Minimizing Energy Consumption of Hard Real-Time Systems with Simultaneous Tasks Scheduling and Voltage Assignment Using Statistical Data*

Lap-Fai Leung, Chi-Ying Tsui, Wing-Hung Ki

Department of Electrical and Electronic Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong SAR, China

Abstract — In this paper, we tackle the problem of minimizing the energy consumption of single-processor-core systems in both offline and online phases. The tasks are reordered in the offline scheduling by taking into account the statistical information such that on average more slacks will be resulted during the tasks’ execution. The energy consumption is further optimized by optimally re-distributing the slack time and re-assigning the voltage for each task at runtime. Also, the correlations among the tasks are considered. Experimental results show that more than half of the energy reductions are achieved using the proposed approaches.

1. INTRODUCTION

Low energy consumption is an important design issue for digital systems. At the system level, efficient task assignment and scheduling is critical to achieve high performance and low energy consumption in single-processor-core systems. Many scheduling and voltage assignment techniques focus on either the static or the dynamic behavior [1,2,3] of the tasks but little research works address the relationship between the static and the dynamic phases of the task scheduling and voltage assignment. The orders of the tasks determined in the static scheduling phase impose great influence on the slack distribution [1] in the dynamic runtime phase. Another problem in the slack distribution during the on-line dynamic scheduling phase is the over-estimation of the tasks’ execution cycle. To meet the deadline requirement, existing dynamic voltage scheduling algorithms [1] will use a higher supply voltage than actually required and therefore unnecessary energy consumption is used. In many applications, tasks are correlated with each other and their execution times in each operation period are also correlated. With the help of the static correlation information of the tasks, we can predict the execution cycle at the current operation cycle of the tasks given that the actual execution times of the already-executed tasks are known.

Recently Gruian [1] presented a hard real-time scheduling with fixed priorities tasks assigned in a rate monotonic or deadline monotonic manner. He considered sets of independent tasks running on processors with dynamic voltage supplies. However, he did not cater for the real time behavior of the tasks early in the static scheduling. Zhang et al. [2] proposed a two-phase framework that integrates task allocation, ordering and voltage assignment together to minimize the energy consumption. Worst-case execution cycle is assumed for each task during the scheduling. Without considering the variation in the run-time dynamic workload, the optimality of the solution is not guaranteed because the slack time available during runtime is not fully utilized.

In this work, we propose an off-line and on-line tasks scheduling and voltage assignment algorithm to reduce the energy consumption of single-processor-core systems using multiple variable supply voltages [4]. In the off-line phase, given a Task Flow Graph (TFG) with dependency and the deadline requirements, task scheduling and voltage assignment are carried out for each task concurrently. Here we focus at non-preemptive systems with periodic dependent tasks. Worst case execution cycle is assumed to satisfy the hard deadline requirement. However, most of the time the actual execution workload of the tasks depends on the input data and will vary from period to period and very often it is not the worst case. Slack time will occur during run-time and this can be utilized to further reduce the energy consumption. The scheduled TFGs are reordered to give the best scheduling for the on-line voltage assignment to optimally utilize the slacks. The best scheduling means potential slacks are maximized and distributed to the maximum number of the successive tasks. With this, more tasks can run at a lower speed and more energy can be saved. The scheduling can be obtained by finding the task with the largest potential slack, which is estimated using the statistical data from the benchmark, and scheduling it first.

For a lot of applications, tasks with dependency between them are highly correlated and their workloads are also correlated. Based on this, we can predict the execution cycle of the tasks by using the actual execution cycles of its correlated tasks that are already executed. In the on-line phase, we assign the supply voltage for each task dynamically by using the estimated execution cycle. This estimation helps us to prevent from over-assigning the voltage for the tasks.

2. TASK REORDERING

During runtime, the actual number of the execution cycle of the tasks will be varied and different from the worst-case values. To reduce the energy consumption, we can utilize the slacks obtained from the executed tasks and distribute these slacks to the remaining un-executed tasks so that lower supply voltages can be used. However, the order of the task execution determined in the static phase is important since the amount of slack utilized is different with different tasks’ order. In this section, an approach called Task Reordering (TR) is proposed to reorder the tasks after the static scheduling such that the slack utilization can be maximized at runtime.

In general, tasks with higher mean of the slack should be scheduled earlier. In short, the larger the mean of the slack,
the earlier the task should be scheduled. The expected slack of task $T_i$ can be obtained from the worst case execution cycle $E_i$ and the mean execution cycle ($\bar{E}_i$), and it is equal to $E_{\text{delta-mean}_{-i}} = E_i - \bar{E}_i$. The mean execution cycle can be found from the probability distribution which is obtained from the offline benchmarking.

