Minimum Energy Fixed-Priority Scheduling for Variable Voltage Processors [*]

Gang Quan   Xiaobo Sharon Hu
Department of Computer Science & Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
{gquan,shu@cse.nd.edu}

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.

1 Introduction

Low power design is an important issue for designing economical, complicated, and safe real-time systems. Since such systems usually have a time-varying computation load, how to appropriately modulate the system capability accordingly, by shutting down or dynamically varying the processor supply voltage, has been a major strategy to design low power systems. Over the past several years, many methods and techniques for minimizing power consumption for such systems has been published, e.g. [2, 9, 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 a question.

Power-reduction techniques in general can be classified into two categories [14]: *dynamic* and *static*. Dynamic techniques are generally easy to implement and applied during run time. Examples of such techniques include [1, 4, 8, 11, 15, 16, 17]. 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 [4], several dynamic voltage scheduling algorithms are proposed for real-time systems containing both periodic tasks and sporadic tasks. These approaches are based on the optimal voltage scheduling algorithm in [19] and the optimal acceptance test in [18]. They are optimal in the senses that the schedule leads to the lowest power 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 the tasks have predefined and fixed priorities, or where the timing information of the sporadic tasks are already known. In [1], a stochastic control approach is proposed. It describes the requests of real-time tasks and the state changes of the system components as discrete-time stationary Markov process. Under such formulation, power management becomes 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 advantage of the assumption that system specifications are known *a priori*. Several static power management policies have been investigated, e.g. [5, 6, 13, 19]. In [19], an static approach is proposed to find the optimal voltage schedule for real-time systems scheduled via the earliest deadline first (EDF) policy. The authors in [5] 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 very special scenarios. In [6], the low power 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

[*]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.

deadlines but different context switching penalties. None of the above approaches can be readily applied to address the optimal voltage scheduling problem for a fixed-priority real-time system.

In this paper, we develop an approach to schedule a fixed-priority (FP) real-time system on a variable speed processor which is guaranteed to consume the lowest possible energy. In our approach, we adjust the deadlines of the real-time jobs by carefully analyzing the preemptive effects among them. Then, we transform the low power fixed-priority scheduling problem into a set of low power EDF-based scheduling problems, and find the optimal voltage schedule based on the work in [19]. In regard to the high computation cost of this transformation, we propose a technique which reduces the computational cost significantly. Several experiments are conducted to compare the performances of some previous scheduling techniques with that in this paper. To our best knowledge, this is the first work ever that has provided the theoretical inherent limit to optimally schedule FP real-time jobs on a variable voltage processor in terms of energy saving.

This paper is organized as follows. Section 2 introduces the necessary background and provides several motivational examples. Section 3 explains our algorithm to find the optimal voltage schedule and section 4 discusses the techniques to improve the computational efficiency of our approach. Section 5 contains the experimental results which show the effectiveness and efficiency of our approach. Finally, section 6 concludes the paper.

2 Preliminaries and motivational examples

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. Among these jobs, we assume that there is no such case as $R_p \leq R_q, D_p > D_q, p < q$, since a lower priority job arriving at or after a higher priority one cannot possibly finish before the higher priority job in a feasible FP system. The processor used in our system is similar to the one introduced in [3]. Note that in [3] the voltage can be varied nearly continuously as long as the number of bits of the speed control register is large enough. We assume that the processor speed is proportional to the supply voltage, and the power is a convex function of the voltage. For simplicity, we assume that the processor speed can be changed instantly. The problem we are interested in 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.*

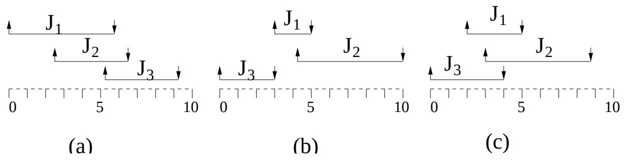

Figure 1. Three real-time system examples.

An intuitive approach for this problem is to apply the EDF-based optimal voltage scheduling algorithm [19] (LPEDF). However, it has been shown in [13] 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 will guarantee the schedulability of the jobs.

