Task Scheduling and Voltage Selection for Energy Minimization

Yumin Zhang
Synopsys, Inc.
700 East Middlefield Road
Mountain View, CA 94043
yumin@synopsys.com

Xiaobo (Sharon) Hu Danny Z. Chen
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556, USA
{shu, chen}@cse.nd.edu

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Scheduling

General Terms

Algorithms, Design

Keywords

voltage selection, task scheduling

ABSTRACT

In this paper, we present a two-phase framework that integrates task assignment, ordering and voltage selection (VS) together to minimize energy consumption of real-time dependent tasks executing on a given number of variable voltage processors. Task assignment and ordering in the first phase strive to maximize the opportunities that can be exploited for lowering voltage levels during the second phase, i.e., voltage selection. In the second phase, we formulate the VS problem as an Integer Programming (IP) problem and solve the IP efficiently. Experimental results demonstrate that our framework is very effective in executing tasks at lower voltage levels under different system configurations.

1. INTRODUCTION

Energy consumption has become a primary concern in today's IC systems. Energy minimization techniques applied at higher design levels tend to be more effective than techniques applied at lower levels [4]. Processors that can operate at variable supply voltages or be switched to sleep mode while idling are becoming available [1, 11]. Two main system level energy saving techniques are: voltage selection (VS) (also called voltage scheduling [4]) which selects processor's supply voltage according to the performance requirement, and power management (PM) [2] which shuts down a processor when it is idle. In general VS is more effective than PM [10]. The switching of voltages values can be done on the fly and the incurred overhead can be ignored if such switching does not happen very frequently [16]. In real-time systems where tasks can take tens or hundreds of thousands of cycles to execute, applying VS judiciously can achieve a large amount of energy saving [17].

In this paper, we consider real-time dependent tasks with deadlines. These tasks are to be executed on variable voltage processors. Our challenge is to find a system implementation that consumes the least amount of energy. A system implementation is determined by the ***task assignment*** (which task runs on which processor), the ***task ordering*** (the execution order of tasks on each processor), and the ***voltage selection*** (which cycles of each task use which voltage level). We refer the combination of task assignment and ordering as task scheduling in the rest of the paper.

Most related work, e.g., [3, 10, 12, 13, 17], concentrates on saving energy of independent tasks or on a single processor. However, tasks in real-world applications usually have control or data dependencies and many systems have multiple processors. Approaches in [8, 15] solve the energy minimization problem for dependent tasks on multiple variable voltage processors. [8] assumes a given task assignment, while [15] assumes a given task scheduling. However, task scheduling without consideration of VS may not present the best energy saving potential. Furthermore, the approach in [8], which evaluates eligible tasks one by one, fails to recognize the joint effect of slowing down certain tasks simultaneously. The joint effect is critical in finding global optimal voltage settings and an example in Section 2 explains the effect in much detail. The heuristics in [15] try to distribute slacks evenly among tasks, which does not necessarily lead to best energy saving because the slowdown of different tasks has different effect on other tasks and contributes differently to the overall energy saving. To achieve the best energy saving, the scheduling should try to present the best potential for VS and the VS decision should be made in a global manner.

In this paper, we present a framework that integrates task scheduling and voltage selection together to minimize energy consumption of dependent tasks on systems with a given number of variable voltage processors. The integration is important because scheduling results determine the potential energy saving that can be achieved through VS. In order to find the scheduling with the best slowing down opportunity in the first phase, we apply an earliest-deadline-first (EDF) scheduling which can be proved to be optimal for a single processor, and a scheduling with priority-based task ordering and a best-fit processor assignment for multiple processors. Tasks after scheduling can be modeled as a Direct Acyclic Graph (DAG). In the second phase, we formulate the VS problem on a DAG as an Integer-Programming (IP) problem. Each task can have cycles running at differ-

DAC2002, June 10-14, 2002, New Orleans, Louisiana, USA.

