Minimal Energy Fixed-Priority Scheduling for Variable Voltage Processors

Gang Quan, and Xiaobo Sharon Hu, *Senior Member, IEEE*,

Abstract— **To fully exploit the benefit of variable voltage processors, voltage schedules must be designed in the context of work load requirement. In this paper, we present an approach to finding the least-energy voltage schedule for executing real-time jobs on such a processor according to a fixed priority, preemptive policy. The significance of our approach is that the theoretical limit in terms of energy saving for such systems is established, which can thus serve as the standard to evaluate the performance of various heuristic approaches. Two algorithms for deriving the optimal voltage schedule are provided. The first one explores fundamental properties of voltage schedules while the second one builds on the first one to further reduce the computational cost. Experimental results are shown to compare the results of this paper with previous ones.**

Index Terms — **Real-time systems, low power, scheduling, fixed-priority, dynamic voltage scaling.**

I. Introduction

LOW power design is an important design issue for designing economic and safe real-time embedded systems and has been tackled in many different ways, e.g. [1], [2]. Since real-time systems usually have a time-varying computation load, to appropriately modulate the system capability accordingly without (greatly) sacrificing the system performance has been a major strategy to achieve low power in such systems. Recent advance in VLSI techniques [3]–[6] has made the *variable voltage (speed) processor* possible. For such a processor, its frequency and supply voltage can be varied dynamically. Commercial examples of such processors include the Intel's XScale [7], Transmeta's Crusoe [8], and AMD's Duron [9]. Judicious use of these processors in the designs can greatly reduce the energy consumption of the system. Over the past several years, many scheduling techniques to minimize energy for such systems has been published, e.g., [1], [2], [10]. Yet how to achieve the best energy efficiency for many of these systems remains unknown, and how close these approaches are to the optimal solutions is still an open question.

Power-reduction scheduling techniques in general can be classified into two categories [11]: *dynamic* and *static*. Dynamic techniques are generally easy to implement and apply during run time. Examples of such techniques include [12]–[24]. Due to its inherent uncertainty and lack of complete knowledge about the timing constraints, no strong optimality results have been proven with these techniques. In [18], several dynamic voltage scheduling algorithms are proposed for real-time systems containing both periodic tasks and sporadic tasks, whose arrival times are completely unknown. These approaches are based on the optimal voltage scheduling algorithm presented in [25] and the optimal acceptance test in [26]. They are optimal in the sense that the voltage schedule leads to the lowest energy consumption for the periodic tasks and sporadic tasks which pass the acceptance test. However, these approaches cannot be easily extended to handle the cases where tasks have predefined and fixed priorities, or where the timing information of the sporadic tasks are already known. In [12], a stochastic control approach is proposed. It models the requests of real-time tasks and the state changes of the system components as a discrete-time stationary Markov process. Under such formulation, power management is transformed to a stochastic optimization problem, and the result is optimal in the statistical sense. Unfortunately, this approach is not favorable for hard real-time systems such as embedded control applications with stringent timing requirements.

Static techniques are applied during design time, such as in the compilation and synthesis process. It takes the advantage that system specifications are known *a priori*. Several static power management policies have been investigated in [25], [27]–[33]. In [25], an optimal voltage scheduling algorithm is proposed for real-time systems scheduling. This approach identifies the so-called critical intervals iteratively, and schedules the real-time jobs via the earliest deadline first (EDF) policy. The authors in [29] studied a more general processor model, where the voltage of the processor cannot change instantly. They proposed a static algorithm which can achieve the optimum in some special scenarios. In [30], [31], the low energy non-preemptive scheduling problem is formulated as an integer linear programming problem. The system consists of a set of tasks with same arrival times and deadlines but different context switching penalties. In [32] an optimal result is obtained for a hard deadline non-preemptive system scheduled by EDF in a variable voltage processor with only two voltage levels. In [33], by associating a unique processor speed for each task, the authors proposed an optimal approach to find a feasible EDF-based schedule for the hard real-time system with tasks having different energy consumption characteristics (due to the different use of hardware components, switching activity, etc.) However, none of the above approaches can be simply applied to address the optimal voltage scheduling problem for systems employing a *fixed-priority preemptive* scheme. Such a scheduling scheme is adopted in most real-time schedulers of practical interest due to its low overhead and predictability [34]. Using the existing approaches would

Manuscript received March 7, 2002.

G. Quan is with the Department of Computer Science and Engineering, University of South Carolina. email:gquan@cse.sc.edu

X. S. Hu is with the Department of Computer Science and Engineering, University of Notre Dame. email:shu@cse.nd.edu

produce either invalid or poor quality results.

Our work in this paper strives to identify the theoretical limit on the energy consumed by a fixed-priority (FP) real-time system, given that the tasks have to be executed and finished by their deadlines. In this paper, we present an approach to optimally schedule an FP real-time system on a variable speed processor. It is optimal since not only every task can meet its deadline, but also the lowest possible energy is consumed. Our approach makes use of the work in [25]. We adjust the deadlines of the real-time tasks by carefully analyzing the preemptive effects among them. Then, we are able to transform the low energy fixed-priority scheduling problem into a *set* of low power EDF-based scheduling problems, and find the optimal voltage schedule for the original system. We find that this transformation may be computationally expensive, especially for real-time systems containing a large number of tasks. Therefore, we propose a technique to reduce the computation cost. We have conducted several experiments to compare the performances of other existing voltage scheduling techniques with our optimal voltage scheduling techniques. The experimental results demonstrate the advantages of our approach in terms of both energy saving and computational efficiency.

This paper is organized as follows. Section II introduces the necessary background. Section III provides several motivational examples. Section IV explains our optimal voltage scheduling algorithm for an FP real-time system. Section V introduces our techniques for reducing the computational complexity of the optimal algorithm. In section VI, we use experimental results to show the effectiveness and efficiency of our approach, and then compare several previous results with the optimal results. Finally, section VII concludes this paper. A preliminary of this paper was presented at a conference [35].

II. Preliminaries

The real-time system that we are interested in consists of N independent jobs, $\mathcal{J} = \{J_1, J_2, \cdots, J_N\}$, arranged in the decreasing order of their statically assigned priorities. Each job, $J_i = (R_i, C_i, D_i)$, is characterized by its arrival time R_i, workload C_i (CPU cycles, for example), and deadlines D_i. The execution time of a job depends on both the workload C_i as well as the the processor clock frequency, i.e., speed. Note that if $[R_j, D_j]$ of a lower priority job J_j is contained in $[R_i, D_i]$ of a higher priority job J_i, then J_i cannot finish after D_j without causing J_j to miss its deadline. Therefore, we assume that

$$R_i > R_j, \ or \ D_i \leq D_j, \ for \ i < j.$$

In our study, we also assume that the voltage can be varied continuously. Finally, we assume that the processor voltage, hence the speed, can be changed instantly. We conduct our research on such an ideal processor model based on the following reasons. First, we are more interested in studying the theoretical limit of energy saving when a variable voltage processor is used to execute real-time tasks. It is definitely important to find the *optimal* solution with the practical processor model, which is most likely a harder problem than the one we considered in this paper. From the research point of view, solving the problem for an ideal processor model can provide some valuable insights on solving the problem for a more practical processor model. Second, with the considerations of discrete voltage levels and voltage transition overhead, the energy saving is apparently lower than that based on the ideal processor model. Thus, the results obtained with the ideal processor model can be reasonably used as an upper bound on the energy saving. Many other previous related work also use the similar assumptions [19], [20], [25], [27], [28], [30].

