Transaction Briefs

Variable Voltage Task Scheduling Algorithms for Minimizing Energy/Power

Ali Manzak and Chaitali Chakrabarti

***Abstract*—In this paper, we propose variable voltage task scheduling algorithms that minimize energy or minimize peak power for the case when the task arrival times, deadline times, execution times, periods, and switching activities are given. We consider aperiodic (earliest due date, earliest deadline first), as well as periodic (rate monotonic, earliest deadline first) scheduling algorithms. We use the Lagrange multiplier method to theoretically determine the relation between the task voltages such that the energy or peak power is minimum, and then develop an iterative algorithm that satisfies the relation. The asymptotic complexity of the existing scheduling algorithms change very mildly with the application of the proposed algorithms. We show experimentally (random experiments as well as real-life cases), that the voltage assignment obtained by the proposed low-complexity algorithm is very close to that of the optimal energy (0.1% error) and optimal peak power (1% error) assignment.**

***Index Terms*—Dynamic voltage scaling, energy minimization, Lagrange multiplier method, low power, off line algorithms, on line, peak power minimization, task scheduling.**

I. Introduction

Energy minimization and power minimization are important performance metrics in todays world. While energy minimization is critically important for portable devices running on batteries, peak power consumption minimization is important in high-power systems where the high temperature causes silicon failure and increases the system cost (cooling, packaging) [1]. Substantial energy/power 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 [2].

In this paper, we consider a system which consists of a processor that is capable of operating over a range of supply voltages and a dc–dc converter that is capable of supplying these voltages. Voltage scheduling on such systems has been studied extensively in the past couple of years [3]–[17]. The algorithms can be broadly classified into off line algorithms [3]–[7] and on line algorithms [3], [7]–[17]. The off line algorithms have low complexity and yet achieve significant energy savings since complete information about the tasks are known *a priori*. If complete information about the tasks are not known or if the variation in the workload is significant, adjusting the voltages dynamically (or on line) results in even greater energy savings.

In this paper, we consider the problem of task scheduling on a processor capable of dynamic voltage scaling (DVS) that minimizes energy or peak power consumption. We consider different kinds of scheduling—periodic, aperiodic, on line, off line, etc. The input to our system consists of task arrival times, deadline times, execution times, switching activities, periods, and/or the task set completion time. Our aim is to determine the operating voltage of the processor as it executes each task such that the energy or peak power consumption is minimum.

Our approach to solving the task scheduling problem is to first develop a theoretical relationship between the task voltages for minimizing energy/power consumption, and then to develop a simple heuristic to satisfy the relation. By using the Lagrange multiplier method, we have shown that the minimum-energy configuration corresponds to the case when $\alpha V(V-V_t)^3/(V+V_t)$ is the same for all the tasks, and that the minimum peak power configuration corresponds to the case when $\alpha V(V-V_t)^2$ is the same for all the tasks, where V is the operating voltage, V_t is the threshold voltage and α is the switching activity. We approximate the minimum energy and the minimum peak power formulation and use the condition $\alpha(V-V_t)^3$ as the same for all tasks. Our procedure has been experimented on 10 000 randomly generated task configurations as well as some real-life cases. For the randomly generated cases, the energy savings are between 43.5% and 45.7% when $T_{\rm total} = 1.5T_{\rm crit}$ for aperiodic schedules.

The main features of our scheme are as follows.

1) It can be used widely for different classes of task scheduling problems (preemptive, nonpreemptive, dynamic, static, periodic, aperiodic, sporadic, etc.).

2) It finds the optimal voltage assignment for minimum energy and minimum peak power when all task information is available in advance.

3) It finds a near-optimal solution with the low complexity algorithm for random and real-life examples with 0.1% error for minimum energy and 1% error for minimum peak power.

4) It has a low complexity; the asymptotic complexity of the scheduling algorithms increases mildly when the iterative voltage adjustment algorithm is applied.

The rest of the paper is organized as follows. Section II describes existing work in dynamic voltage scaling. Section III formalizes the problem and presents the iterative algorithm. Section IV presents the algorithms EDD and EDF for aperiodic task assignment with the examples. Section V presents the algorithms RM and EDF for periodic task assignment with the examples. Section VI includes the results on user generated as well as real life examples. Section VII concludes the paper.

II. Related Work

Voltage scheduling on dynamic voltage processor has been studied extensively in recent years [2]–[17]. When complete information about the tasks is available in advance, voltage scheduling for energy minimization can be done off line [3]–[7]. If information about future tasks is not available or the variations in the workload are significant, voltage scheduling should be done dynamically, i.e. at run time [3], [7]–[17]. The better the estimates of the future task parameters greater are the energy savings.

Off line scheduling for the case when there are no dependencies between the tasks is addressed in [4] and solved using integer linear programming (ILP). An optimal off line algorithm with complexity $O(n\log^2 n)$ for n preemptive tasks is proposed in [3]. The algorithm

Manuscript received November 1, 2000; revised April 5, 2001 and April 23, 2002. This work was supported in part by the Center for Low-Power Electronics, National Science Foundation State, Industry, and University Cooperative Research Center.

A. Manzak was with Lattice Semiconductor Corporation, San Jose, CA 95131-1724 USA. He is now with Department of Electronics and Communication Engineering, Suleyman Demirel University, Isparta 32260 Turkey (e-mail: manzak@mmf.sdu.edu.tr).

C. Chakrabarti is with the Department of Electrical Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: chaitali@asu.edu).

