the start time t_k, so that the resulting profile cost σ is minimized and the following constraints are satisfied: (1) task dependencies are preserved, (2) the profile length is within the delay budget, and (3) the battery survives all the tasks. Unlike the case of task sequencing without voltage/clock scaling, we need to generate not only S_t, but also S_I and S_Δ.

Scaling Based on Lowest-Power Initial Solution: We start with the lowest-power profile, which satisfies the endurance constraint, but may violate the delay constraint. In this profile all the tasks are assigned to V_0 (the lowest supply voltage level), so that the energy consumption is minimized. We use $LookUp(\cdot)$ to look up, in a user-specified table, task currents and durations for a given voltage. For example, for task k at V_0 and ϕ_0, the current and duration are $\{I_k, \Delta_k\} = LookUp(k, V_0, \phi_0) = \{I_{0k}, \Delta_{0k}\}$. Thus, we generate the initial sets S_I and S_Δ. Initial S_t is produced by $GreedySequencing(\cdot)$.

Let L be the computed lifetime of the initial profile $\{S_I, S_\Delta, S_t\}$.

PROBLEM: Battery-Conscious Voltage/Clock Scaling
INPUT: S_V, S_ϕ, $\vec{G}$, B, α, β
OUTPUT: S_I, S_Δ, S_t
OBJECTIVE: $min\{\sigma = \sum_{k=0}^{n-1} I_{ik} F(T, t_k, t_k + \Delta_{ik}, \beta)\}$
CONSTRAINTS:
(1) if q depends on p in $\vec{G}$, then $t_q \geq t_p + \Delta_p$
(2) $T \leq B$
(3) $\forall\, t \leq T,\ \sum_{k=0}^{n-1} I_{ik} F(t, min\{t, t_k\}, min\{t, t_k + \Delta_{ik}\}, \beta) \leq \alpha$

VoltageClockScaling $(S_V, S_\phi, \vec{G}, B, \alpha, \beta)$
$K = |S_V|, \quad n = |\vec{G}|$
FOR $k = 0, 1, ..., n-1$
  $\{I_k, \Delta_k\} = LookUp(k, V_0, \phi_0)$
$S_t = GreedySequencing(\vec{G}, S_I, S_\Delta)$
$T = \sum_{k=0}^{n-1} \Delta_k$
$L = ComputeLifetime(S_I, S_\Delta, S_t, \alpha, \beta)$
IF $T \leq B$ **AND** $L =$ **NULL**
  RETURN $\{S_I, S_\Delta, S_t,$ **SUCCESS**$\}$
IF $T > B$ **AND** $L \neq$ **NULL**
  RETURN $\{S_I, S_\Delta, S_t,$ **FAILURE**$\}$
IF $T \leq B$ **AND** $L \neq$ **NULL**
  $\{S_t, flag\} = IncrementalRecovering(S_I, S_\Delta, S_t, \alpha, \beta)$
  IF $flag =$ **SUCCESS**
    $\{S_t, flag\} = LocalCompressing(\vec{G}, S_I, S_\Delta, S_t, \alpha, \beta)$
  RETURN $\{S_I, S_\Delta, S_t, flag\}$
$\{S_I, S_\Delta, S_t, flag\} = LatencyReduction(S_V, S_\phi, S_I, S_\Delta, S_t, B, \alpha, \beta)$
IF $flag =$ **SUCCESS**
  $\{S_I, S_\Delta, S_t\} = SlackUtilization(S_V, S_\phi, S_I, S_\Delta, S_t, B, \alpha, \beta)$
RETURN $\{S_I, S_\Delta, S_t, flag\}$