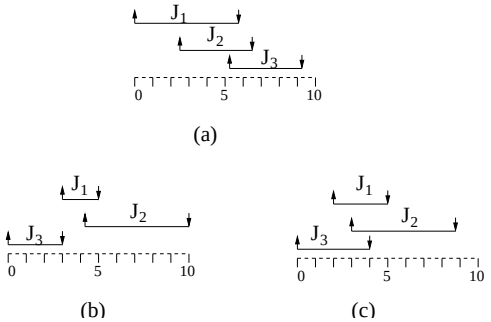


Fig. 1. Three real-time system examples.

The problem we are interested in is to find the optimal voltage schedule for a given real-time system with an FP assignment. This problem can be formulated as follows.

Definition 1: Given a job set $\mathcal{J}$, find a set of intervals, $[t_s^k, t_f^k]$, and their corresponding speeds, $\mathcal{S} = \{\overline{S}(t_s^k, t_f^k), k = 1, 2, \cdots, K\}$, where $\overline{S}(t_s^k, t_f^k)$ is a constant, such that if the processor operates accordingly, all the jobs can be *completed by their deadlines* and no other voltage schedules can consume less energy.

III. Motivational examples

An intuitive approach to search for the optimal voltage schedule is to apply the EDF-based optimal voltage scheduling algorithm [25] (**LPEDF**). However, it has been shown in [27] that simply applying LPEDF to an FP job set may cause a job to miss its deadline. On the other hand, there do exist some cases that applying LPEDF can guarantee the schedulability of the jobs, and thus provide the optimal voltage schedule for a real-time systems with the FP assignment.

Consider the three task systems shown in Figure 1, each of which has three jobs. For this figure and the following figures in this paper, we use an up (down) arrow to represent the arrival time (deadline) of a job, respectively. Note that, after the arrival times and deadlines of all the jobs are given, there is no fundamental difference between the FP-based scheduling and the EDF-based scheduling, except that the jobs may have different "fixed" priorities. Figure 1(a) is an example where the FP assignment is the same as that by EDF. The optimal voltage schedule for Figure 1(a) found by directly applying LPEDF is consequently the optimal voltage schedule under the FP assignment. Therefore, for an FP real-time system, when the EDF based priority assignment is the same as the original fixed priority, applying LPEDF will find the optimal solution.

Moreover, in certain cases, even though some real-time jobs have priorities different from the priority assignment by EDF,

we can still use LPEDF to find the optimal voltage schedule. Figure 1(b) is such an example. Note that in Figure 1(b), J_3 has lower priority and earlier deadline than both J_1 and J_2, but according to EDF it has the highest priority. Note that since J_3 must finishes before the arrivals of the J_1 and J_2, the execution of J_3 never interferes with the execution of J_1 and J_2 in any feasible schedule. Otherwise, J_3 would miss its deadline. For this example, the optimal voltage schedule by EDF scheduling is also the optimal schedule by FP based scheduling.

Specifically, we call the job sets in Figure 1(a) and (b) as *primary job sets*, which are formally defined as follows.

Definition 2: A job set $\mathcal{J}$ is called a **primary job set** if for any jobs $J_p, J_q \in \mathcal{J}$, $p < q$,either $D_p \leq D_q$ or $R_p \geq D_q$. For a primary job set, the following two lemmas can help us determine the optimal voltage schedule.

Lemma 1: A feasible voltage schedule for a primary job set $\mathcal{J}$ scheduled with FP scheme is also feasible for this job set scheduled with EDF scheme and vice versa.

Proof: Suppose that all the jobs in primary job set $\mathcal{J}$ satisfy that for any jobs J_p and J_q, $p < q$, we have $D_p \leq D_q$. It makes no difference to schedule $\mathcal{J}$ according to EDF or FP schemes, since both schemes lead to the same priority assignments. On the other hand, suppose for any two jobs J_p and J_q, $p < q$, we have $R_p \geq D_q$. Even though the priorities of J_p and J_q by FP are different from those by EDF, any feasible schedules will guarantee that J_q finishes before the arrival of J_p,i.e., execution of J_q does not interfere with that of J_p. Thus, the priority difference does not affect the voltage schedules in this case. ■

Lemma 2: The optimal voltage schedule for a primary job set $\mathcal{J}$ can be determined by applying LPEDF to $\mathcal{J}$.

Proof: Since LPEDF has been shown to be an optimal voltage scheduling algorithm based on EDF scheduling [25], together with Lemma 1, we prove the lemma. ■

Now, with Lemma 2, we are able to find the optimal voltage schedule by directly applying LPEDF if the given real-time job set is a primary job set. Unfortunately, not all job sets are primary job sets. Figure 1(c) is such an example. According to EDF, J_3 has the highest priority and should always finish first. However, according to the FP assignment, it can be preempted by J_1 and J_2 due to the choice of the processor speed. Therefore, some EDF feasible voltage schedules are no longer feasible for the FP assignment. This case will never happen for a primary job set. How can we find the optimal voltage schedule for this type of systems then? In the next section, we introduce a technique to transform an arbitrary set of real-time jobs to a set of primary job sets and find the optimal voltage for the original system.

IV. Optimal FP voltage schedule

In this section, we introduce our approach of finding the optimal voltage schedule for FP real-time systems, and provide the theoretical basis for our approach.

A. Overall approach

The basic idea of our approach is to transform the complicated problem of determining the optimal voltage schedule for an FP job set to an easier problem: finding the lowest energy consumption among the optimal voltage schedules for a number of primary job sets.

Two questions may arise for our approach: (*i*) why is the optimal voltage schedule for the original job sets among the optimal voltage schedules for some selected primary job sets? (*ii*) given a real-time system, how to identify such primary job sets? The following definition and theorem tend to answer these two questions.

Definition 3: The **associative job sets of** $\mathcal{J}$, denoted by $\mathcal{A}(\mathcal{J})$, are the job sets such that for any set $\mathcal{J}' \in \mathcal{A}(\mathcal{J})$, $C'_i = C_i, R'_i = R_i$, and $D'_i \leq D_i$ for $1 \leq i \leq N$. If $\mathcal{A}(\mathcal{J})$ also satisfies Definition 2, then it is called the **associative primary (AP) job sets of** $\mathcal{J}$, and is denoted by $\mathcal{AP}(\mathcal{J})$.

Theorem 1: The optimal voltage schedule for a job set $\mathcal{J}$ with an FP assignment is the schedule for the *associative primary (AP)* job set of $\mathcal{J}$ which consumes the minimum amount of energy.

Proof: To prove the theorem, we need only to show that the optimal voltage schedule of $\mathcal{J}$ is equivalent to the optimal voltage schedule of an associative primary job set of $\mathcal{J}$. Suppose $S = \{\overline{S}(t^i_s, t^i_f), i = 1, ..., k\}$ is the optimal voltage schedule of $\mathcal{J}$. After applying S, each job in $\mathcal{J}$ must finish at or before its deadline. We construct another job set $\mathcal{J}'$ as follows. For $J_i = (R_i, C_i, D_i) \in \mathcal{J}$, we introduce a new job $J'_i = (R'_i, C'_i, D'_i) \in \mathcal{J}'$. Let

$$R'_i = R_i, C'_i = C_i,$$

and let D'_i be the actual finishing time of J_i when applying S to $\mathcal{J}$. Apparently,

$$D'_i \leq D_i.$$

According to the FP scheduling, a lower priority job either finishes after the higher priority jobs, or arrives and finishes before the arrival of the higher priority jobs. Therefore, for any J'_p and J'_q $(p < q)$, we have either $D'_p \leq D'_q$ or $R'_p \geq D'_q$. That is, $\mathcal{J}'$ is an associative primary job set of $\mathcal{J}$.