Digital Object Identifier 10.1109/TVLSI.2003.810801

schedules the critical jobs at the highest speed, then constructs a subproblem for the remaining jobs and solves it recursively. An off line heuristic for non-preemptive task scheduling is presented in [5]. More recently, [6] provides a $O(n^3)$ algorithm for voltage scheduling when the timing parameters of the tasks are known *a priori*. The algorithm builds a voltage schedule by identifying critical intervals and finding their corresponding speeds.

On line algorithms track the variations in the workload and result in greater energy savings. The average rate heuristic (AVR) is a popular on line algorithm [3] where at anytime t, the AVR sets the speed of the processor to the sum of the average rate requirements of tasks whose time frame includes t. An extension of AVR, which tries to flatten the task voltage profile has been proposed in [8]. The incremental on line algorithm has a worst case complexity of $O(n^3)$, where n is the number of tasks.

In order to track the variation in the workload, [10] implements two interval based scheduling methods, PAST and AVG (that were originally proposed in [9]) on a PDA. While PAST assumes that the workloads of the future intervals will be like the previous one, AVG takes the exponential moving average of the workload of the previous intervals to determine the speed of the processor. The effectiveness of these algorithms has recently been evaluated on a pocket computer in [11]. An extension of the approach in [10] makes use of information such as recent processor utilization, predicted future behavior, estimates of workload provided by the individual tasks, etc. to determine the desired operating speed [12]. The energy savings are dependent on the workload and the hardness of the deadline constraints, as expected.

The variation in the execution time of the tasks has also been exploited in [13] which allows the processor (at run time) to lower the supply voltage such that the current active job finishes at its deadline or the release time of the next job. If the release times and deadlines are known *a priori*, then greater energy savings can be obtained by dynamically varying the speed of the processor to exploit the execution time variation of each task as in [14].

In order to fully exploit the slack time caused by workload variation, the supply voltage has to be varied during the execution of a task (instead of only at the task boundaries). One such intra-task voltage scheduling method is proposed in [15] where a task is partitioned into several pieces called time slots, and the voltage is dynamically varied timeslot-by-timeslot basis.

A more aggressive intra-task voltage scheduling method based on static program analysis is proposed in [16] and [17]. The algorithm exploits the fact that there are large execution path variations among different execution paths and so if short execution paths can be identified, then the corresponding clock speed can be lowered.

III. Formalization

A. Problem Definition

Given a set of n tasks $(\gamma_1, \gamma_2, \ldots, \gamma_n)$ and a computation time T_{total}, our aim is to schedule the tasks in time T_{total} such that 1) the energy consumption is minimum and 2) the peak power consumption is minimum. Associated with task j are the following parameters:

- a_j: arrival time of task j,
- d_j: deadline time of task j,
- p_j: period of task j,
- α_j: 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.

B. Aperiodic (Single) Tasks

Each task appears once in given time T_{total}. If τ_j is the execution time of task j, and n is the number of tasks, then

$$T_{\text{total}} \geq \sum_{j=1}^{n} \tau_j = \sum_{j=1}^{n} s_j k' C' \frac{V_j}{(V_j - V_t)^2} \tag{1}$$

where s_j is the number of control cycles required to execute task j, V_j is the circuit supply voltage, V_t is the threshold voltage, C' is the load capacitance of the critical path, and k' is a device parameter which depends on the transconductance and the width to length ratio.

1) Formulation for Energy Minimization: The energy of task j is given by $\alpha_j C_L V_j^2 s_j$, where C_L is the total load capacitance and the terms V_j, α_j, s_j are the same as defined before. The total energy consumption is thus

$$E_{\text{total}} = \sum_{j=1}^{n} \alpha_j C_L V_j^2 s_j.$$

Our aim is to minimize E_{total} subject to the constraint $T_{\text{total}} = \text{constant}$. By using the Lagrange multiplier method, we find that E_{total} is minimum when the following condition is satisfied

$$\frac{\alpha_1 V_1 (V_1 - V_t)^3}{V_1 + V_t} = \cdots = \frac{\alpha_n V_n (V_n - V_t)^3}{V_n + V_t}. \tag{2}$$

For $V_j > 3V_t$, this is approximated to

$$\alpha_1 (V_1 - V_t)^3 = \cdots = \alpha_n (V_n - V_t)^3. \tag{3}$$

The error introduced as a result of this approximation is at most ≈0.1% [7]. The low complexity energy minimization algorithm adjusts the voltages of the tasks according to (3) provided that the timing constraints are not violated. The optimal assignment algorithm, on the other hand, adjusts the voltages according to (2).

2) Peak Power Minimization: The power consumption of task j is given by $P_j = \alpha_j C_L V_j^2 f_j$ where f_j is the clock frequency of task j. The average power consumption is, thus,

$$P_{\text{ave}} = \frac{1}{T_{\text{total}}} \sum_{j=1}^{n} P_j \tau_j.$$

Clearly, P_{ave} is minimized when energy is minimized. Let $P_{\max}$ be the maximum power consumption of the set of tasks. We refer to $P_{\max}$ as the *peak power* consumption. Then our aim is to determine the task voltages such that $P_{\max}$ is minimum. Since clock frequency of each task is determined by $f_j = (V_j - V_t)^2 / k' C' V_j$

$$P_j = \frac{\alpha_j C_L V_j (V_j - V_t)^2}{k' C'}, \qquad 1 \leq j \leq n.$$

Clearly, $P_j \leq P_{\max}$ and $P_{\max}$ is minimum when $P_1 = P_2 = \cdots = P_n$. This implies that

$$\alpha_1 V_1 (V_1 - V_t)^2 = \cdots = \alpha_n V_n (V_n - V_t)^2. \tag{4}$$

Note that for $V_j > 3V_t$, (4) can be approximated to

$$\alpha_1 (V_1 - V_t)^3 = \cdots = \alpha_n (V_n - V_t)^3 \tag{5}$$

which is the same as the condition for minimum energy [see (3)]. The approximation can introduce an error as high as ≈8.6% for voltage range of 1 V to 3.3 V, $V_t = 0.4$ V and $\alpha_{\max}/\alpha_{\min} = 2$. This error is clearly higher than the error introduced in the energy formulation. This is because a task with power consumption much higher than others can skew the solution. The low-complexity peak power minimization

algorithm adjusts the voltages of the tasks according to (5) provided that the timing constraints are not violated. The corresponding optimal assignment algorithm uses (4).

