VARIABLE VOLTAGE TASK SCHEDULING FOR MINIMIZING ENERGY OR MINIMIZING POWER

A. Manzak and C. Chakrabarti

Department of Electrical Engineering
Telecommunications Research Center
Arizona State University
Tempe, AZ 85287-5706
istanbul@asu.edu, chaitali@asu.edu

ABSTRACT

In this paper we propose task scheduling algorithms that minimize energy or minimize power for the case when the tasks have different arrival times, deadline times, execution times and switching activities. We theoretically determine the relation between the operating voltages for the minimum energy (power) assignment and develop a polynomial time scheduling algorithm that uses this relation. We show experimentally that the voltage assignment obtained by our algorithm is very close to that of the optimal assignment.

1. INTRODUCTION

Energy minimization is critically important for portable devices running on batteries. Substantial energy minimization can be achieved without sacrificing performance by lowering the supply voltage (and thus slowing the clock) instead of idling when the computational load of a task is low [1]. Thus determining the supply voltage of the processor as it executes tasks with different computational loads such that the energy consumption is minimum (without sacrificing performance) is indeed an important problem.

In this paper we consider the problem of developing a static schedule for a set of tasks with different arrival times, deadline times and execution times that have to be scheduled in a fixed time T. Our aim is to determine the operating voltage of the processor as it executes each task such that the power consumption or the energy consumption is minimum. We assume that the processor executes an entire task at the same voltage. We have used the Lagrange multiplier method to show that (i) the minimum energy configuration corresponds to the case when $\alpha V_{dd}(V_{dd}-V_t)^2$ is the same for all the tasks and (ii) the minimum power configuration corresponds to the case when $\frac{\alpha}{s}(V_{dd}-V_t)^4$ is the same for all the tasks, where V_{dd} is the operating voltage, α is the switching activity and s is the execution time in terms of control cycles.

We have developed a polynomial time scheduling algorithm that utilizes this relation for determining the minimum energy (power) configuration. The algorithm determines temporary schedules for increasingly smaller voltage assignments ($V-\Delta V, k=0,1,2,...$) and finalizes the schedule for a particular task to operate at $V-p\Delta V$ only when a feasible schedule could not be obtained at $V-(p+1)\Delta V$. Experiments show that the voltage assignment obtained by this algorithm is very close to the optimal energy (power) assignment.

There are several off-line algorithms for the variable voltage task scheduling problem [2]-[5]. The scheduling problem for the case when there are no dependencies between the tasks is addressed in [2] and solved using integer linear programming. A two phase scheduling algorithm is presented in [3] where in the first phase, all tasks are scheduled at the nominal voltage and in the second phase, the voltages of each task are adjusted till no further reduction in power is obtained. The scheduling algorithm in [4] schedules the critical jobs at the highest speed and then constructs a subproblem for the remaining jobs and solves it recursively. The differences in the switching activities of the tasks are not taken into consideration in both [3], [4]. On-line algorithms for the variable voltage task scheduling problem have a different flavor [5], [4]. These algorithms are based on using past history to predict the work load.

1.1. Problem Definition

Given a set of n tasks ($Task_1, Task_2, ..., Task_n$) and a computation time T_{total}, our aim is to schedule the tasks in time T_{total} such that the power or energy consumption is minimum. Associated with task j are the following parameters: (i) a_j: the arrival time of task j, (ii) d_j: the deadline time of task j, (iii) e_j: the execution time of task j with the maximum voltage V_{ref} and (iv) α_j: the average switching activity of task j. α_j is related to the type of program being executed and can be obtained by simulation.

Scheduling the tasks is equivalent to determining the supply voltage of the processor during execution of each of the tasks. In our model, the supply voltage remains constant during the execution of the entire task.

