Battery-aware Static Scheduling for Distributed Real-time Embedded Systems

Jiong Luo and Niraj K. Jha
Department of Electrical Engineering
Princeton University, Princeton, NJ, 08544
{jiongluo, jha}@ee.princeton.edu

Abstract

This paper addresses battery-aware static scheduling in battery-powered distributed real-time embedded systems. As suggested by previous work, reducing the discharge current level and shaping its distribution are essential for extending the battery lifespan. We propose two battery-aware static scheduling schemes. The first one optimizes the discharge power profile in order to maximize the utilization of the battery capacity. The second one targets distributed systems composed of voltage-scalable processing elements (PEs). It performs variable-voltage scheduling via efficient slack time re-allocation, which helps reduce the average discharge power consumption as well as flatten the discharge power profile. Both schemes guarantee the hard real-time constraints and precedence relationships in the real-time distributed embedded system specification. Based on previous work, we develop a battery lifespan evaluation metric which is aware of the shape of the discharge power profile. Our experimental results show that the battery lifespan can be increased by up to 29% by optimizing the discharge power file alone. Our variable-voltage scheme increases the battery lifespan by up to 76% over the non-voltage-scalable scheme and by up to 56% over the variable-voltage scheme without slack-time re-allocation.

1. Introduction

Battery-powered portable systems have been widely used in many applications, such as mobile computing, wireless communications, information appliances, wearable computing as well as various industrial and military applications. As systems become more complex and incorporate more functionality, they become more power-hungry. Thus, reducing energy consumption and extending battery lifespan have become a critical aspect of designing battery-powered systems.

High-performance battery-powered distributed embedded systems are generally composed of a network of heterogeneous processing elements (PEs). The PEs can be general-purpose processors, application-specific integrated circuits, field programmable gate arrays or analog circuits. The input specifications of such systems are typically in the form of task graphs. A task graph is a directed acyclic graph in which each node is associated with a task and each edge is associated with the amount of data that must be transferred between the two connected tasks. The period associated with a task graph indicates the time interval after which it executes again. A hard deadline, the time by which the task associated with the node must complete its execution, exists for every sink node and some intermediate nodes. All the hard deadlines must be met. The embedded system can be a multi-rate system, i.e., it may contain multiple tasks graphs with different periods. The goal of real-time scheduling algorithms is to guarantee the deadlines of periodic task graphs while honoring the precedence relationship among tasks. Due to the importance of energy in battery-powered systems, the scheduling scheme should be energy-aware and battery-efficient as well.

Many system-level power optimization techniques have been presented in the literature. The representative work includes voltage scaling [9,10,11], which refers to varying the speed of a processor by changing the clock frequency along with the supply voltage, and power management, which refers to the use of power-down modes when a processor or device is idle in order to reduce power consumption [7,8]. Instead of focusing on reducing power consumption alone, researchers have begun to study the battery behavior and the effect of the battery discharge pattern on the battery capacity as well [1,2,5,6].

This paper addresses the issue of battery-aware variable-voltage scheduling for multi-rate real-time distributed embedded systems. The goal of our scheduling algorithm is to extend the battery lifespan while meeting the hard real-time constraints and precedence relationships among tasks. The scheduling algorithm is able to vary the voltage of PEs that are voltage scalable in order to reduce the power consumption, and manage the power profile of the whole system in order to achieve improved battery efficiency. Our work is motivated by the ideas presented in [2,5], which suggest that reducing the discharge current level and shaping its distribution are essential for reducing the battery capacity loss. The reduction of the average discharge current level is achieved through voltage scaling and PE shutoff. The optimization of the discharge current profile is achieved through a series of schedule transformations starting from an initially valid schedule. The schedule transformations aim to shape the discharge current profile to improve the utilization of the ideal battery capacity, while maintaining the validity of the original schedule. Our work has several contributions: (1) We simultaneously address the issues of optimizing the overall power consumption profile of the distributed embedded system to improve the battery efficiency, and guaranteeing the hard real-time constraints and precedence relationships which are traditional tasks in real-time distributed scheduling. This has not been done in any previous work. (2) For distributed embedded systems consisting of voltage-scalable PEs, we perform variable-voltage scheduling via efficient slack time allocation, which helps reduce the average discharge power consumption as well as flatten the discharge power profile, while still guaranteeing the hard real-time constraints and precedence relationships. Therefore, the scheme is very powerful in maximizing the battery lifespan.

2. Battery Behavior Models

The capacity of a battery cell can be defined in terms of ampere-hours or watt-hours [4]. Many factors influence the performance

Acknowledgment: This work was supported by DARPA under contract no. DAAB07-00-C-L516 .

DAC 2001, June 18-22, 2001, Las Vegas, Nevada, USA.

characteristics and the actual capacity that can be drawn from the battery. Normally, the battery capacity decreases as the discharge current increases. Fig. 1 shows the curve of battery capacity versus the discharge current, i.e., the discharge rate. The load current is represented as the value normalized to the battery's rated capacity.