LatencyReduction $(S_V, S_\phi, S_I, S_\Delta, S_t, B, \alpha, \beta)$
FOR $x = 0, 1, ..., n-1$
  $k =$ the x-th task in the sequence
  FOR $i = 1, ..., K-1$
    $\{\hat{I}_k, \hat{\Delta}_k\} = LookUp(k, V_i, \phi_i)$
    $\{\hat{S}_I, \hat{S}_\Delta, \hat{S}_t\} = UpdateLoadProfile(\hat{I}_k, \hat{\Delta}_k, S_I, S_\Delta, S_t)$
    $T = max_l\{\hat{t}_l + \hat{\Delta}_l\}$
    $L = ComputeLifetime(\hat{S}_I, \hat{S}_\Delta, \hat{S}_t, \alpha, \beta)$
    IF $L \neq$ **NULL**
      BREAK
    $\{S_I, S_\Delta, S_t\} = \{\hat{S}_I, \hat{S}_\Delta, \hat{S}_t\}$
    IF $T \leq B$
      RETURN $\{S_I, S_\Delta, S_t,$ **SUCCESS**$\}$
RETURN $\{S_I, S_\Delta, S_t,$ **FAILURE**$\}$

SlackUtilization $(S_V, S_\phi, S_I, S_\Delta, S_t, B, \alpha, \beta)$
$n = |S_I|$
$T = max_l\{t_l + \Delta_l\}$
IF $T \geq B$
  RETURN $\{S_I, S_\Delta, S_t\}$
FOR $x = n-1, n-2, ..., 0$
  $k =$ the x-th task in the sequence
  Find $j \mid \{I_k, \Delta_k\} = LookUp(k, V_j, \phi_j)$
  FOR $i = j-1, j-2, ..., 0$
    $\{\hat{I}_k, \hat{\Delta}_k\} = LookUp(k, V_i, \phi_i)$
    $\{\hat{S}_I, \hat{S}_\Delta, \hat{S}_t\} = UpdateLoadProfile(\hat{I}_k, \hat{\Delta}_k, S_I, S_\Delta, S_t)$
    $T = max_l\{\hat{t}_l + \hat{\Delta}_l\}$
    IF $T > B$
      BREAK
    $\{S_I, S_\Delta, S_t\} = \{\hat{S}_I, \hat{S}_\Delta, \hat{S}_t\}$
RETURN $\{S_I, S_\Delta, S_t\}$

Figure 4: Task Sequencing with Voltage/Clock Scaling.

Note that $T = \sum_{k=0}^{n-1} \Delta_{0k}$. If $T \leq B$ and $L =$ **NULL**, then we have already obtained final solution. On the other hand, if $T > B$ and $L \neq$ **NULL**, then we abort our attempts to generate a feasible profile. There are two non-trivial cases: (a) $T \leq B$ and $L \neq$ **NULL**, and (b) $T > B$ and $L =$ **NULL**.

In case (a), the delay budget is met, but the battery does not survive the profile. Since the voltage level is the lowest, the task currents cannot be reduced. Therefore, only idle period insertion is applicable: we perform $IncrementalRecovering(\cdot)$, followed by $LocalCompressing(\cdot)$.

In case (b), the battery survives the profile, but the delay budget is exceeded. Therefore, some tasks must be assigned to a higher voltage/clock (resulting in greater currents but shorter durations) to satisfy the delay constraint. We use $LatencyReduction(\cdot)$ to reduce the profile length so that the budget is met. Figure 4 provides further details. At each iteration of the outer **FOR**-loop, we consider the earliest unvisited task k (at position x in the task sequence), and its voltage is scaled up as much as possible, provided that the profile remains failure-free. Upon assigning new voltage/clock to k, we call $UpdateLoadProfile(\cdot)$ to update the load profile, move to the next task (at position $x+1$ in the task sequence), and perform voltage/clock up-scaling again. Intuitively, this procedure attempts to generate a nonincreasing sequence of currents without causing failures before all the tasks are completed.

Slack Utilization: Given a failure-free profile with positive delay slack (i.e. $L =$ **NULL** and $T < B$), we may be able to improve its quality by scaling voltage/clock down, which decreases task currents but increases task durations. The corresponding procedure, $SlackUtilization(\cdot)$, is shown in Figure 4. We consider tasks one by one, starting from the profile's end. The voltage/clock for each considered task is down-scaled as much as possible, provided that the delay constraint is not violated. Thus, this procedure attempts to reduce the battery stress toward the end of the discharge process. Voltage/clock down-scaling does not introduce any failures, in accordance with the following theorem.