Consider the three systems shown in Figure 1, each of which has three jobs. In this figure and the following figures in this paper, we use an up arrow to represent the arrival time of a job, and a down arrow to represent the deadline of the job. Note that, after the timing specifications 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. In Figure 1(a), the FP assignment is the same as that by EDF. The optimal voltage schedule for Figure 1(a) found by applying LPEDF is certainly the optimal voltage schedule under the FP assignment. Furthermore, in some other 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. In Figure 1(b), j_3 has lower priority but earlier deadline than both j_1 and j_2. However, note that, in a feasible schedule, the execution of j_3 never interferes with the execution of j_1 and j_2, since j_3 must finish before the arrival of j_1 and j_2. For this example, the optimal voltage schedule by EDF scheduling is also the optimal schedule by FP based scheduling. Specifically, the job sets in Figure 1(a) and (b) are called *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 $J_p, J_q \in \mathcal{J}$, $p < q$,either $D_p \leq D_q$ or $R_p \geq D_q$.*

For a primary job set, we have the following lemma. (The proofs for the lemmas and theorems in the paper are omitted due to the page limit, interesting readers can refer to [12] for more details.)

Lemma 1 *The optimal voltage schedule for an FP primary job set $\mathcal{J}$ can be determined by applying LPEDF to $\mathcal{J}$.*

According to Lemma 1, we can find the optimal voltage schedule with 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. For such a system, LPEDF no longer produces a valid voltage schedule. This is due to the fact that a lower priority job with earlier deadline might be preempted by other higher priority jobs with later deadlines, and thus make it difficult to meet the deadline with the processor speed computed by LPEDF. How can we find an optimal voltage schedule for this type of systems? In the next section, we introduce a technique to transform a given real-time job sets to a set of primary job sets and identify the optimal voltage schedule for the original job sets.

3 Optimal FP voltage schedule

In this section, we introduce our approach to search for the optimal voltage schedule for an FP real-time system, and provide the theoretical basis for our 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 power consumption among the optimal voltage schedules for a number of primary job sets.

Two questions may arise for our approach: given a real-time job set, how to identify the necessary primary job sets; why the optimal voltage schedule for the original job set is among the optimal voltage schedules for the selected primary job sets. The following definitions and theorem will 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 $1 \leq i \leq N$.*

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 $D_{1i} \leq D_{2i}, 1 \leq i \leq N$, which is denoted by $\mathcal{J}_2 \succeq \mathcal{J}_1$.*

Definition 5 *The non-dominated associative primary (NAP) job sets of $\mathcal{J}$, denoted by $\mathcal{NAP}(\mathcal{J})$, are the job sets such that any $\mathcal{J}_i \in \mathcal{NAP}(\mathcal{J})$ is an associative primary job set of $\mathcal{J}$ and not dominated by any other associative primary job sets of $\mathcal{J}$.*

Theorem 1 *The optimal voltage schedule of $\mathcal{J}$ is the optimal voltage schedule of $\mathcal{J}_i \in \mathcal{NAP}(\mathcal{J})$ which consumes the lowest energy compared with that of any other job set in $\mathcal{NAP}(\mathcal{J})$.*

Since the optimal voltage schedule for any job set in $\mathcal{NAP}(\mathcal{J})$ can be computed with LPEDF (Lemma 1), according to Theorem 1, if we can find all the NAP job sets of $\mathcal{J}$, the voltage schedule with the lowest energy consumption must be the optimal schedule for $\mathcal{J}$. Now the problem becomes how to find all the NAP job sets. There are two challenges to achieve this goal: how to generate the NAP job sets and how to guarantee that all the NAP job sets are identified. The following lemma sheds some lights on constructing the NAP job sets.

Lemma 2 *Let $\mathcal{J}'$ be an associative primary job set of $\mathcal{J}$ and $D_i' = D_i$. Then the following must be true: (i)$D_p' \leq D_i$, if $p < i$ and $R_p < D_i$; (ii) $D_p' \leq R_i$, if $p > i$, and $D_p < D_i$; (iii)$D_p' \leq D_p$.*

With Lemma 2, we devise a procedure to construct the NAP job sets (see Algorithm 1). Specially, we fix, one by one, the deadline of each job, and modify the deadlines for the rest of the jobs to their largest possible values according to Lemma 2. Then, we remove this job and go through this procedure for the rest of the jobs. This procedure continues recursively until the rest of the jobs become a primary job set. Then we put back all the jobs whose deadlines have been fixed to the resultant job sets.