The work in [5] explores the fact that battery efficiency is influenced by the average discharge current as well as the average discharge current profile. They define the actual power drawn out of the battery as

$$p^{act} = \int (V * I / c(I)) * \hat{P}(I) dI \qquad (1)$$

where I is the average discharge current for some period of time. $\hat{P}(I)$ is the probability density function of I. V is the discharge voltage and is assumed to be fixed. $c(I)$ is the utilization factor, which is the ratio of the battery capacity (in terms of watt-hours) at discharge current I to the ideal battery capacity CPA_0. Hence, it can represent the battery efficiency compared to the ideal condition. The duration of battery service life should equal CPA_0 divided by p^{act}. This work shows that even under the constraint that the average power consumption is the same, i.e., $p_{ave} = V * I_{ave} = \int V * I * \hat{P}(I) dI$ is constant, different discharge current distributions still lead to different p^{act}. The maximum battery life is achieved when the variance of the discharge current distribution is minimized. Their results are supported by experimental study based on PSPICE simulations.

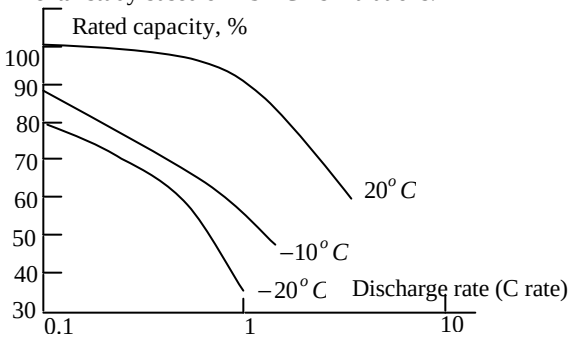


Fig. 1 Performance of $C/LiNiO_2$ Lithium-Ion AA-size cell at various temperatures and discharge rates

The work in [2] studies the effect of intermittent discharges on the capacity of Lithium rechargeable batteries and demonstrates that peak power predicts battery capacity better than average power. The work in [6] employs a cycle-accurate battery model and evaluates the instantaneous battery capacity on a cycle-by-cycle basis. The battery recovery effect in communication devices is studied in [1].

3. Motivational Examples

This section presents two examples that motivate our work in this paper. We use Equation (1) to evaluate the actual power p^{act} drawn from the battery. If the battery cell voltage is assumed to be nearly constant, the relationship between the battery capacity and the discharge current would hold for discharge power as well. In this section, we use Peukert's formula [4], an empirical equation to evaluate the relationship between the battery capacity and the discharge current

$$c(I) = k / I^a \qquad (2)$$

where k and a are constants. We assume $a = 0.5$.

Example 1: Fig. 2 gives an embedded system specification consisting of three task graphs. Assume for simplicity that all these have a period of 16.0 seconds. The embedded distributed system implementing the task graphs consists of two PEs, PE1 and PE2, connected by a bus. Figs. 3 and 4 give two feasible schedules for one period. The worst-case execution time of *t1*, *t3*, *t4*, *t5*, *t6*, *t7* and *t8* on their allocated PE are all 4 seconds, while the worst-case execution time of *t2* on its allocated PE is 2 seconds. The execution time of inter-PE communication edge *e1* on the bus is also 2 seconds. The average power consumption number for each scheduled event is shown in brackets in the schedule, e.g., for *t1* it is 5 units. Based on the traditional assumption in distributed computing, we assume intra-PE communications, *e2*, *e3*, *e4* and *e5*, all take zero time. We assume both PE1 and PE2 are buffered.

