Abstract:
Energy consumption has become an important aspect while designing a portable embedded system. Dynamic voltage scaling (DVS) is an energy saving technique, reducing energy dissipation of the core by lowering the supply voltage and operating frequency, so that battery life of portable devices can be extended. In this paper it is discussed about a simulation environment to test various DVS algorithms because there are no standard simulators readily available in market. The simulator environment provides a framework for objective performance evaluations of various DVS algorithms. It is compared several key DVS algorithms recently proposed for hard real-time periodic task sets, analyze their energy efficiency, and discuss the performance differences quantitatively. We show through simulations these real time DVS algorithms closely approach the theoretical lower bound on energy consumption, and can easily reduce energy consumption 20% to 30% in an embedded real-time system.

Key words: EDF, RM, RT-DVS, Utilization Factor, Energy consumption

1 Introduction
In the past, computers were judged mainly on two criteria: price and performance. Lately, however, energy consumption has gained increasing importance, due to several factors. Portable computers, including notebooks and palm-sized devices, have gained popularity. These computers are limited in their operating time by the amount of energy in their batteries. And, unlike most properties of computers, battery capacity per unit weight has improved little in recent years and shows little indication of substantial improvement in coming years. Processor energy management involves switching between processor states of variable power. Modern processors have two main types of low-power states: sleep and reduced voltage. Essentially all modern processors, even ones not designed for mobile use, have sleep states.

DVS as an effective low-power design technique, many DVS algorithms have been proposed to several commercial variable-voltage microprocessors, especially targeting for hard real-
time systems [2,3,4]. Although proposed DVS algorithms are shown to be effective in reducing the energy consumption of a target system under their own experimental scenarios, these algorithms have not been quantitatively evaluated each other under a unified evaluation framework. The lack of comprehensive evaluation studies makes to that of the existing DVS algorithms. In this paper, we describe DVS simulator, an integrated simulation environment for DVS algorithms, which can be used in comparing the energy efficiency of various DVS algorithms. In the next section, it is briefly explained DVS algorithm for real time embedded system. Section 3 describes our simulator and the assumptions made in its design. Simulation studies and result analysis are shown in section 4. Conclusions and future directions are given in last section.

2 Existing DVS Algorithms

Research into DVS algorithms can be classified into two categories: (1) Algorithms that attempt to estimate the future utilization of the processor based on the past information and (2) Algorithms that use task deadlines to guide performance setting decisions. The disadvantage of the former is its unresponsiveness to dynamically changing workloads, but simpler in their implementation. For hard real-time systems, there are two types of voltage scheduling approaches depending on the voltage scaling granularity: intra-task DVS and inter-task DVS. The intra-task DVS algorithms [17] adjust the voltage within an individual task boundary, while the inter-task DVS algorithms determine the voltage on a task-by-task basis at each scheduling point. The main difference between two approaches is whether the slack times are used for the current task or for the tasks that follow. Inter DVS algorithms distribute the slack times from the current task for the following tasks, while Intra DVS algorithms use the slack times from the current task for the current task itself.

Three heuristics were brought up by Pillai and Shin working with RM or EDF task scheduling [5-8]. The first one, static scheduling (static_EDF), selects only one lowest possible operating frequency to let all tasks meet all the deadlines. The second one, cycle-conserving scheduling (cc_EDF), determines the lowest frequency for each schedule task satisfying the acceptance test. In the acceptance test, the bound of the total utilization is decreased to the optimum speed of the system. Then, the system updates the actual utilization, according to the full speed, that the previous task used in order to calculate the next task speed. The last heuristic, look-ahead scheduling (la_EDF), tries to spread all tasks backwards and considers the future tasks simultaneously. The simulation results showed that the look-ahead scheduling is the best among three heuristics in almost all cases.

3 DVS Simulator

We developed a simulator for the operation of hardware capable of voltage and frequency scaling with real-time scheduling. It supports all the inter DVS algorithms and can be used to compare the energy efficiency of different inter DVS algorithms using the same task set specification under the same machine configuration. Using this evaluation, one can decide the best DVS algorithm for the given application on the given hardware platform. It can be used as well when evaluating a given DVS algorithm under various evaluation conditions. The EDF scheduler without any DVS support is also simulated for comparison. The following subsection describes our simulator and the assumptions made in its design. Later, we show some simulation results and provide insight into the most significant system parameters affecting RT-DVS energy savings.

3.1 Simulator

It requires three inputs a task set specification, duration of simulation and a machine specification. The task set specification describes various task set characteristics that affect the energy efficiency of a DVS algorithm while the machine specification describes the machine characteristics that affect the energy efficiency of a DVS algorithm. The Inter DVS Module is responsible for the whole operation of simulator.

3.2 Task set specifications

Task set specification the energy-efficiency of DVS algorithms can be affected by the characteristics of a given task set such as the number of tasks, the task execution time distributions and the worst-case execution time (WCET). Therefore, when evaluating DVS algorithms, it is necessary to understand how the performance of the DVS algorithms varies depending on task sets with different characteristics.

In simulator the real-time task sets (Ti) are specified using a pair of numbers for each task by indicating its worst-case computation time (Ci) and period (Pi). In order to automatically generate a task set with specific characteristics, the Task set
A generator is used. The Task set generator takes inputs as given in Table 1 and generates as an output the corresponding script file for a task set satisfying the requirements. The Task set is schedulable[5,6,7].