Algorithm 1 Construct the NAP job sets

```
1:  Search_Primary ($\mathcal{J}, \mathcal{T}$)
2:  Input: $\mathcal{J} = (J_1, ..., J_N)$, 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 $J_k \in \mathcal{J}, k = 1 \cdots N$ do
5:    Copy $\mathcal{J}$ to $\mathcal{J}_k$;
6:    Adjust the deadlines of $J_i \in \mathcal{J}_k, i \neq k$ according to
      Lemma 2 with the deadline of $J_k \in \mathcal{J}_k$ fixed;
7:    $\mathcal{J}_k = \mathcal{J}_k - J_k$;
8:    if $\mathcal{J}_k$ is not a primary job set then
9:      Search_Primary($\mathcal{J}_k$, $\mathcal{T}'$);
10:     Add $J_k$ to each job set in $\mathcal{T}'$;
11:     Add $\mathcal{T}_i' \in \mathcal{T}'$ to $\mathcal{T}$ if $\mathcal{T}_i'$ is a primary job set;
12:   else
13:     $\mathcal{J}_k = \mathcal{J}_k + J_k$;
14:     Add $\mathcal{J}_k$ to $\mathcal{T}$ if $\mathcal{J}_k$ is a primary job set;
15:   end if
16: end for
```

The following lemma, together with Theorem 1, can demonstrate that our approach indeed produces the feasible optimal voltage schedule for an FP real-time system.

Lemma 3 *The job sets $\mathcal{T}$ output from Algorithm 1 cover all the NAP job sets of J.*

Figure 2 shows the NAP job sets constructed by applying Algorithm 1 to the system in Figure 1(c). The computation cost for finding the optimal schedule consists of two parts: the cost for constructing the NAP job sets (Algorithm 1), and the cost for searching the optimal schedule among these job sets (LPEDF). Note that the worst-case

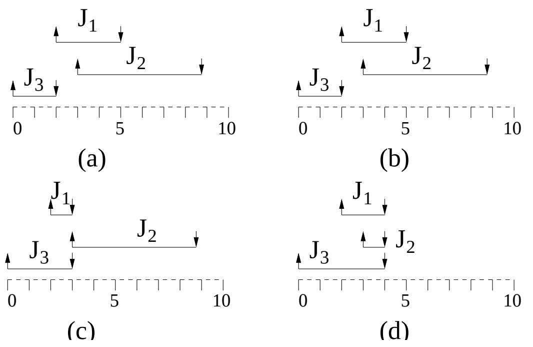

Figure 2. NAP job sets output from Algorithm 1 for the real-time job sets shown in Figure 1(c). (a) is the result by first fixing J_1 to its deadline. (b) and (c) are the results by first fixing J_2 to its deadline. (d) is the result by first fixing J_3 to its deadline.

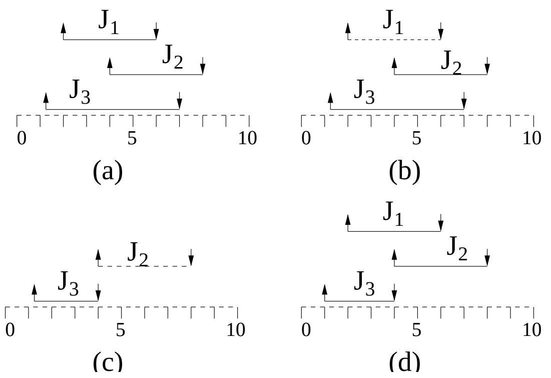

Figure 3. Non-primary job sets may be generated from Algorithm 1. (a) original job sets. (b) fixing the deadline of J_1. (c) fixing the deadline of J_2. (d) putting back J_1 and J_2 and the result is not a primary job set.

complexity of Algorithm 1 is $O(N!)$, where N is the number of jobs, and the complexity of LPEDF is $O(N^2)$ [19]. Therefore, the complexity to search for the optimal voltage schedule is $O(N! + MN^2)$, where M is the job sets output from Algorithm 1. When N increases, searching for the optimal schedule with this approach can be quite time consuming.

4 Improve the computational efficiency

In this section, we discuss our effort to improve the computational efficiency of Algorithm 1. After a careful study, we note that same primary job sets may be constructed more than once by Algorithm 1, and all these primary job sets are evaluated with LPEDF. For example, in Figure 2, (a) and (c) are identical. This is because the same primary job set may be constructed in different recursive calls in Algorithm 1. This situation exasperates when the number of jobs becomes larger. Moreover, we note that not all the job sets constructed from Algorithm 1 are primary job sets. Figure 3 is such an example. In Algorithm 1, we simply discard these job sets. We refer the non-primary job sets and extra copies of the same primary job sets as *redundant job sets*. Even though constructing such redundant job sets has no negative effect to the optimality of the results, they do make the algorithm quite inefficient.

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 large, M can be very high. On the other hand, constructing the redundant job sets is still an unnecessary effort, which can be extremely costly for large job sets. Therefore, we focus our effort on how to *avoid* generating these redundant job sets in the algorithm. The following theorem and corollary can help us greatly reduce the redundant job sets.