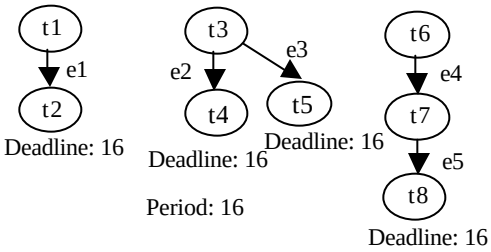


Fig. 2 Task graphs for Example 1

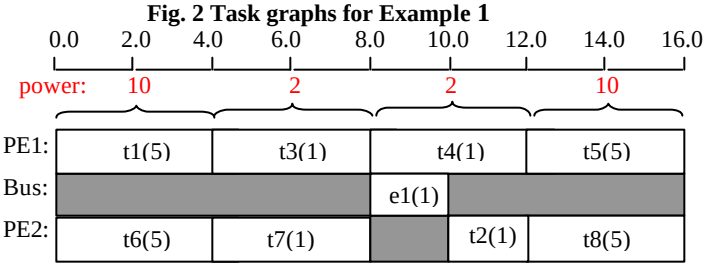


Fig. 3 Original valid schedule for Example 1

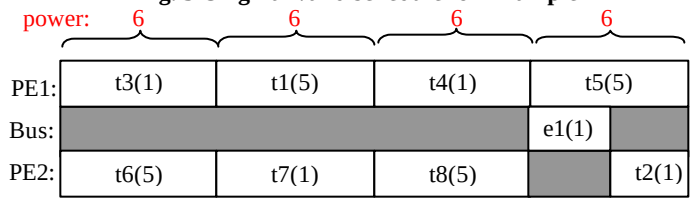


Fig. 4 New valid schedule for Example 1

For simplicity, we assume that the power consumption in the shut-off state (shaded parts in the schedule) is zero and that there is no overhead in entering and leaving this state. Note, that our algorithm, which is presented later, does not need to make the above assumptions. The overall discharge power of the system is the summation of all the power consumptions in all the PEs and buses. For the schedule in Fig. 3, the discharge power distribution is approximately $\hat{P}(p = 10) = 1/2$ and $\hat{P}(p = 2) = 1/2$, while for the schedule in Fig. 4, the discharge power distribution is $\hat{P}(p = 6) = 1$. Using Equations (1) and (2), the actual power drawn from the battery in the schedule in Fig. 3 is 17.23**c*, while the value for the schedule in Fig. 4 is 14.70**c*, where *c* is some constant. The latter schedule results in a 15% reduction in the actual power drawn out of the battery, and correspondingly a 17% improvement in the battery lifespan. □

Example 2 below is used to illustrate the effect of voltage scaling in real-time distributed embedded systems composed of voltage-scalable PEs. The relationships among clock period, supply voltage and power consumption, which is used in this example to calculate power consumption, are presented next.

The processor clock period, T, can be expressed in terms of the supply voltage, V_{dd}, and threshold voltage, V_t, as follows:

$$T = kV_{dd} / (V_{dd} - V_t)^2 \qquad (3)$$

where k is a constant. We assume V_t = 0.8V. The processor power, p, can be expressed in terms of the frequency, f, switched capacitance, N, and the supply voltage, V_{dd}, as:

$$p = \frac{1}{2} fNV_{dd}^2 \qquad (4)$$

Example 2: Consider the task graphs shown in Fig. 5. Fig. 6(a) gives an as-soon-as-possible feasible static schedule on a distributed system consisting of PEs, PE1 and PE2, connected by a bus. Assume a power supply voltage of 3.3V. The worst-case execution time of *t1*, *t3*, *t4*, *t5* and *t7* on their allocated PE are all 0.2 seconds. The worst-case execution time of *t2* and *t6* on their allocated PE are both 0.3 seconds. The execution time of inter-PE communication edges *e1* and *e2* are both 0.1 seconds. We assume the average power consumption for each task is 1 unit, while the average power consumption for each inter-PE communication edge is 0.2 unit. Fig. 7(a) gives a new feasible schedule after schedule slots interchanging and shifting of the schedule in Fig. 6(a).

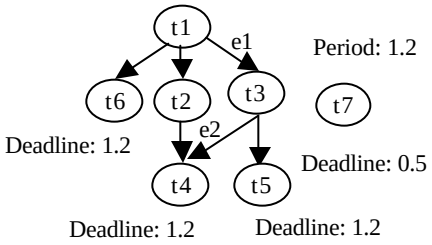
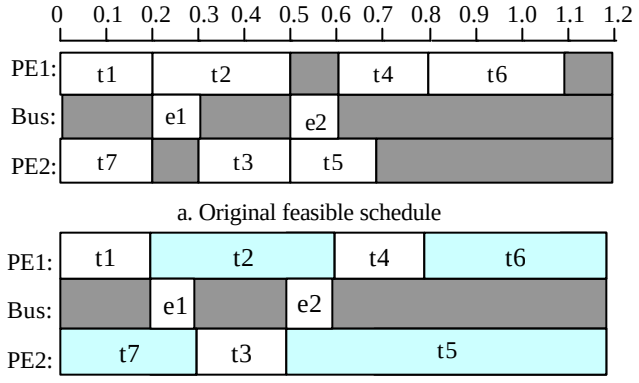


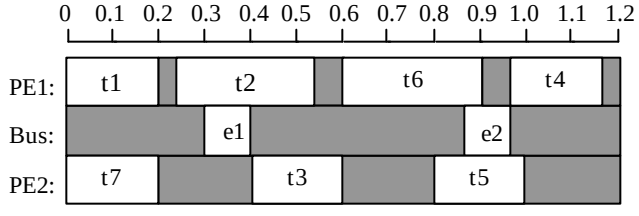
Fig. 5 Task graphs for Example 2



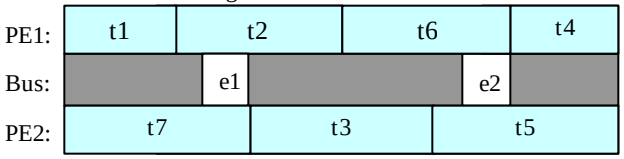
a. Original feasible schedule

b. Corresponding variable-voltage schedule

Fig. 6 Original schedule and the corresponding variable-voltage schedule for Example 2



a. Original feasible schedule



b. Corresponding variable-voltage schedule

Fig. 7 New Schedule after schedule slot shifting and swapping and the corresponding variable-voltage schedule