Next, we use contradiction to show that S must be the optimal voltage schedule for $\mathcal{J}'$. Suppose S is not the optimal schedule for $\mathcal{J}'$, while S' is the optimal voltage schedule for $\mathcal{J}'$. Then S' must be able to feasibly schedule the jobs in $\mathcal{J}$ and consume less energy than S. This contradicts S being the optimal voltage schedule for $\mathcal{J}$. ■

From Theorem 1, one can conclude that the optimal voltage schedule for an FP job set $\mathcal{J}$ must be among the optimal voltage schedules for *all* the AP job sets of $\mathcal{J}$. However, according to Definition 3, there are infinite number of such job sets. It would be impossible to search all these job sets for the optimal voltage schedule. Fortunately, not all these primary job sets have to be constructed and checked for the optimal schedule. The following definition and theorem can help us reduce the search space for the optimal solution.

Definition 4: Given two real-time job sets, $\mathcal{J}_1 = J_{11}, J_{12}, ..., J_{1N}$ and $\mathcal{J}_2 = J_{21}, J_{22}, ..., J_{2N}$, where $C_{1i} = C_{2i}$ and $R_{1i} = R_{2i}, 1 \leq i \leq N$, job set $\mathcal{J}_2$ **dominates** $\mathcal{J}_1$ if $D_{1i} \leq D_{2i}$ for $1 \leq i \leq N$, which is denoted by $\mathcal{J}_2 \succeq \mathcal{J}_1$.

Lemma 3: If $\mathcal{J}_2$ dominates $\mathcal{J}_1$, the energy E_1 due to the optimal voltage scheduling of $\mathcal{J}_1$ is no less than that E_2, the

energy due to the optimal voltage scheduling of $\mathcal{J}_2$. That is,

$$\mathcal{J}_2 \succeq \mathcal{J}_1 \Longrightarrow E_2 \leq E_1.$$

Proof: Consider the optimal voltage schedule of $\mathcal{J}_1$. By applying the same voltage schedule, every job in $\mathcal{J}_2$ can also meet its deadline, since it has a later deadline compared with the corresponding job in $\mathcal{J}_1$. Hence, the energy due to the optimal scheduling of $\mathcal{J}_2$ will never be larger than that of $\mathcal{J}_1$. ■

According to Theorem 3, if job set $\mathcal{J}_1$ dominates $\mathcal{J}_2$, we need only check if $\mathcal{J}_1$ is the optimal schedule. Thus, to search for the optimal voltage schedule, we only need to examine those AP job sets not dominated by others. Next, we formally define the term *non-dominated associative primary job sets of* $\mathcal{J}$, then summarize this conclusion in Theorem 2.

Definition 5: The **non-dominated associative primary (NAP) job sets of** $\mathcal{J}$, denoted as $\mathcal{NAP}(\mathcal{J})$, is the AP job sets of $\mathcal{J}$ such that none of the job sets in $\mathcal{NAP}(\mathcal{J})$ dominates another, and any other AP job set of $\mathcal{J}$ is dominated by at least one of the job sets in $\mathcal{NAP}(\mathcal{J})$.

Theorem 2: The optimal energy for scheduling $\mathcal{J}$ with an FP assignment is the energy $E_{opt} = \min E_i$, where E_i is the energy due to the optimal voltage scheduling of job set $J_i \in \mathcal{NAP}(\mathcal{J})$.

Proof: According to Theorem 1, the optimal voltage schedule for $\mathcal{J}$ is the voltage schedule for one of the associative primary job sets of $\mathcal{J}$, i.e., $\mathcal{AP}(\mathcal{J})$, which consumes the least energy. Therefore, from Lemma 3, the conclusion must be true. ■

Based on Theorem 2, we have an algorithm (see Algorithm 1) to find the optimal voltage schedule of job set $\mathcal{J}$. Algorithm 1 first searches all the NAP job sets of $\mathcal{J}$. Then the energy due to the optimal scheduling of each of these job sets are computed, and the voltage schedule with the lowest energy consumption is output as the optimal schedule for $\mathcal{J}$. The main challenge in Algorithm 1, however, is how to find all the NAP job sets (function "Search_Primary"), which is discussed in the next subsection.

Algorithm 1 Finding the Optimal Voltage Schedule

1: **Input:** A real-time job set $\mathcal{J} = \{J_1, ..., J_N\}$ ordered in the decreasing order of their priorities
2: **Output:** The optimal voltage schedule S_{opt} and its energy consumption.
3: **Search_Primary**($\mathcal{J}, \mathcal{T}$)
4: // search for the NAP job sets of $\mathcal{J}$ and put in $\mathcal{T}$
5: **for** each $T_i \in \mathcal{T}$ **do**
6: S_i= the optimal voltage schedule of T_i obtained by LPEDF;
7: **end for**
8: $S_{opt} = S_k$, S_k has the lowest energy consumption among $S_i, 1 \leq i \leq N$;

B. Finding the NAP job sets

To find the NAP job sets for a given real-time job sets, we need to tackle two problems: (*i*) how to generate the AP job sets, and (*ii*) how to guarantee that all the NAP job sets are covered. To achieve this goal, we explore more attributes of the NAP job sets. The following lemmas reveal some interesting characteristics of NAP job sets, and will be used extensively later.

Lemma 4: Let $\mathcal{J}' \in \mathcal{NAP}(\mathcal{J})$. Then for job $J'_m \in \mathcal{J}'$, with $D'_m = \max\{D'_i | J'_i \in \mathcal{J}'\}$ (if ties happen, select the one with the lowest priority), we have $D'_m = D_m$.

Proof: We prove this lemma by contradiction. Let $\mathcal{J}' \in \mathcal{NAP}(\mathcal{J})$, and $J'_m \in \mathcal{J}'$ such that $D'_m = \max\{D'_i | J'_i \in \mathcal{J}'\}$ (if tie happens, select the one with the lowest priority). Assume that $D'_m < D_m$. Then, by extending the deadline of J'_m to D_m while keeping the deadlines of the rest of jobs unchanged, we get another job set $\mathcal{K}'$.

Since $\mathcal{J}'$ is a primary job set of $\mathcal{J}$, according to Definition 2, for any job $J'_p \in \mathcal{J}', p < m$, we have $D'_p \leq D'_m$, and thus $D'_p < D_m$; for any job $J'_p \in \mathcal{J}', p > m$, we have $D'_p \leq R'_m < D_m$. Therefore, $\mathcal{K}'$ must also be a primary job set of $\mathcal{J}$. Moreover, since other jobs than $J'_m \in \mathcal{J}'$ have the same deadlines as those in $\mathcal{K}'$, and $J'_m \in \mathcal{J}'$ has a smaller deadline than its corresponding job in $\mathcal{K}'$, so $\mathcal{K}' \succeq \mathcal{J}'$ according to Definition 4. This contradicts to our assumption that $\mathcal{J}'$ is an NAP job set of $\mathcal{J}$.

On the other hand, D'_m cannot be greater than D_m according to Definition 3, therefore, $D'_m = D_m$. ■

Lemma 4 essentially states that the latest deadline in any NAP job set of $\mathcal{J}$ must equal the original deadline of its corresponding job in $\mathcal{J}$. The importance of this lemma will be seen later.