1.2. Formalization

Let the operating clock frequency of the system $f=1/\tau_c$, where τ_c is the critical path delay. So, the delay of each task (in ns) can be written as

$$\tau_j = s_j \tau_c, \tag{1}$$

where s_j is the number of control cycles required to execute task j. Recall that the critical path delay is strongly related to the supply voltage by

$$\tau_c(V_{dd}) = k' C_L \frac{V_{dd}}{(V_{dd} - V_t)^2} \qquad (2)$$

where $\tau_c(V_{dd})$ is delay of the circuit for supply voltage V_{dd}, V_t is the threshold voltage, C_L is the load capacitance and k' is a device parameter which depends on the transconductance and the width to length ratio. Thus equation (1) can be rewritten in the following form.

$$\tau_j = s_j k' C_L \frac{V_{dd_j}}{(V_{dd_j} - V_t)^2} \qquad (3)$$

Since the computations need to be completed in time T_{total},

$$T_{total} = \sum_{j=1}^{n} \tau_j$$

The energy of task j is given by $\alpha_j C_L V_{dd_j}^2 s_j$, and the total energy consumption is

$$E_{total} = \sum_{j=1}^{n} \alpha_j C_L V_{ddj}^2 s_j$$

The power consumption of task j is given by $\alpha_j C_L V_{dd_j}^2 f_j$, and the average power consumption is

$$P_{ave} = \frac{1}{n} \sum_{j=1}^{n} \alpha_j C_L V_{ddj}^2 f_j$$

Our aim is to minimize E_{total} or P_{ave} subject to the time constraint T_{total}. We use the Lagrange multiplier method to determine the supply voltage of each task. It is found that E_{total} is minimum when

$$\alpha_1 V_{dd_1}(V_{dd_1} - V_t)^2 = ... = \alpha_n V_{dd_n}(V_{dd_n} - V_t)^2$$

and P_{ave} is minimum when

$$\frac{\alpha_1}{s_1}(V_{dd_1} - V_t)^4 = ... = \frac{\alpha_n}{s_n}(V_{dd_n} - V_t)^4$$

Since computing the supply voltage such that $V_{dd_j}(V_{dd_j} - V_t)^2$ is the same for all the tasks can be a computational burden, we simplified this term in our actual implementation. This simplification results in a very mild error (see fig 1) and can be ignored for all practical purposes. We can approximate $V_{dd_j}(V_{dd_j} - V_t)^2$ to $(V_{dd_j} - V_t)^3$ for $V_{dd_j} > 2V_t$. Thus E_{total} is minimum when

$$\alpha_1(V_{dd_1} - V_t)^3 = ... = \alpha_n(V_{dd_n} - V_t)^3 \qquad (4)$$

and P_{ave} is minimum when

$$\frac{\alpha_1}{s_1}(V_{dd_1} - V_t)^4 = ... = \frac{\alpha_n}{s_n}(V_{dd_n} - V_t)^4 \qquad (5)$$

The results are graphically illustrated in Figure 1 for the case when there are three independent tasks (i.e. there is no dependency between the tasks) that have to be computed in time T_{total}. In Figure 1 $s_1 = 10$, $s_2 = 1$, $s_3 = 1$, $\alpha_1 = 1$, $\alpha_3 = 1$, and α_2 is varied from 1 to 11. E_{best} (P_{best}) is the lowest energy (power) when switching activities are considered, $E_{algorithm}$ ($P_{algorithm}$) is the energy (power) using our formulation in eq. 4 (eq. 5), $E_{no-\alpha}$ ($P_{no-\alpha}$) is the lowest energy (power) when the switching activities are ignored and E_{power} (P_{energy}) is the energy (power) when power (energy) is minimized using eq. 5 (eq. 4). Note that $E_{algorithm}$ ($P_{algorithm}$) is very close to E_{best} (P_{best}) for all switching activities, both $E_{no-\alpha}$ ($P_{no-\alpha}$) and E_{power} (P_{energy}) deviate significantly from E_{best} (P_{best}) for large switching activities. This reiterates our observation that (i) switching activity cannot be ignored in task scheduling for both minimum energy and minimum average power and (ii) the voltage assignment for minimum energy and minimum average power is different.

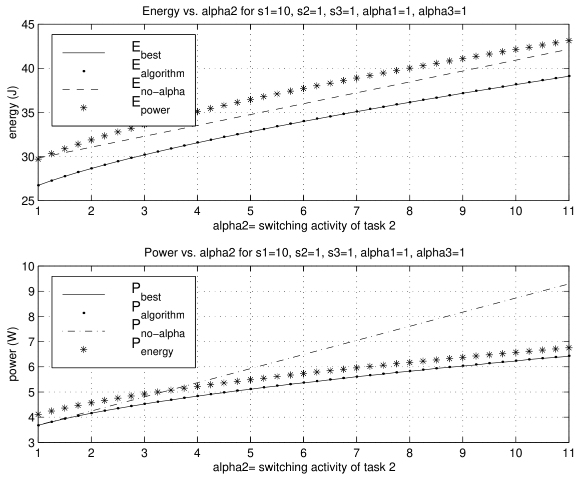

Figure 1: Energy vs. α_2 and power vs. α_2 plots for a three task assignment problem