THEOREM 4. *Assume that a given task sequence is failure-free. Let voltage and clock be scaled down for some tasks. The resulting profile is still failure-free.*

6. RESULTS

The battery lifetime depends on both the load profile and the battery parameters. In this section we let $\alpha = 40000$ and $\beta = 0.2$. Figure 5 shows the lifetime surface for the simplest case of a constant-current discharge. Note that for sufficiently large β the battery behaves as a practically ideal power source, i.e. the lifetime increases almost linearly as α grows. We use small $\beta = 0.2$ to expose nonlinear effects in battery behavior.

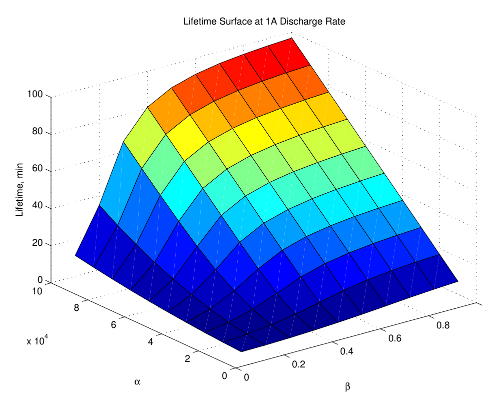


Figure 5: Lifetimes for Various α and β under Constant-Rate Discharge.

Consider a set of tasks described in Table 2. There are two supply voltages V and $V/2$, and each task is associated with the two possible battery current values and the two possible durations. We

Task	Voltage V		Voltage $V/2$		Parents
	I, mA	Δ, min	I, mA	Δ, min	
T1	1000	5	125	10	-
T2	750	5	93	10	-
T3	500	10	62	20	T1
T4	250	10	31	20	-
T5	800	5	100	10	T2, T3
T6	600	5	75	10	T4, T5
T7	400	10	50	20	T1
T8	200	10	25	20	T2, T7

Table 2: Task Currents and Durations.

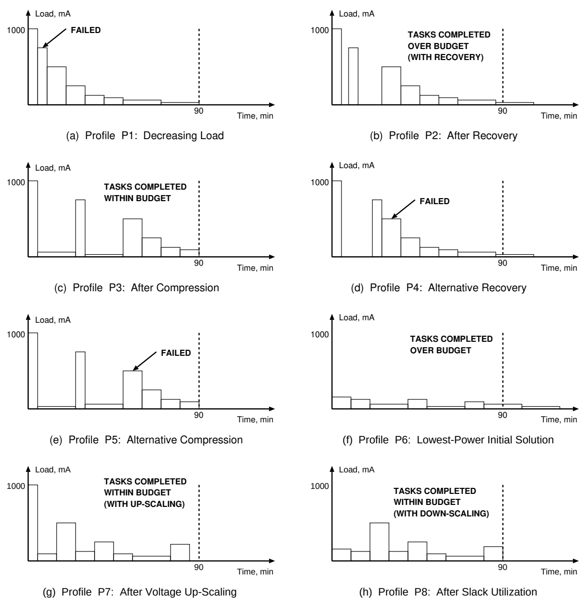


Figure 6: Generated Load Profiles.

let the delay budget B equal 90 minutes. To illustrate the proposed methods, we examine eight profiles P1-P8, shown in Figure 6. Table 3 shows the task sequence as well as the corresponding length T and the cost σ, for each profile.

Profile	Task Sequence	T, min	σ @ T, mA-min
P1	(T1, T2, T3, T4, T5, T6, T7, T8)	90	23435
P2	(T1, T2, T3, T4, T5, T6, T7, T8)	106	23180
P3	(T1, T7, T2, T8, T3, T4, T5, T6)	90	29558
P4	(T1, T2, T3, T4, T5, T6, T7, T8)	106	23292
P5	(T1, T8, T2, T7, T3, T4, T5, T6)	90	29646
P6	(T1, T2, T3, T5, T4, T6, T7, T8)	120	9886
P7	(T1, T2, T3, T5, T4, T6, T7, T8)	85	30139
P8	(T1, T2, T3, T5, T4, T6, T7, T8)	90	26103

Table 3: Profile Lengths and Costs.