Theorem 2 *For job $J_q \in \mathcal{J}$, if there is any job J_p or $J_r \in \mathcal{J}$, $p < q < r$, such that (i) $R_q \le R_r$, and $D_q \le D_r$, or (ii)$R_p \ge D_q$, then the primary job sets constructed by first fixing the deadline of J_q are redundant.*

Corollary 1 *After removing the redundancy defined in Lemma 2, all the job sets output by Algorithm 1 are primary job sets.*

Theorem 2 identifies the recursive calls generating the job sets which can be generated from other recursive calls in Algorithm 1. Corollary 1 guarantees that if the recursive calls identified in Theorem 2 are removed, none of the non-primary job sets will be constructed. Based on Theorem 2 and Corollary 1, an improved algorithm is shown in Algorithm 2. Algorithm 2 is far more efficient than Algorithm 1 since the non-primary job sets are eliminated and the number of identical job sets are significantly reduced. The improvement achieved by Algorithm 2 will be demonstrated through experimental results in the next section.

5 Experimental results

In this section, we use experiments to compare the voltage scheduling approach in this paper (**OPT**) with two other related approaches, the heuristic approach (**VSLP**) introduced in [13], and the approach (**LPFS**) presented in [16].

Algorithm 2 Improved algorithm for searching the NAP job sets

```
1:  Search_Primary2 ($\mathcal{J}, \mathcal{T}$)
2:  Input: $\mathcal{J} = \{J_1, ..., J_N\}$, 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 $J_k \in \mathcal{J}, k = 1 \cdots N$ do
5:      if there is $J_q$, 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;
7:      end if
8:      Copy $\mathcal{J}$ to $\mathcal{J}_k$;
9:      Adjust the deadlines of $J_i \in \mathcal{J}_k, i \neq k$ according to
        Lemma 2 with the deadline of $J_k \in \mathcal{J}_k$ fixed;
10:     $\mathcal{J}_k = \mathcal{J}_k - J_k$;
11:     if $\mathcal{J}_k$ is not a primary job set then
12:         Search_Primary2($\mathcal{J}_k$, $\mathcal{T}'$);
13:         Add $J_k$ to each job set in $\mathcal{T}'$;
14:         Add each job set in $\mathcal{T}'$ to $\mathcal{T}$;
15:     else
16:         $\mathcal{J}_k = \mathcal{J}_k + J_k$;
17:         Add $\mathcal{J}_k$ to $\mathcal{T}$;
18:     end if
19: end for
```

We also use experimental results to show the significant improvement of the computational efficiency achieved by Algorithm 2. All the experiments are conducted using ACER Travelmate 602, with a 650MHz Pentium III processor, 128M memory, and running Windows 98. In our experiments, we assume that the power is a quadratic function of the processor speed.

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 arbitrarily chosen without any special consideration. The execution time of each job is randomly generated from 1 to half of its deadline to make the job sets easier to be schedulable under the maximum processor speed. Only the job sets which are schedulable by our simulation test are used in our experiment. Each group contains at least 50 randomly generated schedulable job sets. To reduce statistical errors, the average power consumption and CPU time for each group using three approaches, *i.e.*, **OPT**,**VSLP**, and **LPFS**, are used to compare their performance. Note, while **VSLP** and **LPFS** do not guarantee to always find the optimal voltage schedule for a given FP real-time system, they do find the optimal ones sometimes. Therefore, we also compare these two approaches in terms of the percentage of optimal solutions they find ($\mathcal{N}_{OPT}/\mathcal{N}$) which are shown in the last two columns in Table 1.

Table 1 shows **OPT** has a much higher computational cost than **VSLP** and **LPFS**, and its CPU times increases rapidly as the number of jobs increase. This agrees with our theoretical analysis since the worst-case complexity is $O(N^3)$ for **VSLP**, $O(N^2 logN)$ for **LPFS**, and $O(N! + Mlog^2 M)$ for **OPT** (where N is the number of jobs, and M is the number of primary job sets constructed in **OPT**). Table 1 also shows that **VSLP** represents an excellent trade-off choice in searching for an optimal voltage schedule. First, from Table 1, the difference between the average power consumption of **VSLP** and that of **OPT** is within 2%, which is much better than that of **LPFS**. Secondly, **VSLP** can find the optimal voltage schedules in most cases, especially when the number of jobs is small. For example, for four jobs, 92% of the optimal schedules can be found by **VSLP**, but **LPFS** finds none. Finally, **VSLP** takes much less CPU time than **OPT**.