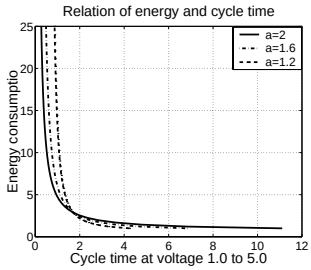

Figure 1: Convex relation of energy and delay

ent voltages. We have implemented the framework and conducted experiments on different systems and tasksets. The results show that our framework is very effective in reducing energy consumption at the system level. Our IP approach for solving the VS problem can slowdown 58% more cycles than a baseline scaling VS approach.

Our main contributions can be summarized as:

First integrated framework: we solve the energy minimization problem of dependent tasks on variable voltage processors in two phases with the first phase task scheduling being guided by the VS approach in the second phase.

Exact algorithm for VS problem: our IP formulation solves the VS problem exactly and we prove the continuous voltage and special discrete voltages case are polynomial-time solvable. The ILP for general discrete voltages is solved with approximation in short runtime and the approximation is tested to be within 97% of the optimal solution.

Substantial energy saving: the framework can slowdown 7-98% of cycles with very short runtime. It can be used in design space exploration to meet energy target and find the best system configuration.

2. PRELIMINARIES AND A MOTIVATIONAL EXAMPLE

The number of cycles, N_u, that task u needs to finish remains a constant during VS, while processor's cycle time CT, u's delay d_u and the the dominant part of total energy consumption, dynamic energy consumption, E_u, change with the supply voltage V_{dd}. CT, d_u and E_u can be computed as

$$CT = \frac{k \cdot V_{dd}}{(V_{dd} - V_{th})^a} \quad (1)$$

$$d_u = N_u \cdot CT = N_u \cdot \frac{k \cdot V_{dd}}{(V_{dd} - V_{th})^a} \quad (2)$$

$$E_u = N_u \cdot C_u \cdot V_{dd}^2 \quad (3)$$

where k is a device related parameter, V_{th} is the threshold voltage, a ranges from 2 to 1.2 and C_u is the effective switching capacitance per cycle. We depict the relation of E_u and d_u for different a in Figure 1. It is clear that E_u is a convex function of d_u [12] and both are functions of V_{dd}. Note that our work allows tasks to have different power characteristics, such as switching capacitance.

We use a DAG to represent a taskset. In the DAG, each node u represents a task u, while an edge $e(u, v)$ indicates that v can only start after u finishes. Deadline dl_u constrains u's finish time and T_{con} on a set restricts all tasks' finish time. A taskset after scheduling can still be represented as a DAG with additional edges, if necessary, to capture processor sharing. The DAG for a 5-task set before and after scheduling to two processors is shown in Figure 2 (a) and (b). **Slack** is the maximum amount of time that a task can be slowed down without violating timing constraints.

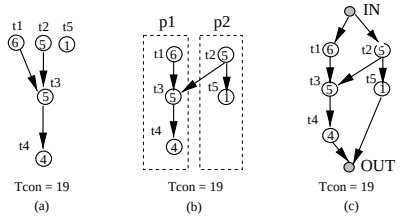

Figure 2: (a) A 5-task set (b) Tasks scheduled on P_1 and P_2 (c) DAG with IN and OUT

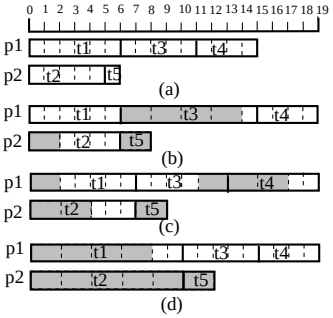

Figure 3: VS results by different approaches. Cycles in shade are executed at V_l.

2.1 A motivational example

We use the example in Figure 2 to show that different VS approaches on multiple processors perform quite differently. This example motivated us to make VS decisions in a global manner. In Figure 2 (b), the number inside each task is N_u, the number of cycles needed to finish the task. Assume P_1 and P_2 can operate on V_h and V_l and $V_h > V_l$. For simplicity, assume CT_{V_h} is 1 time unit, CT_{V_l} is 2, and the average energy saving per slowdown cycle is the same for all tasks on P1 and P2. Let T_{con} be 19.