We perform voltage scaling on these two schedules by extending the execution time of the tasks to their latest finish time. The new schedules are shown in Fig. 6(b) and Fig. 7(b), respectively. For example, in Fig. 6(a), *t2* is scheduled at time instant 0.2. Since it can finish as late as time instant 0.6, the speed of PE1 can be scaled down by a ratio of (0.6 – 0.2) / 0.3 for *t2*. Correspondingly, the supply voltage can be scaled down from 3.3V to 2.8V, extending the actual running length of *t2* from 0.3 to 0.4. In Fig. 6(b), the working voltages for task *t1*, *t2*, *t3*, *t4*, *t5*, *t6* and *t7* are 3.3, 2.8, 3,3, 3.3, 1.8, 2.8, and 2.7V, respectively. In Fig. 7(b), the working voltage for tasks *t1*, *t2*, *t4* and *t6* are all 3V, while for task *t3*, *t5* and *t7* are all 2.3V. The performance metrics for the different schedules, including the average power consumption and battery service life evaluated by Equation (2) using average power consumption, are shown in Table 1. In Table 1, c' is some constant.

Table 1: Performance characteristics of different schedules

Schedule	Overall average power consumption of the system	Service life evaluated based on average power consumption
Fig. 6(a)	1.37	0.62 * c'
Fig. 6(b)	1.05	0.93 * c'
Fig. 7(b)	0.96	1.06 * c'

Compared to the schedule in Fig. 6(a), the schedule in Fig. 6(b) results in a 23% reduction in average system power consumption and a 50% improvement in battery service life evaluated based on average power consumption. For the schedule in Fig. 7(b), there is a 30% reduction in the average system power consumption and a 71% improvement in battery service life evaluated based on average power consumption, compared to the schedule in Fig. 6(a). This example shows, not suprisingly, that voltage scaling reduces system power consumption and increases the battery lifespan. Moreover, a more efficient voltage scaling scheme can lead to better results, as the difference between Fig. 6(b) and Fig. 7(b) shows. □

4. Static Resource Allocation, Assignment and Scheduling

The static resource allocation, task/communication assignment and scheduling algorithms we use are from a system synthesis tool presented in [12]. It uses a slack-based list scheduling algorithm to generate static PE and communication link schedules for each task and communication event along the hyperperiod, which is the least common multiple of all the task graph periods in a multi-rate system specification. It is well known that there exists a feasible schedule for the periodic task graphs if and only if there exists a feasible schedule for the hyperperiod [15]. A slack-based list scheduling scheme is used in the inner-loop of system synthesis in order to generate a cost-efficient distributed architecture and a feasible schedule. The scheduling scheme is not optimized for battery-aware power consumption. We modify the static schedule in a post-processing stage through a series of schedule transformations, which we discuss in Sections 5 and 6.

5. Battery-aware Scheduling Scheme

In this section, we present a battery-efficient scheduling scheme which aims to optimize the system discharge power profile. Heuristics to optimize the battery efficiency, as suggested in Section 3, are based on minimization of the peak power consumption and reduction of the variance of the discharge current profile. The goal of our scheduling scheme is to reduce the overall average of the actual power drawn out of the battery, p^{act}, which is evaluated by

$$p^{act} = \frac{1}{hyperperiod} \int_0^{hyperperiod} \frac{p(t)}{c_p(t)} dt \qquad (5)$$

where $p(t)$ is the power consumption at time t, and $c_p(t)$ is the battery utilization factor evaluated at time t. Note that Equation (5) is just a variation of Equation (1). $p(t)$ is the summation of all the power consumptions in all the PEs and buses, or any other system component which draws power from the battery. Thus, we assume $p(t) = \sum_{i \in (all\ PEs \cup all\ buses)} p_i(t)$. Other components of system power consumption can be easily incorporated as well, which normally can be represented as a fixed contribution. For each task, we assume we know its average power consumption and its worst-case execution time through simulation and analysis tools [16,17]. The energy consumption of a PE in the idle period ip of a system entering sleep state i can be modeled as $EC_i = e_i * p_{e_i} + w_i * p_{w_i} + (ip - e_i - w_i) * p_i$ [8], where e_i (w_i) is the delay overhead and p_{e_i} (p_{w_i}) is the power consumption in entering (leaving) sleep state i, and p_i is the power consumption in this state. A PE always assumes a sleep state that minimizes *EC*.