C. Periodic Task Assignment

The assignment of periodic tasks can be calculated in the same way as the aperiodic assignment. The only difference is that in time T_{total}, there are now multiple instances of a task. If p_j is the period of task j, then τ_j/p_j is essentially the fraction of processor time spent in execution of task j. So, the utilization factor U is

$$U = \sum_{j=1}^{n} \frac{\tau_j}{p_j} \leq 1. \tag{6}$$

Since T_{total}/p_j is the number of instances of task j in time T_{total}, the total energy consumption in time T_{total} is

$$E_{\text{total}} = \sum_{j=1}^{n} \frac{T_{\text{total}}}{p_j} \alpha_j C_L V_j^2 s_j.$$

Our aim is to minimize E_{total} subject to the constraint $U = \text{constant}$ by using the Lagrange multiplier method, we find that the energy and peak power minimization formulation are the same as in aperiodic tasks.

D. Slack Adjustment

1) Slack Adjustment for Low-Complexity Algorithm: Let V_k be the voltage of task k. Then, if the Lagrange relation (3) is to be satisfied, the voltage of task j, V_j is related to V_k by

$$V_j = \left(\frac{\alpha_k}{\alpha_j}\right)^{1/3} (V_k - V_t) + V_t.$$

Now if the voltage of task k is decreased by ΔV, where ΔV is the converter sensitivity, the voltage of the remaining tasks have to be adjusted. The new voltage of task j is

$$V_{j_{\text{new}}} = \left(\frac{\alpha_k}{\alpha_j}\right)^{1/3} (V_k - \Delta V - V_t) + V_t.$$

The change in voltage of task j, $\Delta V_j = V_j - V_{j_{\text{new}}}$ is a function of ΔV

$$\Delta V_j = \left(\frac{\alpha_k}{\alpha_j}\right)^{1/3} \Delta V. \tag{7}$$

From (7), we see that the maximum voltage change occurs for the task with the minimum α value. Let that task be referred to as task m. Then the task voltages are changed with respect to task m, since this ensures that the voltage change for the remaining tasks will not be more than ΔV. Thus

$$V_j = \left(\frac{\alpha_{\min}}{\alpha_j}\right)^{1/3} (V_m - V_t) + V_t. \tag{8}$$

Starting voltage of task m: The task voltages are iteratively adjusted so that the Lagrange relation is satisfied. In the first iteration, task m is set to V_{start}. V_{start} is higher than $V_{\max}$ to ensure that in the beginning no task is assigned a voltage lower than $V_{\max}$. V_{start} is calculated assuming that the task with the highest α value is assigned $V_{\max}$

$$V_{\text{start}} = \left(\frac{\alpha_{\max}}{\alpha_{\min}}\right)^{1/3} (V_{\max} - V_t) + V_t. \tag{9}$$

Number of iterations: In the iterative procedure, in each step, the voltage of task m decreases by ΔV; $V_m(k) = V_{\text{start}} - k\Delta V$. Thus in the kth iteration, the voltage of the task j ($1 \leq j \leq n$) is

$$V_j(k) = \left(\frac{\alpha_{\min}}{\alpha_j}\right)^{1/3} (V_{\text{start}} - k\Delta V - V_t) + V_t. \tag{10}$$

The maximum number of iterations is given by $k_{\max} = (1/\Delta V)(V_{\text{start}} - V_{\min})$. By doing a binary search, the number of iterations can be reduced to $\log(k_{\max})$.

2) Slack Adjustment for Optimal Energy and Optimal Peak Power Assignment: Slack adjustment procedure for optimal energy and peak power assignment is very similar to the procedure for the low-complexity (approximate) algorithm. The main difference is that instead of using the approximate relation in (3), the exact relation in (2) is used. As a result, the voltage assignment in each step is not as straight forward as in (10). In the kth iteration, the voltage of task j is related to the voltage of the critical task by the following relation

$$\frac{V_j(k)(V_j(k) - V_t)^3}{(V_j + V_t)} = \frac{\alpha_{\min} V_m(k)(V_m(k) - V_t)^3}{\alpha_j (V_m + V_t)}. \tag{11}$$

Calculating $V_j(k)$ from $V_m(k)$ takes more than one step. This is because V_j can take a maximum of $k = (V_{\max} - V_{\min})/\Delta V$ different voltage values, and finding $V_j(k)$ from $V_m(k)$ using (11) takes a maximum of $\log k$ steps using binary search. This makes the overall complexity of the optimal assignment algorithms $O(\log k)$ times more than the complexity of the low-complexity algorithms.

IV. Aperiodic Task Assignment

In this section, we show how the aperiodic task assignments, earliest due date (EDD) and earliest deadline first (EDF) can be used to reduce the total energy and peak-power consumption.