Lemma 5: Consider a job set $\mathcal{J} = \{J_1, J_2, \cdots, J_N\}$ and one of its AP job sets $\mathcal{J}' = \{J'_1, J'_2, \cdots, J'_N\}$. For some $J_k \in \mathcal{J}$ and $J'_k \in \mathcal{J}'$, let $D'_k = D_k$, then the following must be true:

- for any $i < k$,

$$D'_i \leq D_k \quad if\ R_i < D_k\ and\ D_i > D_k, \tag{1}$$

- for any $i > k$,

$$D'_i \leq R_k \quad if\ R_k < D_i < D_k. \tag{2}$$

Proof: According to Definition 2 and 3, (1) and (2) must be true. ■

Based on Lemma 4 and Lemma 5, we devise a procedure to search for the NAP job sets and summariz it in Algorithm 2. In Algorithm 2, to construct the NAP job sets for a given job set, we fix, one by one, the deadline of each job. By *fixing the deadline of a job*, we mean that the job's deadline is set to its largest possible value, and the deadlines for the rest of the jobs are adjusted according to Lemma 5. After fixing the deadline of a job, we remove it and go through this procedure for the rest of the jobs again. This procedure continues recursively until the job set containing the rest of the jobs is a primary job set. Then we put back all the jobs whose deadlines have been fixed to the resultant job sets. Figure 2 shows the NAP job sets found by applying Algorithm 2 to the system in Figure 1(c).

To demonstrate that Algorithm 1, combined with Algorithm 2, indeed produces the feasible optimal voltage schedule for an FP real-time system, we have the following lemma and theorem (The proofs are shown in the Appendix).

Algorithm 2 Search for the NAP job sets

Search_Primary ($\mathcal{J}$, $\mathcal{T}$)
Input: $\mathcal{J} = (J_1, ..., J_N)$, where $J_i = (R_i, C_i, D_i)$
Output: A set $\mathcal{T}$ which covers all the NAP job sets of $\mathcal{J}$.
for $k = 1 \cdots N$ **do**
  Copy $\mathcal{J}$ to $\mathcal{J}_k$;
  for $J_i \in \mathcal{J}_k, i < k$ **do**
    if $R_i < D_k$ *and* $D_i > D_k$ **then**
      $D_i = D_k$;
    end if
  end for
  for $J_i \in \mathcal{J}_k, i > k$ **do**
    if $R_k < D_i < D_k$ **then**
      $D_i = R_k$;
    end if
  end for
  $\mathcal{J}_k = \mathcal{J}_k - J_k$;
  if $\mathcal{J}_k$ is not a primary job set **then**
    Search_Primary($\mathcal{J}_k$, $\mathcal{T}'$); //recursive calls
    Add J_k to each job set in $\mathcal{T}'$;
    Add each $\mathcal{T}_i' \in \mathcal{T}'$ to $\mathcal{T}$ if $\mathcal{T}_i'$ is a primary job set;
  else
    $\mathcal{J}_k = \mathcal{J}_k + J_k$;
    Add $\mathcal{J}_k$ to $\mathcal{T}$;
  end if
end for

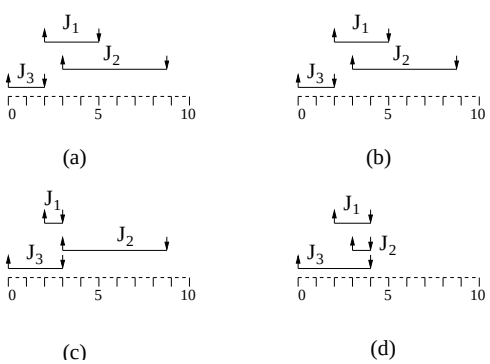


Fig. 2. Non-dominated primary job sets output from Algorithm 2 for the task set shown in Figure 1(c). (a) is the result by first fixing J_1 to its deadline. (b) and (c) are the results by first fixing J_2 to its deadline. (d) is the result by first fixing J_3 to its deadline.

Lemma 6: The job sets $\mathcal{T}$ output from Algorithm 2 cover all the NAP job sets of $\mathcal{J}$.

Theorem 3: Algorithm 1 produces the optimal voltage schedule for real-time job set $\mathcal{J}$.

The computation cost for Algorithm 1 consists of two parts: the cost for searching the NAP job sets (Algorithm 2), and the cost for searching the optimal schedule among these job sets (LPEDF). Note that the computational complexity of Algorithm 2 is $O(N!)$, where N is the number of jobs, and the complexity of LPEDF is $O(N^2)$(or $O(Nlog^2N)$ with an more efficient implementation) according to [25]. Therefore, the complexity for Algorithm 1 is $O(N! + MN^2)$, or $O(N! + MNlog^2N)$ if LPEDF is more efficiently implemented [25],

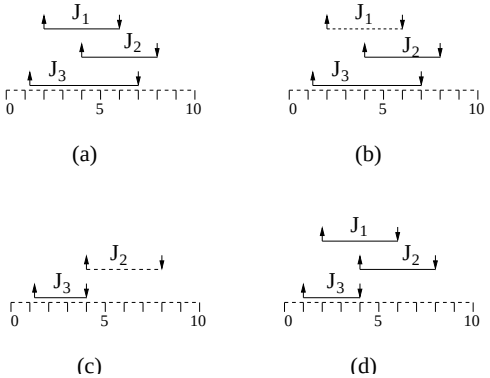


Fig. 3. Non-primary job sets may be generated from Algorithm 2. (a) A given job set, (b) fixing the deadline of J_1, (c) fixing the deadline of J_2, (d) putting back J_1 and J_2 and the job set is not a primary job set.

where M is the job sets output from Algorithm 2. When N increases, Algorithm 1 can be quite time consuming.

After a careful study, we also note that not all the job sets constructed during the execution of Algorithm 2 are primary job sets. Figure 3 is such an example. Figure 3(a) is a real-time system with three jobs. Let us first fix J_1 to its deadline and adjust the deadlines for J_2, J_3, we have the result in Figure 3(b). Again, fixing the deadline of J_2 to its largest possible value, we have the result shown in Figure 3(c). However, when we put back both J_2 and J_1 back to Figure 3(c), the job set, as shown in Figure 3(d), is not a primary job set since J_3 has a earlier deadline but lower priority than J_1. In our algorithm, we simply discard these job sets. While these job sets will not affect the search for the optimal schedule (Theorem 2), it does make the program take unnecessary CPU time. In next section, we will discuss how to eliminate these job sets and improve the computational efficiency of this algorithm.

V. Improve the computational efficiency

In this section, we propose an approach to improve the efficiency of Algorithm 2. Recall that not all the job sets constructed during the execution of Algorithm 2 are primary job sets. Searching for these job sets does not help find the optimal voltage schedule. Moreover, same primary job sets may be constructed more than once by Algorithm 2, and all these primary job sets are then evaluated with LPEDF. Figure 2 is such an example. In the primary job sets shown in Figure 2, Figure 2(a) and (c) are identical. This is because the same primary job set may be constructed in different recursive calls in Algorithm 2. Note that the primary job set in Figure 2(a) can either be searched by first fixing the deadline of J_1 and then J_2, or vice versa. This situation exasperates when the number of jobs is large. Even though such redundancy in the algorithm will not damage the optimality of the results (Theorem 2 and Theorem 3), they do make the algorithm quite inefficient, especially for systems with large number of jobs. We call both the non-primary job sets and the identical copies of the primary job sets as *redundant job sets*.

One way to reduce the redundancy is to eliminate the identical copies of the same primary job sets once all the primary

job sets have been constructed. However, a straightforward implementation of doing this will have a worst case complexity of $O(M^2)$, where M is the total number of the job sets. When the number of jobs is quite large, M can be very high. Furthermore, constructing all these redundant job sets is an unnecessary effort. Therefore, we focus our effort on how to *avoid* generating the redundant job sets in the algorithm. Since same primary job sets may be constructed by different loops in Algorithm 2, our problem then is how to identify those loops that will generate redundant job sets. Before we introduce our approach in this endeavor, we first introduce two important lemmas that helps to identify the cases where fixing two different job deadlines (in two different outmost loops of Algorithm 2) results in the same AP job sets of $\mathcal{J}$.