First, we define some variables and functions that are used later in presenting our heuristics. We define *event_list* as a list of statically scheduled events in the order of their start times on each PE or bus for one hyperperiod. *sched* is an array of *event_list* for all the PEs and buses. The scheduled event can be a periodic task or a communication event. In the static schedule, every event is characterized by a *start* time, a *finish* time, and a *duration*, which is the worst-case execution time for that event. For a scheduled event, *next_event* is the next scheduled event in the same *event_list*. For a task, *in-edges* (*out-edges*) refers to all the *inter-PE* communication edges entering (coming out of) the task, where *inter-PE* communication edges refer to those edges for which the parent task and child task are assigned to different PEs. A *deadline* may be associated with a task. For a task i,

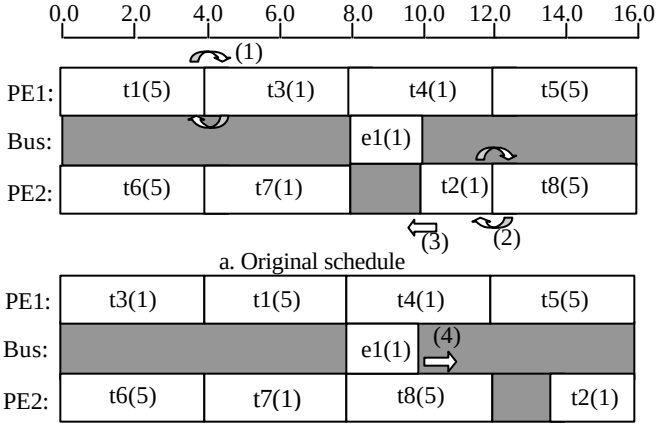
$$finish\_constraint(i) = \min(\min_{j \in out-edges(i)} j \rightarrow start, i \rightarrow deadline, (i \rightarrow next\_event) \rightarrow start)$$

The battery-aware schedule optimization scheme is composed of two parts. The initial schedule is first optimized through global shifting with a goal to reduce the peak power consumption and to increase the flexibility in the schedule. Then local schedule transformations are employed to optimize the discharge power profile. The details of the scheduling scheme are presented in sub-sections 5.1 and 5.2.

5.1 Battery-aware local schedule transformations

In the battery-aware local schedule transformation scheme, we first rank the time point along the hyperperiod in the order of $p(t)$. Then from the highest power consumption time point to the lowest point, we try to interchange adjacent events or shift forward or shift backward events around that time point, with a goal to reduce cost function p^{act} evaluated by Equation (5). In order to guarantee the validity of the schedule in each transformation, if interchanging two scheduled events i and j, or shifting forward a scheduled event i, or shifting backward a scheduled event j violates the precedence relationship, we evaluate the possibility of shifting forward the *out-edges* of i and/or shifting backward the *in-edges* of j for exactly the amount needed in case i and j are tasks, or shifting forward the child task of i and/or shifting backward the parent task of j exactly for the amount needed in case i and j are communication events, and take into consideration these effects on p^{act} as well. No local schedule transformation is performed if it violates the precedence relationship or hard timing constraints, or it does not reduce p^{act}. After each round of transformations, the power profile is re-ranked and the above process repeats until *sched* is no longer changed. The following example illustrates the scheme.

Example 3 Consider the task graphs in Fig. 2 and the initial schedule in Fig. 3 once again. Fig. 8 illustrates the steps involved in applying the above-mentioned method to the schedule in Fig. 3. The ranking of time periods in terms of power profile $p(t)$ initially is {(0,4),(12,16),(4,8),(8,10),(10,12)}. There are four steps involved. In the first step, *t1* and *t3* are interchanged to reduce the power consumption in time period (0,4). Similarly, in the second step, *t2* and *t8* are interchanged. Then *t8* is shifted backward to deal with time period (10,12). The resulting schedule is shown in Fig. 8(b). The ranking of time periods in terms of $p(t)$ is then updated. In the second round, *e1* is shifted forward to relax the current peak power consumption in time period (8, 10). The resulting schedule is shown in Fig. 4. At each step, p^{act} is reduced.



a. Original schedule

b. New schedule after first three steps

Fig. 8 The schedule transformation steps for the task graphs in Fig. 2

5.2 Global shifting scheme

The above local transformation scheme is greedy, and is dependent upon a good initial solution. This can be illustrated through Example 4.

Example 4: Fig. 9 shows an embedded system specification consisting of three task graphs. Fig. 10(a) gives a feasible static schedule on a distributed system consisting of two PEs, PE1 and PE2, connected by a bus. The worst-case execution time of *t1*, *t2*, *t3*, *t4*, *t5* and *t6* on their allocated PE are all 2 seconds, while the worst-case execution time of *t7* and *t8* on their allocated PE are both 1 second. The execution time of inter-PE communication edge *e1* on the bus is 1 second. The power consumption number for each scheduled event is shown in brackets in the schedule. There is no opportunity for local movements in order to reduce p^{act} in the schedule of Fig. 10(a). However, the schedule is not optimal for battery efficiency.