The VS results by approaches in [8, 15] are shown in Figure 3 (b) and (c). Figure 3 (a) shows the execution of tasks on P_1 and P_2 when the two processors are always on V_h and Figure 3 (d) shows an optimal solution for this problem. It is clear that slowing down t_1 and t_2 simultaneously is the best even though slowing down each will take away the potential of slowing down t_3, t_4 and t_5. Also the even distribution of slack is not optimal in saving energy. The number of cycles at V_l is 10 in Figure 3 (d) and this is an optimal solution, which is is 67% more than the 6 V_l cycles by approach in [8] and 43% more than the 7 V_l cycles by approach in [15]. Our VS approach, which is to be presented in the next section, achieves the optimal solution for this example.

3. VOLTAGE SELECTION

Our approach integrates task scheduling and voltage selection together and strives to achieve the maximum energy saving on variable voltage processors. The framework can be used to optimize energy at the system level in a design

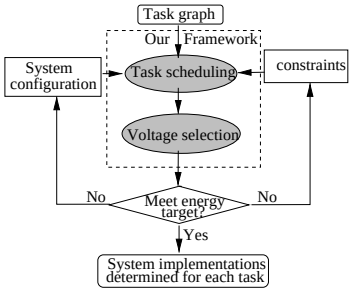

Figure 4: Design flow with our approach

flow like the one shown in Figure 4. In this section, we will present an IP formulation that solves the VS problem on a given scheduling of dependent tasks on multiple processor. A simplified version of the IP formulation can be used to solve the VS problem on a single processor. The IP formulation solves the VS problem exactly and it provides guidance for task scheduling to be discussed in Section 4.

3.1 Unified IP formulation for VS

We need to guarantee that after the VS technique is applied, tasks with deadlines still finish before their deadlines and the last task on each processor finishes no later than T_{con}. That is the delay on each path in the DAG should still not be greater than T_{con}. However, the number of paths in a DAG can be exponential to the number of edges and makes it not practical to do optimization on a path base. This is the reason that VS approaches on single processor systems cannot be readily extended to solve the VS problem on multiple processors. Here we adopt a model proposed in [5] to formulate timing constraints on a DAG not on paths, but on nodes and edges. We add two nodes, IN and OUT, edges from IN to the first task on each processor, and edges from the last task on each processor to OUT to form a new DAG. Figure 2 (c) shows the new DAG, $G(V, E)$, for the taskset in Figure 2 (a). Execution time of task u at the highest supply voltage V_h is a constant T_u, and its deadline dl_u remains a constant. T_{OUT} and T_{IN} are set to be 0. T_{con} is the timing constraint on the DAG. Besides task's delay d_u, we associate another variables D_u with u. Timing constraints on $G(V, E)$ in Figure 2 (c) can be modeled as

$$D_{OUT} - D_{IN} \leq T_{con} \quad (4)$$

$$D_v - D_u - d_u \geq 0 \quad \forall e(u,v) \in E \quad (5)$$

$$D_u + d_u \leq dl_u \ \forall u \text{ with deadline} \quad (6)$$

$$d_u \geq T_u, \ D_u \geq 0, \text{ int}, \ \forall u \in V \quad (7)$$

Intuitively, D_u represents u's start time. For a feasible scheduling, if D_{IN} is set to be 0, the above constraints guarantee that tasks with deadlines will finish before their deadlines and the finish time of all task is not greater than T_{con}.

Single processor systems are a special case of systems with multiple processors. The timing constraints in (4)-(7) on multiple processors also constrain the timing on a single processor. However, the scheduling on a single processor has special features that we can utilize to simplify the constraints. The main feature on a single processor is that all tasks are on one path. The constraints for single processor can be simplified as follows.

$$\sum_{u \in V} d_u \leq T_{con} \quad (8)$$

$$\sum_{v \in B_u} d_v \leq dl_u \ \forall u \text{ withdeadline} \quad (9)$$

$$d_u \geq T_u, \text{ int}, \ \forall u \in V \quad (10)$$

where B_u is the set that includes u and all tasks scheduled before u, d_u's are variables whose values need to be determined, T_{con}, dl_u and T_u are constants.