A. Aperiodic EDD (Jackson's) Algorithm

The EDD algorithm schedules tasks with earlier due date first. Tasks have synchronous arrival times, but can have different computation times and deadlines. Jackson's algorithm for n tasks has a complexity of $O(n \log n)$ [18]. The guarantee test is $\sum_{k=1}^{j} \tau_k \leq d_j$, where d_j is the deadline for task j.

1) Low-Energy EDD Algorithm: If after the application of the EDD algorithm, a feasible solution exists and unused slack is available, then the following algorithm is invoked to decrease the energy consumption. In each iteration, the voltage of the critical task (or task m) is decreased, the voltages of the other tasks adjusted, and the finishing time of the task, f, compared with its deadline. If there is a violation scheduling task j at step k, i.e., there is a violation in the assignment $V_j(k)$, then the previous voltage value ($V_j(k-1)$) is chosen. Furthermore, since all tasks with a due date earlier than task j could have caused deadline violation of task j, all these tasks are assigned voltages corresponding to iteration $k-1$, $V_l = V_l(k-1)$ for $l = 1$ to j. The algorithm continues until voltages for all the tasks are determined.

```
for k = 1 to k_max
  update V_m(k), τ_m(k) for the critical task
  for each unscheduled task γ_j ∈ Γ
    update V_j(k), τ_j(k) for the tasks using
    eqn. (10)
    f_j = f_{j-1} + τ_j(k)
    if f_j > d_j(k)
      for l = 1 to j
        Schedule: V_l = V_l(k − 1)
```

Complexity: The worst case complexity of the algorithm is $O(nk_{\max} + n \log n)$. The optimal assignment algorithm has a complexity of $O(nk_{\max} \log k_{\max} + n \log n)$.

B. Aperiodic EDF Algorithm

EDF is a dynamic scheduling algorithm that at any instant executes the task with the earliest absolute deadline among all the ready tasks

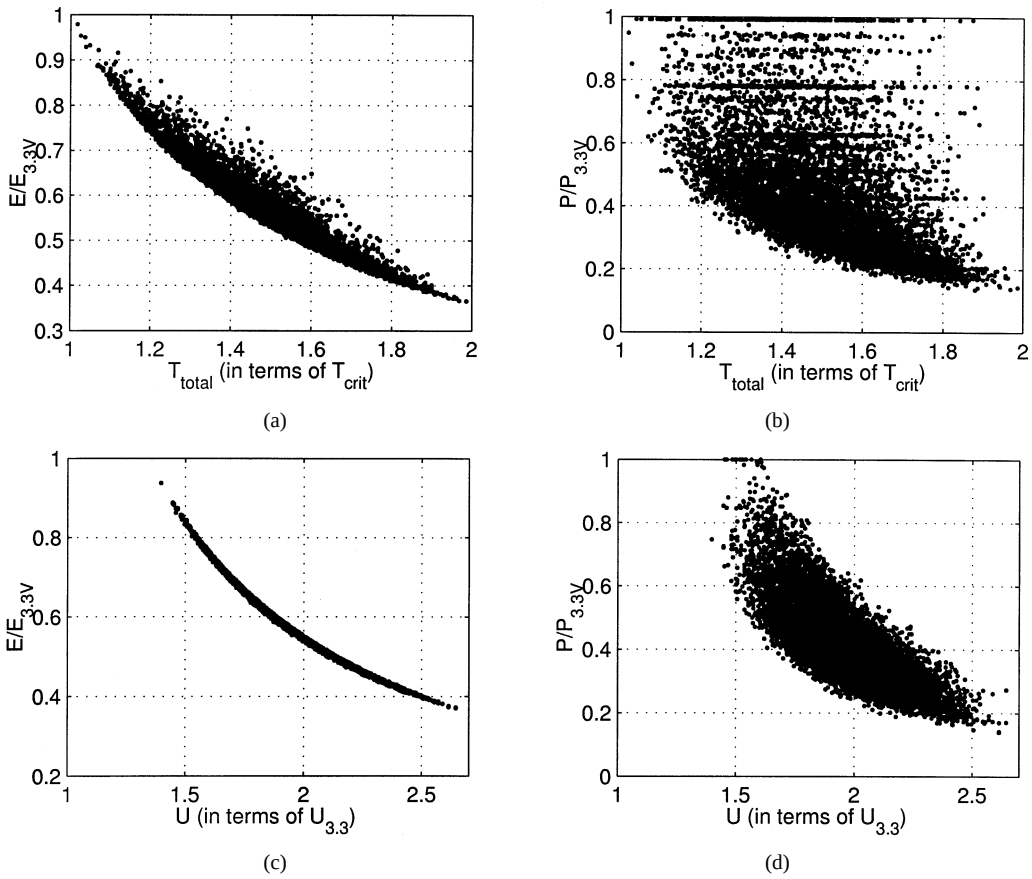

Fig. 1. Five task assignment problem for $\alpha_{\max} = 2\alpha_{\min}$, $s_{\max} = 10s_{\min}$, $p_{\max} = 2p_{\min}$. (a) EDD: normalized energy versus normalized delay. (b) EDD: normalized peak power versus normalized delay. (c) RM: normalized energy versus normalized utilization. (d) RM: normalized peak power versus normalized utilization.

[19]. Preemption is allowed; all tasks consist of a single job, and have different arrival times, computation times, and deadlines. The guarantee test is $\sum_{k=1}^{j} \tau_k(t) \leq d_j \ \forall j = 1, \ldots, n$. The complexity of EDF is $O(n^2)$.