Table 1: Task set generator inputs

<table>
<thead>
<tr>
<th>Task(Ti)</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation time(Ci) ms</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Period(Pi) ms</td>
<td>8</td>
<td>10</td>
<td>14</td>
</tr>
</tbody>
</table>

3.3 Machine Specifications

We assume a simulated machine specification which includes the available voltage and clock levels of a target variable-voltage processor [11-15]. The machine specification reflects the characteristics of the target variable-voltage processor. Using simulator with the target task sets and DVS algorithms fixed, the DVS-related architectural exploration is possible in designing variable-voltage processors. If necessary, other machine specifications are easily supported.

3.4 Inter DVS Module

The Inter DVS Module is responsible for scheduling tasks plays a role of a real-time scheduler in a hard real-time system. The priority-based scheduling can be implemented by maintaining two queues, one called run queue and the other called delay queue[16]. The run queue holds tasks that are waiting to run and the tasks in the queue are ordered by priority. The task that is running on the processor is called the active task. The delay queue holds tasks that have already run in their periods and are waiting for their next periods to start again. They are ordered by the time at which their release is due. When the scheduler is invoked, it searches the delay queue to see if any tasks should be moved to the run queue. If some of the tasks in the delay queue are moved to the run queue, the scheduler compares the active task to the task at the head of the run queue. It takes as an input a task specification for periodic tasks and simulates each task based on the specified scheduling policy (e.g., RM or EDF). The Inter DVS Module consists of three sub modules one for estimating available slack times, one for execution of task and the another one for estimating energy consumed by a task in a particular invocation. The slack estimation is done by the Slack Estimation Module, which computes the total available time for the scheduled task while the Task Execution Module, which determines the operating speed for

4 Simulation and Result Analysis

Using C++ and VB we have developed a simulator for the operation of hardware capable of voltage and frequency scaling with real-time scheduling. The following subsection describes the assumptions made in its design. Later, we show some simulation results and provide insight into the most significant system parameters affecting RT-DVS energy savings.
4.1 Assumptions
1. The tasks are periodic and their period is same as their worst case deadlines
2. The switching between frequency levels may occur anywhere within the task and the switching overheads are negligible in comparison with the task deadlines
3. The scheduler follows the Early Deadline First (EDF) Scheme
4. For simplicity only task execution and idle (halt) cycles are considered.

4.2 Simulation results and discussions
In the following simulations, we compare RT-DVS algorithms to each other and to a non-DVS system under various evaluation conditions like different task sets, machine configurations, worst case execution times and idle factors. We also include a theoretical lower bound for energy dissipation. This lower bound reflects execution throughput only and does not consider any timing issues (e.g., whether any task is active or not). It is computed by taking the total number of task computation cycles in the simulation and determining the absolute minimum energy with which these can be executed over the simulation time duration with the given platform frequency and voltage specification. No real algorithms can do better than this theoretical lower bound.

From simulations results it is observed that the RT-DVS algorithms show potential for large energy savings, particularly for task sets with midrange worst-case processor utilization values. The look-ahead RTDVS mechanism, in particular, seems able to follow the theoretical lower bound very closely. Although the total utilization greatly affects energy consumption, the number of tasks has very little effect. For simplicity 3 tasks only considered for simulations.

To see how an imperfect halt feature affects power consumption, we performed several simulations varying the idle level factors 0.01 and 1.0. Figure 2 shows the results for idle level factors 0.02, 0.2, and 1.0. The most significant result is that even with a perfect halt feature (i.e., idle level is 0), where the non-energy conserving schedulers are shown in the best light, there is still a very large percentage improvement with the RT-DVS algorithms. Obviously, as the idle level increases to 1 the percentage savings with voltage scaling improves.

Further we studied the effects of varying the simulated machine specifications[11-15]. The following summarizes the hardware voltage and frequency settings, where each tuple consists of the relative frequency and the corresponding processor voltage.

Machine 1: \{(0.5,3), (0.75,4), (1.0,5)\}  
Machine 2: \{(1.4, 0.36), (1.5,0.55),(1.6, 0.64), (1.7, 0.73), (1.8, 0.82), (1.9, 0.91),(2.0, 1.0)\}  
Machine 3: \{(2.5,0.375),(3.0,0.5),(3.5, 0.625), (4.0, 0.75),(4.5, 0.875),(5.0, 1.0)\}  

From figure 3 it is observed that the cc EDF and static EDF benefit from the large number of settings but la EDF benefit with fewer settings. It is indicated that the energy savings from the various RT-DVS algorithms depend greatly on the available voltage and frequency settings of the platform.
In this set of experiments, we vary the distribution of the actual computation required by the tasks during each invocation to see how well the RT-DVS mechanisms take advantage of task sets that do not consume their worst-case computation times.

Figure 4 shows simulation results for tasks that require a constant 85%, 75%, and 65% of their worst-case execution cycles for each invocation. We observe that the statically-scaled mechanisms are not affected, but both the cycle-conserving and look-ahead EDF schemes show great reductions in relative energy consumption as the actual computation performed decreases.

5 Conclusions
Here, we have presented several novel algorithms for real-time dynamic voltage scaling that, when can be coupled with the underlying OS task management mechanisms and real-time scheduler, can achieve significant energy savings, while simultaneously preserving timeliness guarantees made by real-time scheduling.

We also discussed in details about designing of DVS simulator, a unified simulation environment for evaluating dynamic voltage scaling algorithms. The simulator has been designed such a way that any new DVS algorithm can be evaluated easily by adding new task execution module to it. It also supports to add new machines specifications to in it.

At last we have presented extensive simulation results showing the most significant parameters affecting energy conservation through RT-DVS mechanisms and the extent to which CPU power dissipation can be reduced. Furthermore, look-ahead and cycle-conserving RTDVS
mechanisms can achieve close to the theoretical lower bound on energy. Our simulated results indicate that 20% to 30% energy savings can be achieved, even including irreducible system energy overheads and using task sets with high values for both worst-case and average-case utilizations.

References