Lemma 7: In p-th outmost loop of Algorithm 2, the AP job sets constructed by fixing the deadline of $J_p \in \mathcal{J}$ cover all those NAP job sets $\mathcal{J}'$ with $D'_p = D_p$.

Proof: Let $\hat{\mathcal{J}}(J_p)$ be the job set by fixing the deadline of J_p but not including J_p (i.e., generated by line 6-15 in Algorithm 2).

- If $\hat{\mathcal{J}}(J_p)$ is not a primary job set, according to Lemma 6, the AP job sets found in the subsequent recursive call, i.e. line 18, must contain all the NAP job sets associated with $\hat{\mathcal{J}}(J_p)$. On the other hand, for any $\mathcal{J}' \in \mathcal{NAP}(\mathcal{J})$ and $J'_p \in \mathcal{J}'$ with $D'_p = D_p$ (thus $J'_p = J_p$), we must have $(\mathcal{J}' - J'_p) \in \mathcal{NAP}(\hat{\mathcal{J}}(J_p))$. Therefore, after putting back J_p, i.e. line 19-20 in Algorithm 2, the resultant AP job sets must contain $\mathcal{J}'$.
- If $\hat{\mathcal{J}}(J_p)$ is a primary job set, according to Lemma 5, $\mathcal{J}' = \hat{\mathcal{J}}(J_p) + J_p$ is the only NAP job set of $\mathcal{J}$ with $D'_p = D_p$. $\mathcal{J}'$ can certainly be found, i.e., through line 22-23 in Algorithm 2.

■

From Lemma 7, the AP job sets found by the p-th outmost loop in Algorithm 2 will cover those NAP job sets with (at least) the deadline of the p-th job equals its original one. Since more than one job in an NAP job set may have deadline equals its original one, this is part of the reasons why different outmost loops may result in the same AP job sets. The following Lemma will help us identify and then reduce the overlap among the AP job sets found by each of the loops.

Lemma 8: Let $\mathcal{J}'$ be an NAP job set of $\mathcal{J}$ and assume that for some $J'_p \in \mathcal{J}'$, we have $D'_p = D_p$. Let

$$\mathcal{A} = \{J_k | J_k \in \mathcal{J}, R_k \geq D_p, k < p\},$$

$$\mathcal{B} = \{J_k | J_k \in \mathcal{J}, R_k \geq R_p, k > p\}.$$

Then, if $\mathcal{A} \cup \mathcal{B} \neq \emptyset$, there must be a job J_q $(p \neq q)$ such that $J'_q \in \mathcal{J}'$ satisfies $D'_q = D_q$.

Proof: To prove this lemma, we first show that if $\mathcal{A} \cup \mathcal{B} \neq \emptyset$, there must exist one job $J'_i \in \mathcal{J}'$ such that $D'_i \geq D'_p$. Consider the following cases.

- $\mathcal{A} \neq \emptyset$:
  Let $J_i \in \mathcal{A}$. Then $R_i \geq D_p$ and $i < p$. For $J'_i \in \mathcal{J}'$, we have $D'_i > R_i$. Since $D'_p \leq D_p$, so $D'_i > D'_p$.
- $\mathcal{B} \neq \emptyset$:
  Let $J_i \in \mathcal{B}$. Then, $R_i \geq R_p$ and $i > p$. If $R_i \geq D_p$, we have $D'_i > D_p = D'_p$ for $J'_i \in \mathcal{J}'$. On the other hand, if $R_p \leq R_i \leq D_p$, according to Definition 2, we must have $D'_i \geq D'_p$ since $\mathcal{J}'$ is a primary job set.

Overall, if $\mathcal{A} \cup \mathcal{B} \neq \emptyset$, we must be able to find a job $J'_i \in \mathcal{J}'$ such that $D'_i > D'_p$ or $D'_i = D'_p$ with $(i > p)$. With Lemma 4, we know that there must be a job $J'_q \in \mathcal{J}'$ such that $D'_q = D_q$. ■

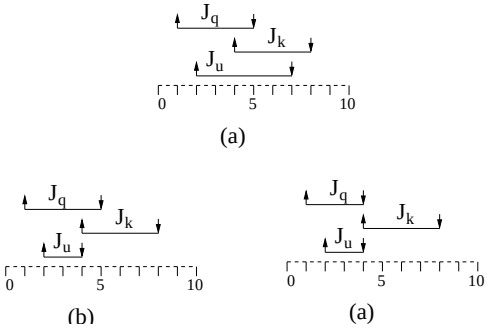


Fig. 4. (a) A given original job set, (b) the associative job set by first fixing the deadline of J_q then $J_k(\hat{\mathcal{J}}(J_qJ_k))$, (c) the associate job set by first fixing the deadline of J_k then J_q $(\hat{\mathcal{J}}(J_kJ_q))$.

Based on the above lemmas, we have the following theorem which forms the basis for reducing the redundant job sets and lead to a dramatic improvement of computational efficiency of Algorithm 1.

Theorem 4: Let job $J_p \in \mathcal{J}$, and

$$\mathcal{A} = \{J_k | J_k \in \mathcal{J}, R_k \geq D_p, k < p\},$$

$$\mathcal{B} = \{J_k | J_k \in \mathcal{J}, R_k \geq R_p, k > p\}.$$

if $\mathcal{A} \cup \mathcal{B} \neq \emptyset$, then the NAP job sets constructed by first fixing the deadline of J_p are redundant.

Proof: Let $\mathcal{J}'$ be one of the NAP job sets constructed by first fixing the deadline of J_p. Then according to Lemma 8, there must exist a $q \neq p$ such that for $J'_q \in \mathcal{J}'$, we have $D'_q = D_q$. However, according to Lemma 7, by first fixing the deadline of J_q, Algorithm 2 produces all the NAP job sets with $D'_q = D_q$, including $\mathcal{J}'$. Therefore, $\mathcal{J}'$ is constructed more than once by Algorithm 2. ■

Based on Theorem 4, we propose an improved algorithm for finding the NAP job sets of a given job set and summarize it in Algorithm 3. Algorithm 3 avoids the construction of AP job sets corresponding to fixing the deadlines of the jobs determined by Theorem 4 (see line 5-7). Therefore, it is far more efficient than Algorithm 2 because it checks and removes the possibility of producing redundant job sets in each recursive call. Since a great number of identical associative primary job sets are removed, the effort to search for the optimal voltage schedule among these job sets is also saved. The improvement achieved by Algorithm 3 will be further demonstrated through experimental results in the next section. Moreover, Algorithm 3 provides another important improvement. Recall that Algorithm 2 may result in non-primary job sets which need to be identified to avoid applying LPEDF to these sets (see line 20 in Algorithm 2). By using Algorithm 3, the construction and detection effort for such non-primary job sets are completely eliminated as stated in