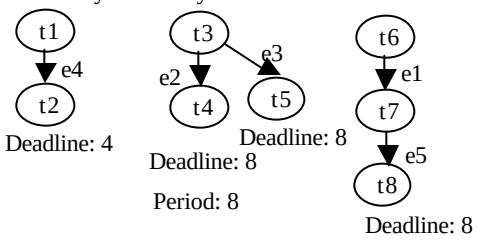
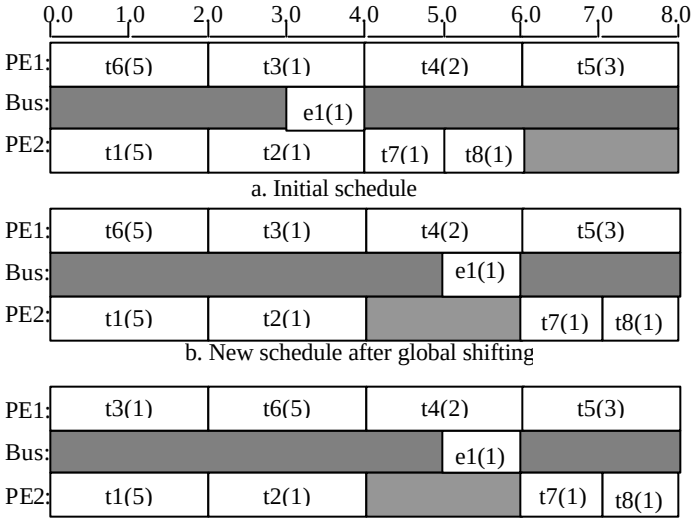


Fig. 9 Task graphs for Example 4

In order to get a good initial solution, we process the schedule through a battery-aware global shifting stage, which tries to shift the schedule slots in a global manner with the goal of reducing the peak power consumption and increasing the flexibility in the

schedule. This process starts from an initial schedule where every scheduled event is shifted backward to its as early as possible position. Then we create an event processing queue and initialize it by inserting the last event on every *event_list* which does not have *out-going* communication edges. Then we try to shift the tasks and communication events in the processing queue as late as possible, so long as in the new position where the tasks and communication events are shifted to, the overall average power consumption for that duration does not exceed some given threshold value (*power_threshold*), while the negative effect, if any, resulting from the changing of the grouping of idle periods, are less than some threshold value (*side_effect_threshold*). If there is no such position, we shift forward the scheduled events to the best position in terms of the reduction in p^{act}. A new task or communication event is added into the processing queue if its *next_event* and all those events which have data dependency on it have finished shifting. Shifting forward as late as possible helps increase the flexibility of the overall schedule so that more opportunities can be opened up for further schedule transformation.

The global shifting scheme is illustrated through Example 4. In the initial schedule in Fig. 10(a), there are no valid local movements possible to reduce p^{act}. We take the average power consumption (4.675) as the *power_threshold* and assume the *side_effect_threshold* is zero. During global shifting, first, *t8* is shifted to the as-late-as-possible slot. The average power consumption for the new time period (7, 8) of *t8* is 4, hence, the *power_threshold* is not exceeded. Similarly, *t7* is shifted as late as possible to time period (6,7). Then *e1* is shifted as late as possible to (5,6). After this global shifting procedure, the new schedule is shown in Fig. 10(b). Now *t6* and *t3* can be interchanged to reduce p^{act}, without violating the precedence relationships and hard real-time constraints. The final schedule is shown in Fig. 10(c). Compared to the schedule in Fig. 10(a), the variance of the discharge power profile is reduced in the schedule in Fig. 10(c). □



a. Initial schedule

b. New schedule after global shifting

c. Final schedule after schedule interchanging

Fig. 10 Battery-aware optimization for Example 4

6. Variable-voltage Scheduling Scheme

Some embedded systems may be composed of voltage-scalable PEs, for example, Crusoe processors [14]. Since voltage scaling has a high potential for reducing system energy consumption, our algorithm is tuned to facilitate the possibility of scaling down the voltage for each task whenever possible.