The objective of VS is to minimize the sum of each task u's energy consumption, $\sum_{u \in V} E_u$. Combing the objective and the constraints in (4)-(7) for multiple processors and constraints in (8)-(10) for single processor, we have the IP formulation for the VS problem. To trade the increase of delay for energy saving, we need to establish the relationship between d_u and E_u. If the supply voltage can change continuously, d_u can also change continuously. In this case, E_u is a convex function of d_u, as depicted in Figure 1, and we have $E_u = f(d_u)$, where $f(\cdot)$ is a convex function. Substituting E_u with $f(d_u)$, we have the IP formulation for the continuous voltage case.

In the discrete voltage case, only a certain number of voltages are available. For example, the Athlon 4 Processor from AMD [1] can operate at five voltage levels. The discreteness of voltage levels implies that a task's delay can only take discrete values. Let the highest voltages be V_h and other m available voltages be $V_1, V_2, \ldots, V_m$, where $V_i < V_h, 1 \leq i \leq m$. We introduce m new variables, $N_{u,i}$, to represent the number of cycles that task u is executed at V_i. For any given V_i, cycle time CT_i and energy consumption per cycle $E_{u,i}$ are constants as computed in (1) and (3) in Section 2. The total number of cycles u takes, N_u, remains a constant. d_u and E_u are computed as linear functions of $N_{u,i}$.

$$d_u = T_u + \sum_{i=1}^{m} N_{u,i}(CT_i - CT_h) \quad (11)$$

$$E_u = C_u \cdot (\sum_{i=1}^{m} N_{u,i} V_i^2 + (N_u - \sum_{i=1}^{m} N_{u,i}) \cdot V_h^2) \quad (12)$$

Substituting (11)-(12) as E_u and d_u and adding $\sum_i N_{u,i} \leq N_u$, we have the ILP for the discrete voltage case, where D_u's and $N_{u,i}$'s are variables.

3.2 Solving the IP problem

In general, IP problems are hard to solve. But the IP problem for the continuous voltage case on systems with a given number of processors is polynomial time solvable. A convex programming solver, like *COPL_LC* from [6], can be used to solve the problem in polynomial time.

THEOREM 1. *The VS problem of executing dependent tasks on systems with one or more processors with continuous voltage is polynomial time solvable.*

Proof: The VS problem is formulated as an IP problem with objective $\sum_u f(d_u)$, constraints in (4)-(7) for multiple processors, and constraints in (8)-(10) for a single processor. The objective function is the sum of convex functions, while the coefficients of variables D_u and d_u in (4)-(7), and variable d_u in (8)-(10) are 1, 0 or -1. Thus the IP formulation has a totally unimodular constraint matrix for both single processor and multiple processors. An IP problem is polynomial time solvable if the objective function is a separable convex function and constraint matrix is totally unimodular [9]. The IP formulation for the continuous voltage case has the same form as such a polynomial time solvable IP problem and thus the VS problem on a given number of processors that can operate on continuous voltage is polynomial time solvable. □

The ILP problem for some special discrete values is also polynomial-time solvable on a single processor and multiple processors. If there is only one lower voltage V_l and $CT_{v_h} - CT_{V_l} = 1$, the ILP problem for one processor and multiple processors both have a totally unimodular constraint matrix. Such an ILP problem is polynomial time solvable. An LP solver, such as *LP_SOLVE* from [7], can be used to get the integer solutions in polynomial time. To solve the ILP problem for general discrete voltages in short runtime, we employ an efficient approximation. First the integer constraint on variables is relaxed and the correspond-

ing LP problem is solved. Then we use the floor integer of the non-integer solutions for $N_{u,i}$ in the LP as the integer solutions for $N_{u,i}$ in the ILP to guarantee that the approximation satisfies the timing constraints. This approximation is tested to be very efficient by experiments in Section 5.

4. TASK SCHEDULING

Task scheduling should provide the maximum slowing down potentials for voltage selection to utilize for energy saving. In this section, we present the scheduling that is guided by our VS approach presented in the last section. For a single processor, we apply an EDF scheduling, which can be proved to be optimal in providing slowdown opportunities. For multiple processors, we use a priority-based scheduling in order to provide the best energy saving opportunity.