1) Low-Energy Aperiodic EDF Algorithm: In this algorithm, whenever a new task arrives, the voltage values of the unscheduled tasks are updated according to the minimum energy equation provided that the deadline constraints are not violated. The voltage value for the task with the earliest deadline is found using binary search.

```
insert the newly arrived task into the
ordered queue.
τ_l is the task with the earliest deadline
in the queue.
binary search for each k, (k = 1 to k_max)
  update V_m(k), τ_m(k) for the critical task
  for each unscheduled task γ_j ∈ Γ
    update V_j(k), τ_j(k) using eqn. (10)
    f_j(k) = f_{j-1}(k) + τ_j(k)
    if f_j(k) > d_j and f_j(k-1) ≤ d_j
      Schedule: V_l = V_l(k-1).
```

Complexity: In each cycle, the task with the earliest deadline is scheduled. This takes $n \log(k_{\max})$ iterations. The overall complexity of the algorithm is $O(n^2 \log(k_{\max}))$. The complexity of the optimal algorithm is $O(n^2 \log^2(k_{\max}))$.

The proposed EDF algorithm is an on line algorithm and is not optimal with respect to energy minimization. Since arrival time information of the tasks is not available *a priori*, slack distribution of the tasks can not be done optimally. In fact, no on line algorithm can be optimal with respect to energy minimization if there are unknown parameters such as arrival time, execution time, etc.

V. Periodic Task Assignment

In this section, we show how the periodic task assignments, RM, and EDF, can be used to reduce the energy consumption.

A. Rate Monotonic Schedule

The rate monotonic (RM) scheduling algorithm is a simple scheme that assigns priorities to tasks according to their request rates [20]. Specifically, tasks with higher rates (that is with shorter periods) have higher priorities. In addition, deadline of a task (d_j) is equal to its period. The guarantee test is $U \leq U_{\text{lub}} = n(2^{1/n} - 1)$.

Although CPU utilization factor can be improved by increasing task computation times (decreasing voltages), there exists an upper bound $U_{lub}(\Gamma, A)$ of the processor utilization factor for a task set Γ under a given algorithm A. A feasible schedule is guaranteed when $U \leq U_{\text{lub}}$. If $U < U_{\text{lub}}$, there is an extra slack available and this slack can be used to decrease the voltage of the tasks, while still guaranteeing feasibility.

1) Low Energy RM Algorithm: In the proposed method, voltage assignment based on slack utilization is done first followed by RM scheduling. Both voltage assignment and RM scheduling are done off line. The voltage assignment algorithm finds the optimum value of k (in

$\log k$ steps using binary search) for which the utilization $U = \sum_{j=1}^{n} (\tau_j(k)/p_j) < U_{lub}$ and U is maximum.

Voltage assignment:
binary search for k, $k = 1$ to $k_{\max}$
  update $V_m(k)$, $\tau_m(k)$ for the critical task
  for each task $\gamma_j \in \Gamma$
    update $V_j(k)$, $\tau_j(k)$ using eqn. (10)
  if $U = \sum_{j=1}^{n} (\tau_j(k)/p_j) \leq U_{lub}$ and U is maximum
    return $Assignment(k)$
$Assignment(k)$: Assign $V_j(k)$ to task j, $1 \leq j \leq n$.
Scheduling: Schedule the tasks with RM.

Complexity: The worst case complexity of the algorithm is $O(n \log n + n \log k_{\max}))$. The optimal algorithm has a complexity of $O(n \log n + n \log^2 k_{\max})$.

B. EDF Schedule

The EDF algorithm is a dynamic scheduling rule that selects tasks according to their absolute deadlines. Tasks with earlier deadlines are given higher priorities. Deadlines of the tasks are assumed to be equal to their periods for simplicity. The guarantee test is $U = \sum_{i=1}^{n} (\tau_i/p_i) \leq 1$.

1) Low-Energy EDF Algorithm: In this algorithm, the voltage assignment is done off line followed by dynamic task scheduling using the EDF algorithm. The voltage assignment is based on the slack distribution method that is dictated by the minimum energy equation along with the feasibility condition $U \leq 1$.

Voltage assignment (Off line):
binary search for k value ($k = 1$ to $k_{\max}$)
  update $V_m(k)$, $\tau_m(k)$ for the critical task
  for each task $\gamma_j \in \Gamma$
    update $V_j(k)$, $\tau_j(k)$ using eqn. (10)
  if $U = \sum_{j=1}^{n} (\tau_j(k)/p_j) \leq 1$ and U is maximum
    return $Assignment(k)$
$Assignment(k)$: Assign $V_j(k)$ to task j, $1 \leq j \leq n$,
Scheduling: Schedule the tasks dynamically with EDF.

Complexity: The voltages of the tasks are determined off line with the complexity of $O(n \log(k_{\max}))$. The optimal algorithm requires $O(n \log^2(k_{\max}))$ complexity for the same computation. Since the complexity of EDF algorithm is $O(n^2)$, the overall complexity of both algorithms remain at $O(n^2)$.

VI. Results

In this section we describe the energy and peak-power savings obtained by our algorithm for randomly generated task configurations as well as for real-life examples.

A. Randomly Generated Task Configurations

We experiment with 10 000 different task configurations, where each configuration consists of five different tasks with task execution times randomly chosen between $s_{\min}$ and $10s_{\min}$ and switching activities randomly chosen between $\alpha_{\min}$ and $2\alpha_{\min}$.