We define *slack* time for each scheduled task as the difference between its *finish_constraint* and its *finish* time. The slack time in the distributed schedule makes it possible to scale down the voltage without sacrificing the real-time constraint. Our scheduling scheme tries to allocate the slack time in a close-to optimal way to improve the performance of the consequent voltage scaling. Assume for each task *i*, d_i is its execution time plus its slack time, e_i is its execution time, and p_i is its power consumption under maximum voltage $V_{\max}$. For a PE, *total_slack* is the summation of the *slack* times of all the tasks on that PE in the initial schedule, and *total_duration* is the summation of the execution times of all the tasks on that PE. We use *total_slack* to approximate the total available *slack* time for all the tasks. Using Equations (3) and (4) to evaluate the effects of voltage scaling, for a task *i*, the speed reduction ratio should be $scale_i = d_i / e_i$, and the corresponding working voltage should be $V_i = (V_t + \frac{e_i}{d_i * b}) + \sqrt{(\frac{e_i}{d_i * b} + V_t)^2 - V_t^2}$, where $b = 2 * \frac{V_{\max}}{(V_{\max} - V_t)^2}$. Our objective is to minimize the energy consumption of all the tasks after voltage scaling, which is

$$energy = \sum_{i \in all\ tasks} p_i * (d_i / scale_i) * (V_i^2 / V_{\max}^2) = \sum p_i * e_i * V_i^2 / V_{\max}^2 \quad (6)$$

under the constraint

$$\begin{cases} \sum d_i = total = total\_duration + total\_slack \\ d_i \geq e_i \end{cases}$$

If the threshold voltage V_t is close to zero, the optimal solution can be approximated by $d_i = total * \frac{e_i \sqrt[3]{p_i}}{\sum e_i \sqrt[3]{p_i}}$, so long as $d_i \geq e_i \ \forall i$.

The allocation of slack time is performed through global schedule shifting and schedule slots interchanging to match the optimal slack assignment, which is $d_i - e_i$ for task *i*.

7. Experimental Results

In this section, we present the experimental results. The task graphs in our example are generated with the aid of a randomized task graph generator, TGFF [13].

In the first experiment, we evaluate the performance of our battery-aware scheduling scheme presented in Section 5. The actual power consumption drawn out of the battery is evaluated by Equation (5), where $c_p(t)$ is evaluated using the short-term average power consumption. The duration of the short-term average should match the order of the battery's time constant for response to the change of the discharge rate, which is assumed to be 1 second [2]. The evaluation of the battery efficiency is based on data extracted from the specifications for Lithium-Ion Polymer batteries in [3]. We evaluate two test sets, *a* and *b*, based on the same four task graphs. For the purpose of evaluation, we set the rated battery capacity (in terms of W-hours) for test set *a*(*b*) to be 2X(1.67X) of the average power consumption of the system. The results for the original schedule and the schedule optimized in terms of the discharge power profile are compared in Table 2. In Table 2, p^{ave} is the average power consumption of the system. The optimized schedule results in an improvement of battery lifespan in the range of 8.5% to 16.6% and 12.6% to 28.8% for test sets *a* and *b*, respectively. This experiment shows that without sacrificing the performance constraints and introducing overheads into the system, the shaping of the discharge power profile alone can help boost the battery performance effectively. The optimization scheme would be more powerful under stringent discharge conditions, for example, at lower temperatures or limited battery capacity, where the battery capacity loss is more pronounced when the discharge rate is high. As shown for test set

b, as the rated battery capacity decreases compared to test set *a*, the optimization of the battery discharge power profile is more effective in increasing battery performance.

In the second experiment, we evaluate the performance of our variable-voltage scheduling scheme presented in Section 6. We compare three schemes: (1) non-variable-voltage scheme, (2) variable-voltage scheduling without slack time re-allocation, and (3) variable-voltage scheduling with slack time re-allocation. We evaluate both the battery performance with and without considering the shape of the discharge power profile. The experimental results are shown in Table 3 for another set of task graphs. Scheme (3) achieves an average power reduction in the range of 17% to 38% and 14% to 31% compared to Scheme (1) and Scheme (2), respectively. In terms of the battery lifespan evaluated using p^{act}, Scheme (3) results in an improvement in the range of 26% to 76% and 20% to 56% over Scheme (1) and Scheme (2), respectively. In terms of the battery lifespan computed as the battery capacity (evaluated using average power consumption) divided by average power consumption, Scheme (3) results in an improvement in the range of 23% to 68% and 17% to 50% over Scheme (1) and Scheme (2), respectively. It can be observed that the improvement is more pronounced when the shape of the discharge power profile is taken into consideration, i.e., the evaluation is based on p^{act}, which indicates our scheme is helpful in reducing both the average discharge power level and its variance. Thus, the scheme is very powerful in boosting the battery performance.

Table 2: Comparison of different scheduling schemes for battery-aware power consumption

Test	#tasks	# PEs / # buses	p^{ave} (mW)	p^{act} (mW) / Battery lifespan(hours)		Battery lifespan increase (%)
				Non-optimized	Optimized	
1(a)	71	2/1	126	164/1.53	152/1.66	8.5%
2(a)	114	8/16	361	445/1.62	409/1.77	9.3%
3(a)	94	6/13	359	463/1.55	420/1.71	10.3%
4(a)	146	6/13	537	738/1.45	636/1.69	16.6%
1(b)	71	2/1	126	185/1.14	159/1.32	15.8%
2(b)	114	8/16	361	476/1.27	420/1.43	12.6%
3(b)	94	6/13	359	508/1.18	439/1.36	15.3%
4(b)	146	6/13	537	857/1.04	669/1.34	28.8%