4.1 Scheduling on a single processor

It is clear that individual task's deadline imposes direct constraint on the design space of delays of tasks that are scheduled before it, as shown in constraints in (9). Energy saving is affected by the design space of tasks' delays because the delay will be traded for energy saving. To maximize the design space of delays, tasks with earlier deadlines should be scheduled earlier in order to avoid imposing constraints on more tasks. We apply an EDF scheduling in the following recursive algorithm $EDF(S,t)$. The input to the algorithm is the taskset S, and t is initialized to be 0. Denote PRE_u as the set of u's predecessors. The output of the algorithm is the same taskset with each task's start time s_u being assigned and the order on the processor being decided.

EDF(S, t)
Begin
  If S $\neq \phi$
    If $\exists$ u with dl_u
      Find u with $dl_u = \min\ (dl_v | \forall v \in S)$;
    Else Find a $u \in$ S;
    Endif
    $S1 = PRE_u$; $S2 = S - S1 - \{u\}$;
    EDF(S1, t);
    $s_u = t$; $t = t + T_u$;
    EDF(S2, t);
  Endif
End

We have proved that the EDF scheduling provides the biggest solution space for the single processor VS problem.

THEOREM 2. *The EDF scheduling is optimal for the VS process to save energy on a single processor*

Due to space limit, the proof is omitted. The EDF scheduling is feasible if a feasible scheduling exists [14]. Combining the EDF scheduling which is optimal and our IP formulation that solves the VS problem exactly, our framework solves the general energy minimization problem of dependent tasks on a single variable voltage processor exactly.

4.2 Scheduling on multiple processors

The EDF scheduling is optimal on a single processor, but not on multiple processors because tasks will be on multiple paths and affect tasks on these paths differently. In the following example, the scheduling that does not follow the EDF ordering presents more slowdown opportunities. Consider the taskset in Figure 5 (a), an EDF scheduling in Figure 5 (b), and another scheduling that does not follow

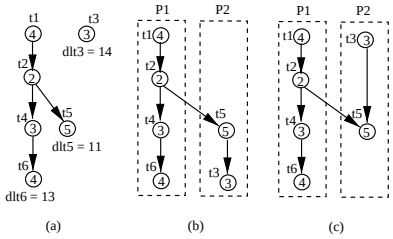

Figure 5: Scheduling by different approaches

the deadline order in Figure 5 (c). Using our IP formulation on the same system used in the example in Figure 3, the optimal number of cycles at V_l is 0 and 3 for the scheduling in Figure 5 (b) and (c), respectively. This example tells us that other information needs to be combined with deadline to find a scheduling that provides the best slowdown opportunity in multiprocessor systems.

To provide more energy saving opportunities in the scheduling for VS to utilize, we should try to avoid putting tasks on unnecessary paths which might pose tighter constraints, making unnecessary long paths and putting unrelated tasks on the same path, and try to let less tasks constrained by smaller deadlines. To achieve these goals, we use a priority to select one task at each scheduling step, find a best-fit processor for the selected task, and repeats the process until all tasks are scheduled.

The priority relates to a task's deadline, dependencies and the usage of processors in the system. Tasks are assigned a latest finish time that they must meet for them or their successors to meet deadlines. Task u's latest finish time lft_u is defined as

$$lft_u = \min(dl_u, lft_v - T_v | \forall v, e(u,v) \in E) \qquad (13)$$

Leaf task's latest finish time is its deadline or T_{con} if it does not have a deadline. Start from leaf tasks and traverse the taskset DAG backward, we can assign each task a latest finish time. Denote task u's ready time when all u's predecessors finish as r_u, its earliest start time when it is ready and there is a processor available as es_u, the number of processors in the system as N, and processor P_i's available time as a_{P_i}. r_u, es_u and a_{P_i} are updated during the scheduling.