TABLE I
Energy and Peak-Power Savings for 1) Aperiodic Schedules With Different Converter Utilizations (U_c) and $T_r = T_{\text{tot}}/T_{\text{crit}}$ and 2) Periodic Schedules With Different U_c and $U_r = U_{\text{lub}}/U_{3,3}$ for RM and $U_r = U/U_{3,3}$ for Periodic EDF

Algorithm		Energy Savings %		Peak Power Sav. %	
EDD	U_c	$T_r = 1.5$	$T_r = 2$	$T_r = 1.5$	$T_r = 2$
	0	43.5	63.5	55.6	85.7
	0.05	40	61.2	53	84.8
Aperiodic EDF	U_c	$T_r = 1.5$	$T_r = 2$	$T_r = 1.5$	$T_r = 2$
	0	43.5	57.4	51.7	85.7
	0.05	40	55.8	48.9	84.8
RM	U_c	$U_r = 1.5$	$U_r = 2$	$U_r = 1.5$	$U_r = 2$
	0	45.7	62.8	70.3	86.1
	0.05	41.7	60.3	68.1	84
Periodic EDF	U_c	$U_r = 1.5$	$U_r = 2$	$U_r = 1.5$	$U_r = 2$
	0	45.7	62.8	70.3	86.1
	0.05	41.7	60.3	68.1	84

TABLE II
Task Description for Real-Life Examples

Applications	# tasks	Range of WCETs(ms)
Avionics	17	1,000 - 9,000
INS	6	1,180 - 100,280
CNC	8	35 - 720

TABLE III
Energy and Peak Power Savings for Real-Life Examples for Different Converter Utilizations (U_c) and Different Switching Activities (α)

Example	Alg.	α	Energy Sav. %		Power Sav. %	
			$U_c = 0$	$U_c \neq 0$	$U_c = 0$	$U_c \neq 0$
CNC	RM	same	44.7	44.7	62.7	59.3
		diff	46	43.3	68.8	66
	EDF	same	64.7	64.7	82.8	81.6
		diff	65.4	64.3	85.5	84.6
INS	EDF	same	37.3	37.3	53.8	53
		diff	45	44.4	66	65.2
Avionics	EDF	same	22.2	22.2	33.9	33.4
		diff	25.9	25.5	43.7	43.3

EDD algorithm: To guarantee a feasible schedule, the deadline of task j is chosen as $d_j = d_{j-1} + k_j * s_j$, where k_j is an independent random variable between 1 and 2. As a result, for EDD, the task execution times are distributed normally between T_{crit} to $2T_{\text{crit}}$, with a mean of $1.5T_{\text{crit}}$ and standard deviation of $0.16T_{\text{crit}}$. Fig. 1(a) and (b) plot the normalized energy and peak power as a function of T_{total}. We see that as the delay increases, tasks can be assigned to lower voltages and as a result the normalized energy/power reduces. While the energy reduction is proportional to the reduction in V^2, power reduction is proportional to the reduction in $V(V - V_t)^2$. As a result, the peak-power reduction is (on the average) higher than the energy reduction. Another interesting point is that deadline constraints have a stronger effect on peak power reduction. The extreme case is when the deadline constraint of the task with the highest power consumption is T_{crit}, resulting in the peak-power reduction being zero. Finally, for all the task assignments, the peak power obtained by the optimal assignment is only 1% less than that obtained by the low complexity (approximate) assignment.

Aperiodic EDF algorithm: To guarantee a feasible schedule, the deadline of tasks is chosen between $d_{j-1} + s_j$ and $d_{j-1} + k_j * s_j$, where k_j is an independent random variable between 1 and 2. The arrival time

TABLE IV
COMPLEXITY OF THE PROCEDURE BEFORE AND AFTER APPLICATION OF THE ENERGY MINIMIZATION ALGORITHM

Algorithm			Complexity		
			before	after	
				low complexity	optimal
static	periodic	RM	$O(nlogn)$	$O(nlog(nk))$	$O(nlogn + nlog^2k)$
	aperiodic	EDD	$O(nlogn)$	$O(nk) + O(nlogn)$	$O(nklogk) + O(nlogn)$
dynamic	periodic	EDF	$O(n^2)$	$O(n^2)$	$O(n^2)$
	aperiodic	EDF	$O(n^2)$	$O(n^2logk)$	$O(n^2log^2k)$

of task j is chosen as $a_j = d_{j-2} + s_{j-1}$ to ensure feasibility. The energy and power savings are almost the same as EDD, since deadline constraints and variation in α are similar.

RM schedule: To guarantee a feasible schedule, the period of tasks is chosen between $U_{\text{lub}} * n * s_j$ and $U_{\text{lub}} * n * s_j * k_j$, where k_j is an independent random variable between 1 and 2. This ensures that task utilizations are distributed normally between $U_{\text{lub}} = 0.74$ to $0.5U_{\text{lub}} = 0.37$. Fig. 1(c) and (d) plot the normalized energy and peak power as a function of the normalized utilization factor. The normalized utilization factor varies from $U = 1/0.74 = 1.35$ to $U = 1/0.37 = 2.7$. We see that as the normalized utilization factor increases, tasks can be assigned to lower voltages and as a result the normalized energy/power reduces.

Periodic EDF algorithm: To guarantee a feasible schedule, the period of tasks is chosen between $n * s_j$ and $n * s_j * k_j$. This ensures that task utilizations are distributed normally between $U = 1$ to $U = 0.5$. The energy curve is similar to that obtained by the RM schedule. However, the energy savings are much higher for EDF, compared to RM, since the slack is fully utilized. For the 5 task assignment problem, the maximum feasible utilization is 0.74 for RM and 1 for EDF. Thus, for $U_{3.3} = 0.74$, the energy saving is zero for RM (since $U_{lub}/U_{3.3} = 1$) and is $U/U_{3.3} = 1/0.74 = 1.35$ for EDF algorithm. This corresponds to a 35% energy saving.