2. ALGORITHM

2.1. Definitions

Given the total computation time T_{total}, the arrival time a_j, the deadline time d_j, and the execution time e_j of each task, we define the following additional parameters to help describe the algorithm.
L_j : the lifetime of task j. $L_j = d_j - a_j$.
$N_j(tr)$: number of tasks assigned to the same time region tr, excluding task j.
C_j : task constant of task j. $C_j = \frac{1}{e_j}\sum_{a_j}^{d_j} N_j(tr)$.
ΔV : DC-DC converter sensitivity.
s_j : number of cycles required to execute task j.
τ_j : allowed computation time of task j. This changes from iteration to iteration.
$O_j(k)$: number of times V_{dd_j} is decremented by $k\Delta V$.

2.2. Overview

The algorithm for minimum energy task scheduling and minimum average task scheduling is very similar. This is to be expected since while $\alpha_j^{1/3}(V_{dd_j} - V_t)$ has to be equal for the minimum energy assignment, $\frac{\alpha_j}{s_j}^{1/4}(V_{dd_j} - V_t)$ has to be equal for the minimum average power assignment. In this section we describe the minimum energy task scheduling algorithm. The algorithm for minimum average power is very similar; the only difference is that $\alpha_j^{1/3}$ has to be scaled to $(\frac{\alpha_j}{s_j})^{1/4}$.

The proposed algorithm operates by determining temporary schedules for increasingly smaller voltage assignments ($V - k\Delta V, k = 0, 1, 2, ...$) and finalising the schedule for a task to operate at $V - p\Delta V$ only when a feasible schedule cannot be obtained at $V - (p+1)\Delta V$. Thus temporary schedules are determined (for increasingly smaller voltage assignments) as long as schedules are feasible and there are unassigned time slots left. If, as a result of lowering the voltage, a feasible schedule is no longer found, the algorithm goes back to its previous state (i.e. increases the voltage) and finalises the schedule of the task with the highest priority.

To aid in this procedure the algorithm maintains two lists: a temporary list and a main list. Initially all the tasks are in both lists. The algorithm operates on the tasks in the temporary list. After the final schedule of a task is determined (this may take several iterations), it is removed from the main list.

In order to compute the temporary schedule efficiently, each task is assigned a scheduling priority, which is a function of the parameter task constant $C_j = \frac{1}{e_j}\sum_{a_j}^{d_j} N_j(tr)$. The lower the value of C_j, the higher is its priority. The parameter C_j is a function of the execution time of the task and the number of other tasks with overlapping lifetimes. Thus if the task's execution time is close to it's lifetime, ie. low freedom, and the lifetime of the task does not overlap with a lot of other tasks, then that task has higher scheduling priority.

The task with the highest scheduling priority or the lowest task constant is selected. For this task, the parameter $V_{dd_j} = V_{ref} - \Delta V_j$ has to be calculated where V_{dd_j} is the operating voltage and ΔV_j is the voltage decrement for task j. It can be shown that in each iteration, the change in supply voltage will be maximum for the task with the minimum switching activity. Let task s be the task with the minimum switching activity ($\alpha_s = \alpha_{min}$). Then the voltage of task s is decreased by $k\Delta V$, where k is a parameter that changes as the algorithm proceeds. The voltage of the other tasks is decreased proportionally. For task j, the following relation holds.

$$\alpha_j(V_{ref} - \Delta V_j - V_t)^3 = \alpha_{min}(V_{ref} - k\Delta V - V_t)^3$$

Solving for ΔV_j, we have

$$\Delta V_j = \tilde{V}_{ref}(1 - (\frac{\alpha_{min}}{\alpha_j})^{1/3}) + k\Delta V(\frac{\alpha_{min}}{\alpha_j})^{1/3}$$

where $\tilde{V}_{ref} = V_{ref} - V_t$. In order to make ΔV_j a multiple of ΔV, it is normalized to $\Delta V_j = \lfloor \frac{\Delta V_j}{\Delta V} \rfloor \Delta V$. Next, the value of τ_j is calculated from $\tilde{V}_{ref} - \Delta V_j$ using eqn. (3) for each j. τ_j is normalized to $\tau_j = \lceil \frac{\tau_j}{tr} \rceil tr$. Finally the starting value of k is given by $k = \lfloor \frac{\tilde{V}_{ref}}{\Delta V}(1 - (\frac{\alpha_{max}}{\alpha_{min}})^{1/3}) \rfloor$ in order to set all $\Delta V_j = 0$ at the beginning of the algorithm.