We build the scheduling step by step. At each scheduling step, eligible tasks's priorities, PRI_u, are evaluated.

$$PRI_u = lft_u + es_u \qquad (14)$$

$$es_u = \max(r_u, \min\ (a_{P_i} | \forall i \leq N)) \qquad (15)$$

Task u with the smallest PRI_u is assigned to a best-fit processor, which is selected according to the following three steps. s_u, and a_{P_i} are set accordingly:

1. select P_i **if** $a_{P_i} = r_u$. That is to select the processor that is available the same time when u is ready. In this case, $s_u = r_u = a_{P_i}$ and $a_{P_i} = s_u + T_u$. If there is no such P_i,

2. select P_i **if** $a_{P_i} < r_u$, $a_{P_i} > a_{P_j} | \forall j \neq i$, $a_{P_j} < r_u$. That is to select the processor that becomes available the latest among all processors available before u is ready. In this case, $s_u = r_u$ and $a_{P_i} = s_u + T_u$. If there is no such P_i,

3. select P_i **if** $a_{P_i} \leq a_{P_j}$, $\forall j \neq i$. That is to select the processor that is available the earliest. In this case, $s_u = a_{P_i}$ and $a_{P_i} = s_u + T_u$.

After the selection and assignment of one task, a new set of eligible tasks are evaluated and another task is selected and assigned to its best-fit processor. The step repeats until all

tasks are scheduled. If the priority-based scheduling is not feasible, we can increase the weight of the latest finish time in the priority function and try to get a feasible scheduling.

5. EXPERIMENTAL RESULTS

To evaluate the effectiveness of our framework in saving energy, we implemented it and conducted experiments on various tasksets and systems. 9 tasksets are generated with *TGFF* [18] where each set consists 10 to 500 tasks. Since single processor systems are a special case of multiprocessor systems and we have proved that the EDF scheduling on a single processor is optimal, we concentrate the experiments on multiprocessor systems. We tested systems consisting of 2 to 8 processors that can operate at 2 and 4 different voltages. (Most commercially available variable voltage processors have 2 to 5 voltage levels.) First, we investigate the effectiveness of our IP approach for solving the VS problem by comparing the number of slowdown cycles with a scaling VS approach on the same scheduling. Then we test how our scheduling helps saving energy by comparing the VS results on different scheduling. The energy saving by our framework on different system configurations are also investigated.

The first experimental system consists of 5 processors that can operate at two different voltages, V_h and V_l. For simplicity, we assume that $CT_{V_l} = 4$, $CT_{V_h} = 1$, the energy saving per cycle at V_l is the same for all tasks. (Note there is no such requirement in our IP formulations.) Thus the total number of slowed down cycle SN represents the energy saving that is proportional to $SN(V_h^2 - V_l^2)$.

We use our scheduling in the first phase and alternate different VS approaches in the second phase to test how much improvement our IP approach provides. Since the approach in [8] also deals with task ordering and the approach in [15] has other objectives during their VS process, we feel it is not an apple-to-apple comparison to compare our IP approach with the two directly. We compare the number of cycles at V_l by our IP approach with a scaling VS approach that is also discussed in [8]. It scales down the delay of all tasks by the ratio of timing constraint over critical path length on a given scheduling. The approach in [15] which distributes slack evenly shares the same spirit with the scaling approach. Table 1 summarizes the performance of our IP approach and the scaling approach. Column NT, NC, T_{con} and T_{cri} show the number of tasks, number of task cycles, timing constraints and critical path delay at V_h, respectively. Timing constraint T_{con} is set to be 1.5 times of T_{cri}. Column SC and IP show the number of cycles at V_l by the scaling approach and our IP approach, and Column IMP shows the improvement of number of cycles at V_l by our IP approach (IP) over the scaling (SC), which is computed as $((IP) - (SC))/(SC)$. It is clear that on average our approach can execute 58% more cycles at V_l and save more energy than the scaling approach.

The approximation we use to get the integer solutions for the ILP problem is very effective. The objective of the ILP, total number of slowdown cycles, after the approximation of integer solutions is within 97% of the objective achieved with non-integer solutions from solving the LP. The objective of the LP is the upper limit of the objective that the corresponding ILP can achieve. This means that solutions by the approximation is within at least 97% to the optimal solution of the ILP problem. The whole process including scheduling, solving LP and approximation finishes within seconds for all tasksets.