The average energy savings and peak power savings for different converter utilizations (U_c) are summarized in Table I. We define converter utilization U_c to be the fraction of slack that is wasted by the converter delay and the delay to vary clock frequency. The savings reduce when converter utilization is taken into account. For instance, for the aperiodic EDF, when $T_{\text{total}} = 1.5T_{\text{crit}}$, the average energy saving is 43.5% when $U_c = 0$ and 40% when $U_c = 0.05$. The corresponding peak power savings are higher since power is proportional to V^3 (while energy is proportional to V^2).

B. Real Life Examples

We have also considered some real-time applications, computerized numerical control (CNC) [21], inertial navigation system (INS) and avionics task set [22] (Table II). The deadlines of the tasks are assumed to be equal to their periods (for simplification). The worst case delay to vary the clock frequency and the supply voltage is taken to be 10 μs [10]. Since all the tasks are periodic, the RM and EDF schedules are applied. $U_{3.3}$ is 0.488 for the CNC example, 0.736 for the INS example, and 0.85 for the avionics example. The U_c values for CNC, INS, and avionics are 2.5%, 0.8%, and 0.26%, respectively.

In these experiments we consider two cases. In the first case, switching activities are assumed the same for all the tasks and in the second case, switching activities are chosen between $\alpha_{\min}$ and $2\alpha_{\min}$. When switching activities are the same, the minimum energy solution corresponds to the case where all the task voltages are the same. So, there is no delay in the converter and energy savings do not change. When switching activities are different, the energy savings change with the converter delay. Table III summarizes our results for those three cases. For instance, when EDF is applied to INS, the energy savings drop from 45% to 44% when switching activities are taken into account.

VII. CONCLUSION

In this paper, we present an extension of existing task scheduling algorithms (EDD, EDF, RM) to minimize energy/power. We theoretically determine the relation between the operating voltages for the minimum energy and minimum peak power assignment using the Lagrange multiplier method and develop heuristics that use this relation. We demonstrate the effectiveness of our procedure using randomly generated examples, as well as real-life cases.

Application of our procedure increases the complexity of the scheduling algorithms mildly. For instance, for RM scheduling the complexity increases from $O(n \log n)$ to $O(n \log(nk))$ for the low-complexity algorithm and to $O(n \log n + n \log^2 k)$ for the optimum assignment algorithm, where n is the number of tasks and k is the number of iterations. There is a trade off between energy/power savings and the complexity of the algorithms. The value of k depends on the voltage converter sensitivity; the higher the sensitivity, the higher the running time of the algorithm and closer the solution is to the optimal assignment. Table IV lists the complexity of the procedure before and after application of the energy minimization algorithm.

REFERENCES

[1] J. M. Rabaey and M. Pedram, *Low Power Design Methodologies.* Norwell, MA: Kluwer , 1996, pp. 46–61.
[2] V. Gutnik and A. P. Chandrakasan, "Embedded power supply for low-power DSP," *IEEE Trans. VLSI Syst.*, vol. 5, pp. 425–435, Dec. 1997.
[3] F. Yao, A. Demers, and S. Shenker, "A scheduling model for reduced CPU energy," in *Proc. IEEE Annu. Found. Comput. Science*, 1995, pp. 374–82.
[4] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processor," in *Proc. Int. Symp. Low-Power Design*, 1998, pp. 197–202.
[5] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M.-B. Srivastava, "Power optimization of variable voltage core-based systems," in *Proc. Design Automation Conf.*, 1998, pp. 176–81.
[6] G. Quan and X. Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors," in *Proc. Design Automation Conf.*, 2001, pp. 828–833.
[7] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling for minimizing energy," in *Proc. Int. Symp. Low-Power Design*, 2001, pp. 279–282.
[8] J. Pouwelse, K. Langendoen, and H. Sips, "Energy priority scheduling for variable voltage processors," in *Proc. Int. Symp. Low-Power Design*, 2001, pp. 28–33.
[9] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for reduced CPU energy," in *Proc. USENIX Symp. Operating Syst. Design and Implementation*, 1994, pp. 13–23.
[10] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scheduling algorithms," in *Proc. Int. Symp. Low-Power Design*, 1998, pp. 76–81.
[11] D. Grunwald *et al.*, "Policies for dynamic clock scheduling," in *Proc. USENIX Symp. Operating Syst. Design Implementation*, 2000, pp. 73–86.

[12] T. Pering, T. Burd, and R. Brodersen, "Voltage scheduling in the lpARM microprocessor system," in *Proc. Int. Symp. Low-Power Design*, 2000, pp. 96–101.
[13] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," in *Proc. Design Automation Conf.*, 1999, pp. 134–139.
[14] Y. Shin, K. Choi, and T. Sakurai, "Power optimization of real-time embedded systems on variable speed processors," in *Proc. Int. Conf. Computer-Aided Design*, 2000, pp. 365–368.
[15] S. Lee and T. Dakurai, "Run-time voltage hopping for low power real systems," in *Proc. Design Automation Conf.*, 2000, pp. 806–809.
[16] D. Shin, J. Kim, and S. Lee, "Low energy intra task voltage scheduling using static timing analysis," in *Proc. Design Automation Conf.*, 2001, pp. 438–443.
[17] D. Shin and J. Kim, "A profile-based energy-efficient intra task voltage scheduling algorithm for hard real-time applications," in *Proc. Int. Symp. Low Power Design*, 2001, pp. 271–274.
[18] J. R. Jackson, "Scheduling a production line to minimize maximum tardiness," Univ. Calif., Los Angeles, Manage. Sci. Res. Project 43, 1955.
[19] W. Horn, "Some simple scheduling algorithms," *Naval Res. Logistics Quart.*, vol. 21, pp. 177–185, 1974.
[20] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming an a hard real time environment," *J. ACM*, vol. 20, pp. 46–61, Jan. 1973.
[21] N. Kim *et al.*, "Visual assessment of a real-time system design: A case study on a CNC controller," in *Proc. IEEE Real-Time Syst. Symp.*, Dec. 1996, pp. 300–310.
[22] A. Burns, K. Tindell, and A. Wellings, "Effective analysis for engineering real-time fixed priority schedulers," *IEEE Trans. Software Eng.*, vol. 21, pp. 475–480, May 1995.