Algorithm 3 Improved algorithm for searching the NAP job sets

1: **Search_Primary** $(\mathcal{J}, \mathcal{T})$
2: **Input:** A real-time job set $\mathcal{J} = J_1, ..., J_n$ ordered in decreasing order of their priorities, where $J_i = (R_i, C_i, D_i)$
3: **Output:** A set $\mathcal{T}$ containing all the NAP job sets of $\mathcal{J}$
4: **for** $k = 1 \cdots N$ **do**
5: **if** there is $J_q \in \mathcal{J}$, such that $q > k, R_q > R_k$, and $D_q \geq D_k$, or $q < k$, and $R_q \geq D_k$ **then**
6: continue; // See Theorem 4
7: **end if**
8: Copy $\mathcal{J}$ to $\mathcal{J}_k$;
9: **for** $J_i \in \mathcal{J}_k, i < k$ **do**
10: **if** $R_i < D_k$ *and* $D_i > D_k$ **then**
11: $D_i = D_k$;
12: **end if**
13: **end for**
14: **for** $J_i \in \mathcal{J}_k, i > k$ **do**
15: **if** $R_k < D_i < D_k$ **then**
16: $D_i = R_k$;
17: **end if**
18: **end for**
19: $\mathcal{J}_k = \mathcal{J}_k - J_k$;
20: **if** $\mathcal{J}_k$ is not a primary job set **then**
21: Search_Primary($\mathcal{J}_k$, $\mathcal{T}'$);
22: Add J_k to each job set in $\mathcal{T}'$;
23: Add each job set in $\mathcal{T}'$ to $\mathcal{T}$; // See Theorem 5
24: **else**
25: $\mathcal{J}_k = \mathcal{J}_k + J_k$;
26: Add $\mathcal{J}_k$ to $\mathcal{T}$;
27: **end if**
28: **end for**

Theorem 5 (see line 23 in Algorithm 3). The proof for this theorem is shown in the Appendix.

Theorem 5: The job sets output from Algorithm 3 are all primary job sets.

VI. Experimental Results

In this section, we use some experiments to compare our optimal voltage scheduling results for FP real-time systems with two other related approaches, the heuristic approach introduced in [27] and the approach presented in [19]. In our experiments, we also demonstrate that previously established low energy consumption bounds, i.e., scheduled EDF scheme, cannot be properly used for the real time systems scheduled by FP scheme. Finally, we use experimental results to show the significant improvement of the computation efficiency by applying Algorithm 3 in Algorithm 1. All the experiments are conducted using Sun Blade 220. According to [6], we assume that the processor speed is proportional to the supply voltage and the processor power consumption is a cubic function of the processor speed. Note that our algorithm only requires that the power consumption is a convex function of the supply voltage.

Our first experiment consists of 10 groups of randomly generated real-time systems with the number of jobs being 2, 4, $\cdots$, 20. The arrival times and deadlines of these jobs are chosen to be uniformly distributed within $[0, 50]$, $[20, 100]$, respectively. These data are randomly chosen without special considerations. The execution time of each job is randomly generated from 1 to half of its deadline to make the job sets easier to schedule under the maximum processor speed. Only the job sets that are schedulable under the maximum processor speed are used in our experiment, and each group contains at least 100 such schedulable job sets. Four algorithms, *i.e.*, the heuristic approach (**VSLP**) in [27], the approach (**LPFS**) in [19], and the optimal EDF approach [25], and the optimal fixed-priority approach (**OPT_FP**, that is, Algorithm 1 combined with Algorithm 3), are tested with these systems. To reduce statistical errors, we collected the average energy consumption for each group and filled into Table I. Within each group, we also recorded the largest deviation of the energy consumption results by each of these three approaches to the corresponding optimal results. All the collected data are normalized against the optimal results. To compare the computational cost for the voltage schedule, we also gather the average CPU time by approach **VSLP** and **OPT_FP** (the average CPU time for **LPFS** is very close to zero and therefore omitted.)

We also performed the same experiments on a real-world application, a typical videophone application introduced in [36], and the results are shown in the last row of Table I.

Table I shows that our optimal approach has a much higher computational cost than **VSLP** and **LPFS**, and its computation complexity increases rapidly as the number of the jobs increases. This agrees with our theoretical analysis since the complexity for **VSLP** is $O(N^3)$, while the worst case computation complexity for **OPT** is $O(N! + mlog^2m)$ (where N is the number of jobs, and m is the number of primary job sets searched in **OPT**). Table I also shows that **VSLP** represents an excellent trade-off choice in searching for the voltage schedule. Note that, in Table I, the difference between its average power consumption and that of **OPT** is very close, which is much better than that of **LPFS**, and it cost much less CPU time than **OPT**. However, for some test cases (when the job number is 16, for example), the voltage schedules found by **LPFS** can consume as 2.3 times energy as that by the optimal ones. Finally, from Table I, we conclude that it is not proper to use the energy consumption bound set up by LPEDF for the real-time systems scheduled by FP policy. From our experiments, the average energy consumption bound for jobs scheduled by EDF is only 50% - 60% of that by FP. In some cases, the optimal energy consumption by EDF is less than 5% of that by FP.

Our second experiment quantifies the improvement of computation efficiency made by Algorithm 3. The same technique are used to generate random systems. The average CPU time of the optimal algorithm by adopting two different strategies, namely Algorithm 2 or Algorithm 3, in searching for the NAP job sets, are collected and shown in Table II. Table II shows the dramatic reduction of computational cost by applying Algorithm 3, especially for systems with large number of jobs. This is because applying Algorithm 3 reduces not only the effort to construct those redundant job sets, but also the effort to search for the optimal voltage schedules among these sets.

TABLE I

EXPERIMENTAL RESULTS FOR COMPARING THE THREE APPROACHES: **OPT**, **VSLP**, AND **LPFS**. THE NUMBERS IN THE COLUMNS LABELLED AS **Avg. Energy** AND **Max. Deviation** ARE NORMALIZED AGAINST THE FP OPTIMAL RESULTS.

Systems	Avg. Energy			Max. Deviation			Avg. CPU Time (s)	
	LPFS	VSLP	LPEDF	LPFS	VSLP	LPEDF	VSLP	OPT_FP
2	4.00997	1.01375	0.676259	224.15	2.59466	0.9647337	0.00	0.00
4	2.98437	1.01153	0.51942	47.4246	0.6645	0.9285487	0.00	0.01
6	2.03882	1.01038	0.471015	19.9358	0.61839	0.942235	0.01	0.01
8	1.73526	1.00931	0.492956	13.3007	0.58081	0.9231593	0.01	0.08
10	1.51272	1.02183	0.487638	7.16493	0.93676	0.9117454	0.01	0.29
12	1.42053	1.00589	0.534376	2.72621	0.43358	0.866363	0.01	0.98
14	1.31968	1.00337	0.583075	2.60185	0.18463	0.789502	0.02	2.81
16	1.27567	1.01304	0.573279	2.80499	1.3268	0.806746	0.01	27.94
18	1.28008	1.00029	0.601162	1.33458	0.01808	0.822035	0.02	395.66
20	1.18782	1.00325	0.617864	0.6283	0.06743	0.740367	0.02	1626.3
video phone	7.75383	1.000000	0.994985	6.75383	0.00000	0.005015	0.00	0.01

Since the number of redundant job sets increases drastically as the number of job increases, this explains the dramatic improvement by using the improved approach.

TABLE II

EXPERIMENTAL RESULTS FOR COMPARING THE COMPUTATION EFFICIENCY OF THE OPTIMAL SCHEDULING APPROACH BY USING TWO STRATEGIES, ALGORITHM 2 AND ALGORITHM 3, IN SEARCHING FOR THE NON-DOMINATED PRIMARY JOB SETS.

No. Jobs	CPU Time(s)	
	Algorithm 2	Algorithm 3
2	0.00	0.00
4	0.01	0.01
6	0.09	0.01
8	3.21	0.08
10	220.56	0.29
12	11356.42	0.98

VII. SUMMARY

In this paper, we present an approach to finding an optimal voltage schedule in terms of energy saving for a variable voltage processor executing fixed-priority, real-time jobs. We introduce the concept of *non-dominated, associative, primary* job sets and prove that an optimal voltage schedule of a given job set must be the same as that of one of NAP job sets. Two algorithms are developed to construct NAP job sets for a given job set with one improving the other one. Experimental results are shown to compare our results with relevant previous ones.