Table 1: Performance by scaling and our approach

set	NT	NC	T_{cri}	T_{con}	SC	IP	IMP %
s1	9	81	49	74	11	37	236
s2	50	422	111	167	51	104	104
s3	101	922	243	365	116	225	94
s4	151	1501	464	696	188	413	120
s5	213	2988	722	1083	413	644	56
s6	245	2871	652	978	376	557	48
s7	305	2643	572	858	314	495	58
s8	463	4310	910	1365	541	755	40
s9	514	4633	949	1424	556	808	45
ave.	228	2263	519	779	285	449	58

Table 2: Performance on different scheduling

set	NT	NC	T_{con}	EDF	PEDF	IMP %
s2	50	422	167	89	104	17
s3	101	922	365	200	225	13
s7	305	2643	858	432	495	15
ave.	152	1329	463	240	275	14

To test how our scheduling helps providing more slowdown opportunities, we alternate two scheduling approaches in the first phase, one is our priority-based ordering and best-fit processor selection approach (PEDF), and another is a baseline approach that uses an EDF ordering and selects the processor that becomes available the earliest (EDF). Our IP formulation is applied in the second phase to minimize energy. The number of cycles at V_l for several representative tasksets on the same 5-processor experimental system used in the VS comparison are summarized in Table 2. In the table, Column EDF and PEDF show the number of cycles at V_l on the baseline scheduling and on our priority-based scheduling, respectively, and Column IMP shows the improvement of cycles at V_l on our scheduling (PEDF) over the baseline scheduling (EDF). The results show that our scheduling can provide 14% more slowing down opportunities than a baseline scheduling.

It is clear that our first-phase priority-based scheduling and second-phase IP approaches all outperform other respective approaches. Here we want to explore how the framework performs on different system configurations. We first investigate how the number of processors affects the energy saving. With fixed timing constraints, systems with more processors should have shorter critical path length and more tasks with more slacks. The slack can be traded for energy saving. Table 3 shows the number of cycles at V_l on systems with different number of processors. Processors can operate at two values and the cycle time changes from 1 unit to 4 unit when voltage is switched from V_h to V_l, which is the same as in the first two experiments. The energy saving increases dramatically when the number of processors increases from 2 to 5, but not so much when the number increases from 5 to 8. This is because the limited parallelism among tasks has been exploited by the increases of number of processors from 2 to 5. Additional processors do not provide much benefit if there are not many tasks can run in parallel. If tasks are independent, every task can run in parallel and we should expect the increase of number of processors lead to energy saving increase more evenly.

The number of voltages and voltage values also affect the energy saving. With more voltages available, the search space for VS problem is bigger and energy savings should

Table 3: Savings on different number of CPUs

set	NC	T_{con}	2 cpus	5 cpus	8 cpus
s1	81	66	9	31	31
s2	422	266	29	268	397
s3	922	581	67	588	803
s4	1501	989	118	905	1318
s5	2988	1861	208	1939	2910
s6	2871	1760	197	1862	2840
s7	2643	1601	177	1728	2640
s8	4310	2616	287	2832	4310
s9	4633	2789	310	3076	4633
ave.	2263	1392	156	1470	2209
per.	100%		7%	65%	98%