Designing Fast On-Chip Interconnects for Deep Submicrometer Technologies

Razak Hossain, Fabrizio Viglione, and Marco Cavalli

***Abstract*—This paper proposes a solution to the problem of improving the speed of on-chip interconnects, or wire delay, for deep submicron technologies where coupling capacitance dominates the total line capacitance. Simultaneous redundant switching is proposed to reduce interconnect delays. It is shown to reduce delay more than 25% for a 10-mm long interconnect in a 0.12-μm CMOS process compared to using shielding and increased spacing. The paper also proposes possible design approaches to reduce the delay in local interconnects.**

***Index Terms*—Capacitive coupling, critical wire, deep submicrometer technology, on-chip interconnect, redundant wire, shielding, simultaneous redundant switching, spacing, wire delay.**

I. Introduction

As our ability to fabricate deep submicron (DSM) processes advances, we start to encounter problems from the convergence of two different factors. First, chips run faster, with global wire lengths increasing, or at least, staying constant. As wire delays do not scale as well as transistor delays, interconnect delay accounts for an increasing proportion of the total delay [1]. A recent state-of-the-art microprocessor implementation, for example, dedicates entire pipeline stages to drive global signals across the chip [2]. The second problem that emerges is that wires run much closer to each other and are stacked more deeply. This causes most of the capacitance of the wire to be coupled to other data wires, instead of to fixed power or ground nodes, such as the substrate.

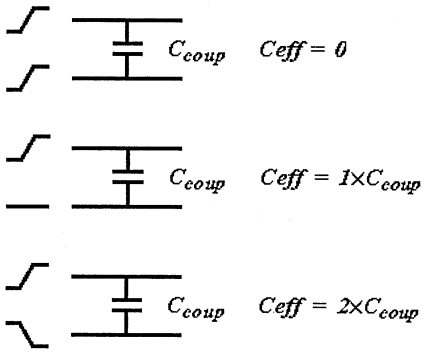

Fig. 1. Effective coupling capacitance under different terminal conditions.

The delay for a coupled lines is complicated as the effective value of a capacitor, with capacitance $C_{\rm coup}$ coupled between two data lines, is no more constant but dynamically dependent on the conditions at the terminals of the capacitor. When the two lines switch in opposite directions at exactly the same time the effective capacitance become $2 \times C_{\rm coup}$, whereas, when they switch in the same direction at the same time the capacitance is 0. If one line switches when the other line stays at a fixed value, the effective capacitance is $1 \times C_{\rm coup}$. This condition is illustrated in Fig. 1. Since wires rarely switch exactly at the same instant, these extreme values are not often encountered. Still, the effective coupling capacitance does dynamically vary. In order to understand how these line capacitance variations can alter delay, let us recall the 50% delay expression for a distributed RC line [3]

$$T_{50\%} = 0.4 \times R_{\rm int} \times C_{\rm int} + 0.7 \times (R_{\rm tr} \times C_{\rm int} + R_{\rm tr} \times C_L + R_{\rm int} \times C_L)$$

where $R_{\rm tr}$ is the on-resistance of the driver transistor, $R_{\rm int}$ and $C_{\rm int}$ are the resistance and capacitance of the wire, and C_L is the load capacitance. Capacitive coupling is more severe in long wires, for which $C_{\rm int} \gg C_L$. This allows the delay expression to be simplified to [3]

$$T_{50\%} = (0.4 \times R_{\rm int} + 0.7 \times R_{\rm tr}) \times C_{\rm int}.$$

The expression illustrates that line delay is directly proportional to its load capacitance. Thus, if coupling were to cause the capacitance of a line, $C_{\rm int}$, to vary from 50% to 150% of its steady-state value, the line delay would vary by a factor of 3. Obviously, there are delay advantages in reducing the line capacitance. Before presenting our approach for reducing line delay in capacitively coupled interconnects, we review the existing techniques to reduce wire delays and capacitive crosstalk.

The standard approach to driving long interconnects is to use repeaters. Repeaters, which divide long wires into shorter segments, have been shown to reduce the delay for long, distributed *RC* [3] and *RLC* lines [4]. This follows as both the capacitance and the resistance of a wire increases linearly with its length, leading to a quadratic impact on delay. Since long interconnects are heavily loaded, using low-swing signaling seems to be another design approach worth considering. While low-swing signaling has been shown to be effective in reducing power dissipation, it has not lead to delay reductions [5] (the paper does not consider coupling capacitance on

Manuscript received August 20, 2001; revised May 2002.

R. Hossain is with STMicroelectronics, Inc., San Diego, CA 92129 USA (e-mail: razak.hossain@st.com).
F. Viglione was with STMicroelectronics, Inc., San Diego, CA 92129 USA, he is now a Consultant.
M. Cavalli is with STMicroelectronics, Inc., Srl. 20041 Agrate, Italy.

Digital Object Identifier 10.1109/TVLSI.2003.810781