We use a greedy approach to re-order the tasks. We define a set $R$, which contains all the un-reordered tasks. Therefore initially it contains all the tasks. We also define another set $O$, which contains all the re-ordered tasks. At first, we select the task $T_k$ that has the largest $E_{\text{delta-mean}_{-k}}$ in $R$. Then we re-arrange $T_k$ to the earliest position in the order of the schedule such that the dependency and the deadline requirements of all the tasks are still satisfied. If it cannot be done, the schedule is unchanged. $T_k$ is then put in the finished set $O$. We repeat this reordering process until $R$ is empty. The re-ordered schedule is the final schedule that tasks with larger expected slacks are executed earlier.

3. Dynamic Voltage Assignment Using Execution Cycle Prediction

In this section, a method called Prediction Method (PM) for estimating the task execution cycle is presented. It can help to facilitate the online voltage assignment. During runtime, available slack is first distributed to the remaining tasks according to the mean execution cycles of the tasks. Supply voltage is then assigned to the next task based on the available execution time. To satisfy the deadline requirement, we have to use the worst-case execution cycle for the assignment of the voltage. However, we may use a higher-than-required supply voltage in many cases and hence waste the energy if the actual execution cycle deviates from the worst-case by a lot. Therefore, if we can have a good estimation of the actual execution cycle, we can increase the utilization of the processor and reduce the energy consumption by using a lower supply voltage. Also, an accurately-predicted execution cycle can be used instead of the mean value when the available slack is distributed. This will provide a more even distribution of the slack among the remaining tasks at runtime.

For non-preemptive systems with dependent tasks, tasks with dependency are normally highly correlated. Therefore we can obtain more accurate estimation on the number of execution cycles of a particular task $T_j$ given that the actual numbers of execution cycles of the tasks that $T_j$ depends on are known. To reduce the estimation complexity, we use the bivariate linear regression model for the execution cycle prediction by considering the correlation factor [5] between the tasks. To cater for the worst-case delay situation, it is better to use an estimated upper bound value instead of the mean value for the assignment of the voltage for $T_j$. Even the upper-bound estimate is used, it is still lower than the worst-case execution cycle and hence a lower voltage can be used. Using a high confidence level to estimate the upper bound will reduce the chance of the actual number of execution cycle falls outside the predictive value. Here we use a 99% confidence level to balance the accuracy and effectiveness of the prediction. Let the upper bound of the predicted execution cycle of the task $T_j$ with 99% confidence level be $\hat{E}_{j-up}$ and the actual execution cycle of the preceding dependent task $T_i$, which is already executed, be $E_i$. It can be shown by [4] that

$$E_{j-up} = k_1 + k_2 E_i + 3.29\sigma$$

where $k_1$, $k_2$ are the intercept and slope of the regression line. The $\sigma$ is the standard derivation of the execution cycle of $T_j$. Due to the page limitation, the derivation of this equation is omitted and the details can be found in [5]. All the parameters of the equation are calculated from the offline benchmark statistical data and can be updated during runtime.

When calculating the voltage for $T_j$, we assume the actual number of execution cycles equal to the predicted upper bound value obtained by (1). The voltage value also depends on the available execution time. Let $W_j$ and $D_j$ be the execution time reserved and the end time obtained in the static scheduling, respectively. The allowable working time is equal to the sum of $W_j$ and the slack ($S_j$) reserved for $T_j$, which is obtained by the mean proportional slack distribution [1] where $S_j = S \cdot E_j / \sum_{i=1}^{n} E_i$, and $S$ is the total slack available. We denote the voltage value obtained from this as $V_{PM, PMC}$. Figure 1a illustrates the calculation of this voltage.

To guarantee to satisfy the deadline requirement, we need to make sure the deadline will not be violated even when the actual number of execution cycle is really the worst-case which is larger than the estimated upper bound values. We calculate the lower bound of the supply voltage, $V_{PM, \text{min}, j}$ of $T_j$ such that no deadline will be missed even the execution cycle is at the worst-case. When calculating the $V_{PM, \text{min}, j}$ we use the worst case workload $E_{j, \text{worst}}$. The available execution time is the sum of $W_j$ and the maximum slack available ($S_j$) for $T_j$ without violating its deadline. This is illustrated in Figure 1b.