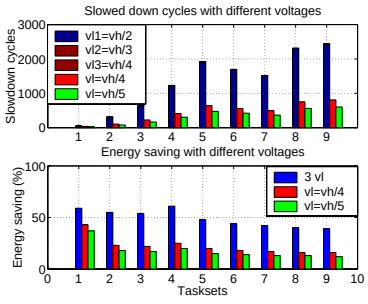

Figure 6: Energy saving of different voltages

be higher. For different voltage values, a lower value increases the cycle time more and may prevent other cycles from slowing down if there is not enough slack. However, the energy saving per cycles is more for the lower voltage value. It is not immediately clear how voltage values affect the overall energy saving. We did experiments on the same 9 tasksets on three 5-processor systems. The configurations for the three systems are Sys1: $V_{l1} = V_h/2$, $V_{l2} = V_h/3$, and $V_{l3} = V_h/4$, Sys2: $V_l = V_h/4$, and Sys3: $V_l = V_h/5$. Assume that cycle time increases from 1 to 2, 3, 4 and 5 when voltage changes from V_h to $V_h/2$, $V_h/3$, $V_h/4$, and $V_h/5$. This is a rough estimation but it should not make any conclusion we draw invalid. The number of slowdown cycles and the energy saving in the three systems are shown in the upper and lower part in Figure 6. In the figure, the first bar corresponds to data for Sys1, the second for Sys2, and the third for Sys3. The results show that our approach is capable of utilizing the bigger design space provided in Sys1 and achieves more energy savings when more voltages are available. In Sys1, where processors can operate at three different levels, most cycles are slowed down to operate at V_{l1}, which is closest to V_h. The number of cycles operating at the other two lower voltages are so few that they are invisible in Figure 6. Sys2 with $V_l = V_h/4$ has more cycles slowed down and saves more energy than Sys3 with $V_l = V_h/5$. This is because that the closer a V_l is to V_h, the more cycles can be slowed down and results in more energy saving even though the saving per cycle is smaller for a V_l that is closer to V_h. Of course, if timing constraint is so loose that every cycle can be executed at the lowest voltage, then the lower voltage, the more energy saving. Continuous voltage is the ideal case and should give the best energy saving.

6. CONCLUSION

In this paper, we present a two-phase framework that integrates task scheduling and voltage selection together to achieve the maximum energy saving of executing dependent tasks on one or multiple variable voltage processors. In the first phase, the EDF scheduling for a single processor is optimal in providing slowdown opportunities and the priority-based scheduling for multiple processors provide more slowing down opportunities than a baseline scheduling. Our IP formulation in the second phase is the first exact algorithm for the VS problem. The IP can be solved optimally in polynomial-time for the continuous voltage and some special discrete voltage cases, or solved efficiently by our approximation for general discrete voltages. Experimental results show that our framework can slow down 7-98% cycles in very short time. Due to its high quality solutions and low computational cost, our framework can be used for design space exploration.

7. ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under Grant CCR-9988468 and MIP-9701416.

8. REFERENCES

[1] AMD Athlon 4 Processor, *Data Sheet Reference #24319*, Advanced Micro Devices, Inc., 2001.

[2] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on VLSI systems*, June 2000, pp. 299-316.

[3] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A dynamic voltage scaled microprocessor system," *IEEE Journal of Solid-State Circuits*, vol. 35, 2000, pp. 1571-1580.

[4] A. Chandrakasan and R. Brodersen, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, 1995.

[5] W. Chuang, S. Sapatnekar, and I. Hajj, "Delay and area optimization for discrete gate sizes under double-sided timing constraints," *CICC'93*, pp. 9.4.1-9.4.4.

[6] *http://dollar.biz.uiowa.edu/col.*

[7] *ftp://ftp.es.ele.tue.nl/pub/lp_solve.*

[8] F. Gruian, K. Kuchcinski, "LEneS: task scheduling for low-energy systems using variable supply voltage processors," *ASP-DAC'01*, pp. 449-455.

[9] D. Hochbaum and J. Shanthikumar, "Convex separable optimization is not much harder than linear optimization," *Journal of ACM*, vol. 37, No. 4, 1990, pp. 843-862.

[10] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. Srivastava, "Power optimization of variable voltage core-based systems," *DAC'98*, pp. 176-181.

[11] *http://www.chips.ibm.com:80/products/powerpc/chips.*

[12] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," *ISLPED'98*, pp. 197-202.

[13] Y. Lin, C. Hwang and A. Wu, "Scheduling techniques for variable voltage low power designs," *ACM Transaction on Design Automation of Electronic Systems*, vol. 2, no. 2, 1997, pp. 81-97.

[14] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, 1973, pp. 46-61.

[15] J. Luo and N. Jha, "Power-conscious joint scheduling of periodic task graphs and a periodic tasks in distributed real-time embedded systems," *ICCAD'00*, pp. 357-364.

[16] N. Namgoong, M. Yu, and T. Meng, "A high-efficiency variable-voltage CMOS dynamic DC-DC switching regulator," *ISSCC'97*, pp. 380-381.

[17] G. Quan and X. Hu, "Energy efficient fixed-priority scheduling for real-time systems on voltage variable processors," *DAC'01*, pp. 828-833.

[18] *http://helsinki.ee.Princeton.EDU/ dickrp/tgff*