The selected task or task j is scheduled in either the time region $[a_j, a_j + \tau_j]$ or $[d_j - \tau_j, d_j]$. The time region with minimum number of overlapping tasks and minimum number of unused time gaps between tasks is chosen. This method ensures minimizing the unused time gaps between the tasks and allowing more time slots for unscheduled tasks. If there are unassigned time regions left, after scheduling all the tasks with a specific value of k, then the delay of each task is increased by incrementing the parameter k. As a result, there maybe a conflict and a task may be assigned outside of its region of support. In such a case, we go back to the last feasible solution (decrement k) and choose the task with the lowest task constant. If that task had its voltage incremented by $k\Delta V$ more than once, then the schedule listed for that task is final. The task is removed from the temporary list and the main list. A temporary schedule is then determined for the remaining tasks in the temporary list for the new value of k and α_{min}.

2.3. Algorithm

1. (a) Create a time region $TR\{a_i, d_j\}$ where i corresponds to the task with the earliest arriving time and j corresponds to the task with the latest deadline time. Create time regions tr in TR.
   (b) $k = k_{start}$;
   (c) $O_j(k) = 0 \qquad \forall j, k$
2. Repeat LOOPS times
   (a) For each tr $\in$ TR, compute $N_j(tr)$.
   (b) For each j, compute C_j.
   (c) Select the task j with minimum C_j.
   (d) For the selected task j, compute $\Delta V_j = \tilde{V}_{ref}(1 - (\frac{\alpha_{min}}{\alpha_j})^{1/3}) + k\Delta V(\frac{\alpha_{min}}{\alpha_j})^{1/3}$.
   If $\Delta V_j > 0$
      $V_{dd_j} = V_{ref} - \lfloor \frac{\Delta V_{dd_j}}{\Delta V} \rfloor \Delta V$
   else
      $V_{dd_j} = V_{ref}$
   (e) Compute τ_j from V_{dd_j} (eq. 3) for task j. Make τ_j a multiple of tr by $\tau_j = \lfloor \frac{\tau_j}{tr} \rfloor tr$. Readjust V_{dd_j} from τ_j.
   (f) if $\{\sum_{a_j}^{a_j+\tau_j} N_j < \sum_{d_j-\tau_j}^{d_j} N_j\}$
      $start_j = a_j$, $end_j = a_j + \tau_j$
   else if $\sum_{a_j}^{a_j+\tau_j} N_j > \sum_{d_j-\tau_j}^{d_j} N_j\}$
      $start_j = d_j - \tau_j$, $end_j = d_j$
   else if there is a previously scheduled task p, such that $a_p = d_j$ and $a_p - a_j \geq \tau_j$
      $start_j = d_j - \tau_j$, $end_j = d_j$
   else
      $start_j = a_j$, $end_j = a_j + \tau_j$
   end if
   Schedule task j from $start_j$ to end_j. $O_j(k) = O_j(k) + 1$.
   Delete task j in the temp list.
   Update a_j, d_j for the remaining tasks in the temp task list.
   (g) If $O_j(k) > 1$, (task j is scheduled for k more than once)
      then the schedule assigned to task j is final.
      Delete task j from the main task list.
      Update $\alpha_s = \alpha_{min}$ and $k = k + 1$.
      Go to step 2.
   endif