The type of systems studied in this paper contains real-time jobs to be scheduled based on the fixed priority, pre-emptive scheme. Such a scheduling scheme is used widely in many real-world real-time systems due to its simplicity and predictability [34]. The static voltage scheduling approach adopted here can be readily used during the design process to fully exploit the timing information known *a priori*. Furthermore, the static approach can be supplemented by a dynamic voltage scheduling such as the one proposed in [19] to achieve the best overall result. The significance of the results presented here is that the inherent theoretical limit in terms of energy saving for the systems of interest is established. Such results can be used as the standard to measure the quality of various heuristic approaches.

APPENDIX I
PROOF OF LEMMA 6

Proof: We prove this lemma by mathematical induction.

- When $N = 1$, the conclusion is true since the job set output from Algorithm 2 contains only one job.
- Suppose the job sets output from Algorithm 2 can cover all the NAP job sets for $N = k - 1$ $(k > 1)$. We prove the case for $N = k$ by contradiction.
  Let $\mathcal{J} = \{J_1, J_2, \cdots, J_k\}$. Assume $\tilde{\mathcal{J}}$ is an AP job set of $\mathcal{J}$ not dominated by any AP job sets found by Algorithm 2. According to Lemma 4, for job $\tilde{J}_p \in \tilde{\mathcal{J}}$ where $\tilde{D}_p = \max_{\tilde{J}_i \in \tilde{\mathcal{J}}}\{\tilde{D}_i\}$ (if tie happens, select the one with the lowest priority), we have $\tilde{D}_p = D_p$ (and thus $J_p = \tilde{J}_p$).
  Now let us consider the associative job sets obtained from Algorithm 2. In the p-th iteration of the outermost loop, the deadline of J_p is fixed to D_p while the procedure recursively construct the primary job set for the rest of the $k-1$ jobs. From the induction hypothesis, there must exist a primary job set, $\mathcal{K}'$, constructed from Algorithm 2, such that $\mathcal{K}' \succeq \{\tilde{\mathcal{J}} - \{\tilde{J}_p\}\}$. (Note that $\{\tilde{\mathcal{J}} - \{\tilde{J}_p\}\}$ is still a primary job set.) According to Definition 4, it follows that $\mathcal{K}' + \{\tilde{J}_p\} \succeq \tilde{\mathcal{J}}$. Next, we only need to show that $\mathcal{K}' + \{\tilde{J}_p\}$ is an AP set of $\mathcal{J}$.
  The jobs in $\mathcal{K}'$ have some useful characteristics:
  - For any job $J'_r \in \mathcal{K}', r < p$, we have $D'_r \leq D_p$.
    If we have $D'_r > D_p$, according to Algorithm 2, there must be a job $J_s \in \mathcal{J}$ such that $R_s \geq D_p$. In this case, the deadline of $\tilde{J}_p$, $\tilde{D}_p$, cannot possibly be the latest. This contradicts our assumption above.
  - For any job $J'_r \in \mathcal{K}', r > p$, we have $D'_r \leq R_p$.
    Since job $\tilde{\mathcal{J}}$ is an AP job set and $\tilde{J}_p$ has the lowest priority among the jobs that may share the same largest deadline, $\tilde{D}_p$, thus for any $\tilde{J}_r \in \tilde{\mathcal{J}}, r > p$, we have $\tilde{R}_r < \tilde{R}_p$ and $\tilde{D}_r < \tilde{R}_p$. Otherwise, $\tilde{J}_r$ cannot has earlier deadline than $\tilde{J}_p$ (see the discussion at the beginning of Section II). Moreover, since $\mathcal{K}' \succeq \{\tilde{\mathcal{J}} - \{\tilde{J}_p\}\}$, so for any job $J'_r \in \mathcal{K}', r > p$, we have $D'_r \leq \tilde{D}_r \leq \tilde{R}_p = R_p$.

  Based on the above properties, we can conclude that the job set $\mathcal{K}' + \{J_p\}$ is an AP job set according to Defini-

tion 2. Furthermore, $\mathcal{K}' + \{J_p\}$ dominates $\tilde{\mathcal{J}}$ according to Definition 4. However, this conclusion contradicts our assumption that $\tilde{\mathcal{J}}$ is not dominated by any AP job sets output from Algorithm 2.

■

APPENDIX II
PROOF OF THEOREM 3

Proof: According to Theorem 1, to prove this theorem, we need to prove the following two conditions,

- *Condition 1*: the schedule can guarantee the feasibility of all the jobs, and
- *Condition 2*: the output from Algorithm 2 has covered all the NAP job sets derived from J.

Since the optimal voltage schedule for $\mathcal{J}$ is the optimal voltage schedule for one of its associative job set, and the deadlines of the jobs in the primary job sets is no later than those in the original job set, this feasible schedule certainly can guarantee the schedulability of each jobs in $\mathcal{J}$, and thus *Condition 1* must be true. *Condition 2* is also true as shown in Lemma 6.

■

APPENDIX III
PROOF OF THEOREM 5

Proof: Let $\hat{\mathcal{J}}(J_k)$ be the job set by fixing the deadline of J_k and adjusting the deadlines of other jobs according to Lemma 5 (or line 9 to line 18 in Algorithm 3), and let $\mathcal{AP}(\hat{\mathcal{J}}(J_k))$ be the AP job sets of $\hat{\mathcal{J}}(J_k)$ obtained in Algorithm 3. Then, to prove the theorem, we only need to show that for any $\tilde{\mathcal{J}} \in \mathcal{AP}(\hat{\mathcal{J}}(J_k))$, $\tilde{\mathcal{J}} + J_k$ is still a primary job set.

In Algorithm 3, we fix the deadline of J_k only if

$$R_i < D_k, \quad \text{for all } i < k, \text{ and} \tag{3}$$

$$R_i < R_k, \quad \text{for all } i > k. \tag{4}$$

So for any $\hat{J}_i \in \hat{\mathcal{J}}(J_k)$, we have

$$\hat{D}_i \leq D_k, \quad i < k, \tag{5}$$

$$\hat{D}_i \leq R_k, \text{ or } \hat{D}_i \geq D_k, \quad i > k. \tag{6}$$

Consider any $\tilde{\mathcal{J}} \in \mathcal{AP}(\hat{\mathcal{J}}(J_k))$, and $\tilde{J}_i \in \tilde{\mathcal{J}}$.

- For $\hat{D}_i \leq D_k, i \neq k$,
  according to (6) and Lemma 5, we have $\tilde{D}_i \leq \hat{D}_i \leq D_k$.
- For $\hat{D}_i \geq D_k, i > k$,
  according to Lemma 5 (or line 9 to line 18 in Algorithm 3), fixing the deadline of $\hat{J}_p, p < k$, does not bring any change to the deadline of $\hat{J}_i$ since $\hat{D}_p \leq D_k \leq \hat{D}_i$; on the other hand, after fixing the deadline of $\hat{J}_p, p > k$, the deadline of $\hat{J}_i$, $\hat{D}_i$, can only be adjusted to $\hat{R}_p$, $\hat{D}_p$, or remain unchanged. From (4),(6), and Lemma 5 (or line 9 to line 18 in Algorithm 3), the newly adjusted deadline of $\hat{J}_i$ can only be less than R_k or greater than D_k.

Overall, for any $\tilde{J}_i \in \tilde{\mathcal{J}}$, we must have

$$\tilde{D}_i \leq D_k, \quad i < k,$$

$$\tilde{D}_i \leq R_k, \text{ or } \tilde{D}_i \geq D_k, \quad i > k.$$

Therefore, from Definition 2, $\tilde{\mathcal{J}} + J_k$ is still a primary job set.