![Fig. 1. Voltage assignment for the (a) normal case (b) worst case.](image)

The final actual supply voltage $V_j$ assigned to task $T_j$ is maximum of $V_{PM, \text{min}, j}$ and $V_{PM, PMC, j}$. By doing so, the deadline requirement is guaranteed for the worst case and in most of the cases, a lower supply voltage, $V_{PM, PMC, j}$, can be used to reduce the energy consumption.

4. Experimental Results

A series of experiments, including both the artificial and real-life applications, were carried out to demonstrate the effectiveness of the two proposed methods. We first constructed 100 random task-sets, each has 50 tasks. Each task-set is simulated with 100 instances. Similar to the experimental settings in [1], we consider the number of execution cycle of each task varying between the best case (BC) and the worst case (WC) according to a normal
distribution. The distribution has a mean of $(BC+WC)/2$ and a standard deviation (SD) of $(WC-BC)/6$. The $BC/WC$ ratio is ranging from 0.1 to 0.9. The worst case execution cycles were chosen from a uniform distribution between 100 and 10000 and the deadline $D_j$ of a particular task $T_j$ was adjusted such that the processor utilization of the task-sets is about half when all the tasks are running at the maximum speed [1].

We compare the improvement of energy reduction to a base-line Dynamic Voltage Scaling scheme, which uses the static scheduling proposed in [2] and runtime DVS with slack distribution method proposed in [1]. We denote this as SDVS. Fig. 2 and Fig. 3 show the percentage improvement of the energy reduction of our proposed schemes, the reordering (TR) and the prediction (PM) approaches, over the SDVS approach, respectively. It can be seen that if the $BC/WC$ ratio is high, the total energy saving is small in all cases because the system is almost fully utilized and most of the tasks run at the maximum speed. This reduces the room for further reduction of the supply voltage. When the $BC/WC$ ratio is low, there are a lot of slacks available and our approach provides the best slack utilization and minimizes the overall energy consumption comparing with the SDVS approach.

We simulated for two different classes of task-set. In the first class of task-set, the tasks are closely dependent and correlated. On average, each task sends data to five other tasks. The results are summarized in Fig. 2. It can be shown when the $BC/WC$ ratio is high, the total energy saving is small because the system is close to fully utilized and most of the tasks run at the maximum speed. However, when the $BC/WC$ ratio is low, a lot of slacks are available and we can execute the tasks at a lower speed with a lower voltage. Since the tasks are highly dependent and the schedule is too tight that it does not have much flexibility to reorder the tasks without violating the dependency and deadline requirement. Therefore it can be seen that the improvement of TR over SDVS is small. On the other hand, since the tasks are correlated, we can accurately predict the upper bound of the number of execution cycles. Therefore, we can see that there is significant energy reduction by using the PM policy.

In the second class of task-set, the tasks are loosely dependent. Only half of the tasks have dependency with others. The results are summarized in Fig. 3. In this case, we have high flexibility to reorder the static schedule and therefore we can achieve a higher energy reduction using the TR method. However, fewer tasks are correlated and so the performance of PM is not as good as that in the first set of experiment. From Figures 2 and 3, we can see that the combined policy, “TR & PM”, outperforms all the others in all testing cases.

We also demonstrated the effectiveness of our proposed method using a real-life example, the MPEG4 video encoding system. QCIF Format with 144*176 pixels was used in the experiments. We divided the frame into macroblocks with size of 16*16 pixels, and dynamically obtained the statistical correlation data among different computation blocks of the encoder by using the data from the previous 100 frames. We carried out simulation for two different video sequences. The first one is “reporter” which has a steady motion. The tasks in different frames are highly correlated. The second sequence is ‘tennis’ which has a fast moving motion. In this sequence, the tasks in different frames are less correlated. Table 1 summarizes the results. It can be seen that a significant reduction in energy consumption can be achieved using the Predictive Method over the conventional SDVS method.

![Fig. 2. Improvement in energy reduction for tasks with high dependency](image)

![Fig. 3. Improvement in energy reduction for tasks with low dependency](image)

<table>
<thead>
<tr>
<th>Movie sequence</th>
<th>Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reporter</td>
<td>46</td>
</tr>
<tr>
<td>Tennis</td>
<td>38</td>
</tr>
</tbody>
</table>

5. CONCLUSIONS

We presented off-line task re-ordering and on-line voltage assignment methods for real-time non-preemptive system. The off-line method re-orders the tasks such that a better slack distribution can be obtained at runtime. The on-line method predicts the worst-case execution cycle of the task at runtime by using the correlation with some already executed task. With an accurate prediction, a lower supply voltage can be used and the energy consumption can be reduced.

REFERENCES