		A	B	C	D	E	F
$T_{total} = 20$	e_j	1	2	2	1	2	3
	α_j	1	8	1	8	1	8
case 1	a_j	6	3	0	3	14	5
	d_j	12	8	4	9	20	17
case 2	a_j	0	1	4	6	8	8
	d_j	6	6	8	12	14	20
case 3	a_j	6	0	16	5	8	4
	d_j	12	5	20	11	14	16
case 4	a_j	5	4	8	0	14	6
	d_j	11	9	12	6	20	18
case 5	a_j	10	11	2	0	6	8
	d_j	16	16	6	6	12	20
$T_{total} = 30$	e_j	1	2	3	4	1	4
	α_j	1	1	1	8	1	8
case 6	a_j	0	1	3	7	16	14
	d_j	4	7	15	20	25	30
case 7	a_j	9	7	11	0	3	14
	d_j	13	13	23	13	12	30
case 8	a_j	0	24	4	11	0	14
	d_j	4	30	16	24	9	30
case 9	a_j	19	13	0	17	12	0
	d_j	23	19	12	30	21	16
case 10	a_j	9	11	0	3	21	11
	d_j	13	17	12	16	30	27

Table 1: Summary of user defined task configurations.

(h) If task j is scheduled outside of its region
    $k = k - 1$, update temp list with the tasks in the main list. Go to step 2.
    (Go back to the last feasible solution)
  else if unassigned task left in the temporary list.
    go to step 2
  else if unassigned time region left
    if unassigned task left in the main list.
      $k = k + 1$,
      update temp list with the tasks in the main list. Go to step 2.
    else
      rearrange the scheduling boundary of the tasks in unassigned time slots.
  /*else*/
    assignment end

3. RESULTS

In this section we present the results obtained by running our algorithm on user specified task configurations. 10 different cases are considered. In each case there are 6 tasks A, B, C, D, E, F. e_j, α_j, a_j and d_j is given for each task. Cases $1-5$ have the same T_{total}, e_j and α_j. Similarly cases $6-10$ have the same T_{total}, e_j and α_j. The different configurations are given in Table I.

The assignment results are presented in Table II. Energy values are calculated for each case. E_{best} corresponds to the lowest energy if α values are considered, $E_{algorithm}$ is the energy obtained if our algorithm is used for scheduling,

case	E_{best}	E_{alg}	% error of E_{alg}	$E_{no-\alpha}$	% error of $E_{no-\alpha}$
1	209.4	209.4	0	239.4	14.3
2	202	202	0	228.3	13
3	207.3	207.3	0	238.4	15
4	206.2	206.2	0	232.6	12.8
5	222.6	227.8	2.3	239.4	7.5
6	253.6	253.6	0	284	12
7	257.8	274.5	6.4	282.9	9.7
8	279.1	279.4	0.1	295.1	5.7
9	254.6	255.7	0.4	284	11.5
10	262.8	263.9	0.4	283	7.6
total	2355.4	2379.8	1.0	2607.1	10.6

Table 2: Comparison of the results with the optimal solution.

$E_{no-alpha}$ is the lowest energy if α values are not considered. % error is $(E - E_{best})/E_{best}$.

An analysis of the results show that the energy savings obtained by our algorithm is very close to the optimal. The error ranges from 0% to 6.4%. The error values are much larger (upto 15%), if switching activity is ignored. Thus if tasks with different switching activities are to be scheduled, ignoring the switching activities will certainly not give the minimum energy schedule.

4. COMPUTATIONAL COMPLEXITY

The worst case running time of our algorithm is $O(nVL)$ where n is the number of tasks, V is the number of possible voltage assignments, L is the number of time slots in T_{total}, $L = T_{total}/tr$. Thus higher the converter sensitivity, the closer is the solution to the optimal energy assignment, and higher is the running time of the algorithm.

5. REFERENCES

[1] V. Gutnik and A. P. Chandrakasan, "Embedded power supply for low-power DSP" IEEE Trans. on VLSI systems. Vol. 5, no 4, 1997.

[2] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processor" Proc. of the Int. Symp. on Low Power Design, 1998, New York NY.

[3] I. Hong, D. Kirovski G. Qu, M. Potkonjak and M-B. Srivastava, "Power Optimization of Variable Voltage Core-Based Systems" DAC 1998, p 176-81.

[4] F. Yao, A. Demers and S. Shenker, "A scheduling model for reduced CPU energy" IEEE Annual Foundations of Computer Science. pp 374-82, 1995.

[5] T. Pering, T. Burd and R. Brodersen, "The simulation and evaluation of dynamic voltage scheduling algorithms" Int. Symp. on Low Power Electronics and Design, pp. 76-81, 1998.