8. Conclusions

In this paper, we presented two schemes to optimize the battery lifespan in battery-powered real-time embedded distributed systems by reducing the average discharge power profile and shaping its distribution. One scheme optimizes the discharge power profile. Another scheme performs variable-voltage scheduling via efficient slack-time re-allocation in the distributed system composed of voltage-scalable PEs. It helps reduce the average discharge power consumption as well as minimize the variance of the discharge power profile. Both schemes increase the battery lifespan while still guaranteeing the real-time constraints and precedence relationships in the distributed embedded system, based on an evaluation metric which is aware of the shape of the discharge power profile. In future work, the evaluation metric should incorporate the battery recovery effect as well.

References

[1] C.F. Chiasserini and R.R. Rao, "Pulse battery discharge in communication devices," in *Proc. Mobilcom*, pp. 88-95, Aug. 1999.

[2] T. Martin, "Balancing batteries, power and performance: System issues in CPU speed-setting for mobile computing," Ph.D. Dissertation, Carnegie Mellon University, Department of Electrical and Computer Engineering, Aug. 1999.

[3]http://www.batteryeng.com/lithium_ion_fs.htm

[4] H. D. Linden, *Handbook of Batteries*, 2nd ed., McGraw-Hill, New York, 1995.

[5] M. Pedram and Q. Wu, "Design considerations for battery-powered electronics," in *Proc. Design Automation Conf.*, pp. 861-866, June 1999.

[6] T. Simunic, L Benini and G. De Micheli, "Energy efficient design of battery powered embedded systems," in *Proc. Int. Symp. Low Power Electronics and Design*, pp. 212-217, Aug. 1999.

[7] Q. Qiu and M. Pedram, "Dynamic power management based on continuous-time Markov decision processes," in *Proc. Design Automation Conf.*, pp. 555-561, June 1999.

[8] E. Y. Chung, L. Benini, and G. De Micheli, "Dynamic power management using adaptive learning tree," in *Proc. Int. Conf. Computer-Aided Design*, pp. 274-279, Nov. 1999.

[9] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power optimization of variable-voltage core-based systems," *IEEE Trans. Computer-Aided Design*, vol. 18, no. 12, pp. 1702-1714, Dec. 1999.

[10] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," in *Proc. Design Automation Conf.*, pp. 134-139, June 1999.

[11] J. Luo and N. K. Jha, "Power-conscious joint scheduling of periodic task Graphs and aperiodic tasks in distributed real-time embedded systems," in *Proc. Int. Conf. Computer-Aided Design*, pp. 357-364, Nov. 2000.

[12] R. P. Dick and N. K. Jha, "MOCSYN: Multiobjective core-based single-chip system synthesis," in *Proc. Design Automation & Test in Europe Conf.*, pp. 263-270, Mar. 1999.

[13] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in *Proc. Int. Workshop Hardware/Software Codesign*, pp. 97-101, Mar. 1998.

[14] http://www.transmeta.com/

[15] E. L. Lawler and C. U. Martel, "Scheduling periodically occurring tasks on multiple processors," *Information Processing Letters*, vol. 7, pp. 9-12, Feb. 1981.

[16] Y. S. Li, S. Malik, and A.Wolfe, "Performance estimation of embedded software with instruction cache modeling," in *Proc. Int. Conf. Computer-Aided Design*, pp. 380-387, Nov. 1995.

[17] W. Ye, N. Vijaykrishan, M. Kandemir, and M. J. Irwin, " The design and use of SimplePower: A cycle-accurate energy estimation tool," in *Proc. Design Automation Conf.*, pp. 340-345, June 2000.

Table 3: Comparison of different voltage-scaling and non-voltage-scaling schemes

Test	#tasks	#PEs/ #buses	p^{act} (mW) / Battery lifespan evaluated by p^{act} (hours)			Battery lifespan (evaluated by p^{act}) increase (%)		Ave. power consumption (mW) / Battery lifespan evaluated by average power consumption (hours)		
			(1)	(2)	(3)	(3) vs. (1)	(3) vs. (2)	(1)	(2)	(3)
1	52	2/1	136/1.57	111/1.94	86/2.51	59.9%	29.4%	107/1.8	92/2.14	72/2.77
2	101	4/6	500/1.70	475/1.79	398/2.14	25.9%	19.6%	425/1.8	406/1.89	351/2.22
3	114	8/16	476/1.74	418/1.98	338/2.45	40.8%	23.7%	413/1.8	369/2.04	307/2.49
4	100	5/10	302/1.72	274/1.89	206/2.52	46.5%	33.3%	260/1.8	240/1.96	187/2.58
5	133	7/16	490/1.70	434/1.92	278/2.99	75.9%	55.7%	416/1.8	375/2.02	258/3.03