■

ACKNOWLEDGMENT

This work is supported in part by NSF under grant number MIP-9701416 and CCR-9988468, and by DARPA and Rome Laboratory, Air Force Material Command, USAF, under cooperative agreement FC 30602-00-2-0525 as part of the Power Aware Communication and Computation (PACC) program.

REFERENCES

[1] G. Micheli and L. Benini, "System-level low power optimization: Techniques and tools," *Trans. on Design Auto. of Electr. Sys.*, vol. 5, no. 2, April 2000.

[2] M.Pedram, "Power minimization in ic design: principles and applications," *ACM Trans. on Design Auto. of Electr. Sys.*, vol. 1, no. 1, pp. 3–56, January 1996.

[3] F. Najm, "A survey of power estimation techniques in vlsi circuits," *IEEE Trans. on VLSI*, vol. 2, no. 4, pp. 446–455, 1994.

[4] S. Devades and S. Malik, "A survey of optimization techniques targeting low power vlsi circuits," in *DAC*, 1995, pp. 242–247.

[5] T. D. Burd and R. W. Brodersen, "Design issues for dynamic voltage scaling," in *ISLPED*, 2000, pp. 9–14.

[6] T. Burd, *Energy-Efficient Processor System Design*. Department of Electrical Engineering and Computer Sciences, Uiversity of California, Berkeley: Ph.D. Thesis, 2001.

[7] Intel. (2002) Developer manual: Intel 80200 processor based on intel xscale microarchitecture. [Online]. Available: http://developer.intel.com/design/iio/manuals/273411.htm

[8] D. Laird. (2000, January) Crusoe processor products and technology. [Online]. Available: http://www.transmeta.com/technology/architecture/index.html

[9] AMD. (2001) Mobile amd duron processor. [Online]. Available: http://www.amd.com/

[10] L. Benini, A. Bogliolo, and G. Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Trans. on VLSI*, vol. 8, no. 3, pp. 299–316, June 2000.

[11] J. Rabaey and M. Pedram, *Low Power Design Methodologies*. Kluwer, 1996.

[12] L. Benini, A. Bogliolo, G.Paleologo, and G. Micheli, "Policy optimization for dynamic power management," *IEEE Trans. on CAD and Sys.*, vol. 18, no. 6, pp. 813–833, June 1999.

[13] Y. Lu, E. Chung, T.Šimunić, L. Benini, and G.D.Micheli, "Quantitative comparison of power management algorithms," in *DATE*, 2000, pp. 20–26.

[14] D. Ramanathan and R. Gupta, "System level online power management algorithms," in *DATE*, 2000, pp. 606–611.

[15] E. Chung, L. Benini, and G. Micheli, "Dynamic power management using adaptive learning tree," in *ICCAD*, 1999, pp. 274–279.

[16] Q. Qiu, Q. Wu, and M.Pedram, "Dynamic power management of complex system using generalized stochastic petri nets," in *DAC*, 2000, pp. 352–356.

[17] E. Chung, L. Benini, A. Bogliolo, and G. Micheli, "Dynamic power management for non-stationary service requests," in *DATE*, 1999, pp. 77–81.

[18] I. Hong, M. Potkonjak, and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processor," in *ICCAD*, 1998, pp. 653–656.

[19] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," in *DAC*, 1999, pp. 134–139.

[20] Y. Shin, K. Choi, and T. Sakurai, "Power optimization of real-time embedded systems on variable speed processors," in *ICCAD*, 2000, pp. 365–368.

[21] R. Golding, P. Bosch, and J. Wilkes, "Idleness is not sloth," in *Proceedings of Winter USENIX Technical Conference*, 1995, pp. 201–212.

[22] M. Srivastava, A. Chandrakasan, and R. Brodersen, "Predictive system shutdown and other architectural techniques for energy efficient programable computation," *IEEE Trans. on VLSI*, vol. 4, no. 27, pp. 42–55, March 1996.

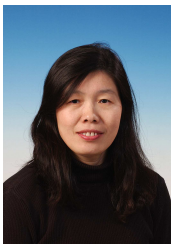
[23] C. Hwang and A. Wu, "A predictive system shutdown method for energy saving of event-driven computation," in *ICCAD*, 1997, pp. 28–32.

[24] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced cpu energy," in *Proceedings of USENIX Symposium on Operating System Design and Implementation*, 1994, pp. 13–23.

[25] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced cpu energy," in *IEEE Annual Found. of Comp. Sci.*, 1995, pp. 374–382.

[26] T.Tia, J. Liu, J. Sun, and R. Ha, *A linear-time optimal acceptance test for scheduling of hard real-time tasks*. Urbana-Champaign, IL: Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign, 1994.
[27] G. Quan and X. S. Hu, "Energy efficient fixed-priority scheduling for real-time systems on voltage variable processors," in *DAC*, 2001, pp. 828–833.
[28] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power optimization of variable voltage core-based systems," in *DAC*, 1998, pp. 176–181.
[29] I. Hong, G. Qu, M. Potkonjak, and M. B. Srivastava, "Synthesis techniques for low-power hard real-time systems on variable voltage processors," in *RTSS*, 1998, pp. 178–187.
[30] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," in *ISLPED*, August 1998, pp. 197–202.
[31] T. Okuma, T. Ishihara, and H. Yasuura, "Software energy reduction techniques for variable voltage processors," *IEEE Design & Test of Computers*, vol. 18, no. 2, pp. 31–41, Mar-Apr 2001.
[32] V. Swaminathan and K. Chakrabarty, "Investigating the effect of voltage switching on low-energy task scheduling in hard real-time systems," in *ASP-DAC*, June 2001, pp. 251–254.
[33] H. Aydin, R. Melhem, D. Mosse, and P. Alvarez, "Determining optimal processor speeds for periodic real-time tasks with different power characteristics," in *ECRTS01*, June 2001.
[34] J. Liu, *Real-Time Systems*. NJ: Prentice Hall, 2000.
[35] G. Quan and X. Hu, "Minimum energy fixed-priority scheduling for variable voltage processors," in *DATE*, 2002.
[36] D. Shin, J. Kim, and S. Lee, "Intra-task voltage scheduling for low-energy hard real-time applications," *IEEE Design and Test of Computers*, vol. 18, no. 2, March-April 2001.

Gang Quan (S'01) received the B.S. degree from the Tsinghua University, Beijing, China, M.S. degree from the Chinese Academy of Sciences, Beijing, China, and PhD degree from the University of Notre Dame, Notre Dame, Indiana. He is currently an assistant professor in the Department of Computer Science and Engineering at the University of South Carolina. His research interests includes real-time systems, low power design, hardware/software codesign, communication network, and reconfigurable computing. He received the Best Paper Award at the 38th Design Automation Conference, 2001.



Xiaobo Sharon Hu (S'85-M'89-SM'02) received her B.S. degree from Tianjin University, China, M.S. degree from Polytechnic Institute of New York, and Ph.D degree from Purdue University, West Lafayette, Indiana. She is Associate Professor in the department of Computer Science and Engineering at University of Notre Dame. She also worked at General Motors Research Laboratories as Senior Research Engineer, and at Western Michigan University as Assistant Professor. Her research interests include hardware-software codesign, real-time embedded systems, low power system design and design automation algorithms. She has published more than 70 papers in the related areas. She has served on the Program Committee of a number of conference such as DAC, ICCAD, DATE, CODES,ICCD, GVLSI. She was the Program Co-Chair of CODES in 2001, and the General Co-Chair of the same conference in 2002. She received the NSF CAREER Award in 1997, and the Best Paper Award at the 38th Design Automation Conference, 2001.