Our second experiment quantifies the improvement in terms of computation efficiency made by Algorithm 2. The same randomly generated systems are used. The average CPU times to find the optimal voltage schedules by applying Algorithm 1 or Algorithm 2 to construct the NAP job sets are collected and shown in Table 2. Table 2 shows the dramatic reduction of computational cost by applying Algorithm 2, especially for systems with a large number of jobs. Applying Algorithm 2 reduces not only the effort to construct the redundant job sets, but also the effort to search for the optimal voltage schedule among these sets. Since the number of redundant job sets increases drastically as the number of jobs increases, this explains the dramatic improvement by using Algorithm 2.

6 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 improves on 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, preemptive scheme. Such a scheduling scheme is used widely in many real-world real-time systems due to its simplicity and predictability [7]. 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

Table 1. Experimental results for comparing the three approaches: OPT, VSLP, and LPFS.

No.	Energy			CPU(s)			$\mathcal{N}_{OPT}/\mathcal{N}$	
Jobs	OPT	VSLP	LPFS	OPT	VSLP	LPFS	VSLP	LPFS
2	4.35	4.35	5.74	0.00	0.00	0.00	96%	34%
4	12.8	12.9	19.6	0.00	0.00	0.00	92%	0%
6	26.2	26.9	36.1	0.01	0.00	0.00	82%	0%
8	33.9	34.1	45.7	0.02	0.00	0.00	75%	0%
10	45.5	46.6	57.0	0.12	0.00	0.00	70%	0%
12	49.2	50.1	57.6	0.31	0.00	0.00	75%	0%
14	50.0	50.7	59.8	0.58	0.01	0.00	75%	0%
16	57.0	58.8	68.7	2.78	0.01	0.00	75%	0%
18	57.4	59.1	65.8	6.41	0.02	0.00	75%	0%
20	63.2	65.4	72.4	30.30	0.01	0.00	52%	0%

in [16] to achieve the best overall result. The significance of the results presented here is that the 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.

Table 2. Computation efficiency improvement by reducing the redundancy

No.	CPU Time(s)	
Jobs	Algorithm 2	Algorithm 1
2	0.001	0.001
4	0.002	0.006
6	0.010	0.032
8	0.021	0.402
10	0.121	46.681
12	0.304	3147.070

References

[1] L. Benini, A. Bogliolo, G.Paleologo, and G. Micheli. Policy optimization for dynamic power management. *IEEE Trans. on CAD and Sys.*, 18(6):813–833, June 1999.

[2] L. Benini, A. Bogliolo, and G. Micheli. A survey of design techniques for system-level dynamic power management. *IEEE Trans. on VLSI Sys.*, 8(3):299–316, June 2000.

[3] T. D. Burd and R. W. Brodersen. Design issues for dynamic voltage scaling. *ISLPED*, pages 9–14, 2000.

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

[5] I. Hong, G. Qu, M. Potkonjak, and M. B. Srivastava. Synthesis techniques for low-power hard real-time systems on variable voltage processors. *Proceedings of RTSS*, pages 178–187, 1998.

[6] T. Ishihara and H. Yasuura. Voltage scheduling problem for dynamically variable voltage processors. *ISLPED*, pages 197–202, August 1998.

[7] J. Liu. *Real-Time Systems*. Prentice Hall, NJ, 2000.

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

[9] G. Micheli and L. Benini. System-level low power optimization: Techniques and tools. *Trans. on Design Auto. of Electr. Sys.*, 5(2), April 2000.

[10] M.Pedram. Power minimization in ic design: principles and applications. *ACM Trans. on Design Auto. of Electr. Sys.*, 1(1):3–56, January 1996.

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

[12] G. Quan. *System level design techniques for real-time embedded systems*. PhD thesis, Department of CSE, University of Notre Dame, Notre Dame, IN, USA, 2001.

[13] G. Quan and X. S. Hu. Energy efficient fixed-priority scheduling for real-time systems on voltage variable processors. *DAC*, pages 828–833, 2001.

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

[15] D. Ramanathan and R. Gupta. System level online power management algorithms. *DATE*, pages 606–611, 2000.

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

[17] M. Srivastava, A. Chandrakasan, and R. Brodersen. Predictive system shutdown and other architectural techniques for energy efficient programmable computation. *IEEE Trans. on VLSI Sys.*, 4:42–55, March 1996.

[18] T.Tia, J. Liu, J. Sun, and R. Ha. *A linear-time optimal acceptance test for scheduling of hard real-time tasks*. Technical report, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana-Champaign, IL, 1994.

[19] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced cpu energy. *IEEE Annual Foundations of Comp. Sci.*, pages 374–382, 1995.