To demonstrate the impact of task sequencing without voltage/clock scaling, we fix the voltage at V for tasks T1-T4 and at $V/2$ for tasks T5-T8. In this case, the minimum profile length is 90 minutes, which the same as the delay budget B. Profile P1 is generated by $GreedySequencing(\cdot)$, profile P2 is obtained from P1 by $IncrementalRecovering(\cdot)$, and finally, P2 is transformed into profile P3 by $LocalCompressing(\cdot)$.

Dependencies allow the tasks to be sequenced in decreasing order of their currents in P1. The battery, however, fails after 8.6 minutes; therefore, idle period insertion follows. Since there is no delay slack, P2 with two idle periods violates the delay constraint by 16 minutes.[3] However, the profile length of P3 is successfully reduced to 90 minutes by compression. Note that P1 and P3 are two task sequences with no idle periods, and as expected, the cost of P1 is lower than the cost of P3 (23435 vs. 29558 mA-min). Profile P1 demonstrates the need for the endurance constraint. For P1 $\sigma < \alpha$ but $L \neq$ **NULL**, which means that a battery failure has been masked by relaxation effects.

Profile P4 shows an alternative to P2. The total idling time in P2 and P4 is the same, 16 minutes. In P4, however, the two idle periods from P2 are combined into one. Note that the battery can no longer survive P5, and as expected, the cost of P2 is lower than the cost of P4. Profile P5 shows an alternative to P3. P5 differs from P3 in that tasks T7 and T8 are swapped. Even though the costs of P3 and P5 are practically the same, P5 violates both the dependency constraint and the endurance constraint.

Profiles P6, P7, and P8 demonstrate the impact of voltage/clock scaling. P7 is obtained by $LatencyReduction(\cdot)$ from the lowest-power initial solution P6, where all the tasks are assigned to $V/2$. The length of P6 is 120 minutes, and the delay slack is –30 minutes. In profile P7, which is failure-free and within the delay budget, the voltage for tasks T1, T3, T4, and T8 is scaled up to V. Note that a 5-minute delay slack is available in P7. After applying $SlackUtilization(\cdot)$ one obtains profile P8, whose cost is lower than that of P7 by more than 10% (26103 vs. 30139). Note that among the eight considered profiles, only P3, P7, and P8 are feasible (i.e. all the constraints are satisfied), and the cost of P8 is the best.

7. CONCLUSION

Since the battery behavior exhibits a nonlinear dependency on the characteristics of the system load profile, achieving a provably optimal lifetime is not an easy task. We addressed the problem of battery-conscious task sequencing without and with voltage/clock scaling. We employed an accurate analytical battery model as a basis for a unique cost function and developed several efficient heuristics for synthesizing battery-efficient load profiles. Our algorithms include insertion of recovery periods and voltage/clock scaling for delay slack distribution.

8. REFERENCES

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

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

[3] T. Burd and R. Brodersen. *Energy Efficient Microprocessor Design*. Kluwer, Boston, 2002.

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

[5] J. Liu, P. Chou, N. Bagherzadeh, and F. Kurdahi. Power-aware scheduling under timing constraints for mission-critical embedded systems. *Proc. DAC*, 2001.

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

[7] T. Martin. *Balancing Batteries, Power, and Performance: System Issues in CPU Speed-Setting for Mobile Computing*. Ph.D. Dissertation, Carnegie Melon University, 1999.

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

[9] D. Rakhmatov, S. Vrudhula, and D. Wallach. A model for battery lifetime analysis for organizing applications on a pocket computer. *To appear IEEE Trans. VLSI Systems*.

[10] M. Viredaz and D. Wallach. Power evaluation of a handheld computer: a case study. *Compaq WRL Research Report 2001/1*, May 2001.

[3] The first idle period lasts 3 minutes, and the second idle period lasts 13 minutes.