A Low Power Scheduling Scheme With Resources Operating at Multiple Voltages

Ali Manzak and Chaitali Chakrabarti, *Member, IEEE*

***Abstract*—This paper presents resource *and* latency constrained scheduling algorithms to minimize power/energy consumption when the resources operate at multiple voltages (5 V, 3.3 V, 2.4 V, and 1.5 V). The proposed algorithms are based on efficient distribution of slack among the nodes in the data-flow graph. The distribution procedure tries to implement the minimum energy relation derived using the Lagrange multiplier method in an iterative fashion. Two algorithms are proposed, 1) a low complexity $O(n^2)$ algorithm and 2) a high complexity $O(n^2 \log(L))$ algorithm, where n is the number of nodes and L is the latency. Experiments with some HLS benchmark examples show that the proposed algorithms achieve significant power/energy reduction. For instance, when the latency constraint is 1.5 times the critical path delay, the average reduction is 39%.**

***Index Terms*—Allocation, data-flow graph, Lagrange multiplier method, low power, multiple voltages, resource and latency constraint, scheduling.**

I. Introduction

POWER considerations have become an increasingly dominant factor in the design of both portable and desk-top systems. An effective way to reduce power consumption is to lower the supply voltage level of a circuit. Reducing the supply voltage, however, increases the circuit delay and reduces the throughput. To maintain the throughput, parallelism and/or pipelining has to be incorporated [1]. The resulting circuit consumes lower average power while meeting the global throughput constraint at the cost of increased circuit area. Another way of maintaining the throughput is to use resources operating at multiple voltages [2]–[7]. This has the advantage of allowing modules on the critical paths to be assigned to the highest voltage levels (thus meeting the required timing constraints) while allowing modules on noncritical paths to be assigned to the lower voltages (thus reducing the power consumption). Supporting multiple voltages on chip poses many challenges, i.e., incorporation of multiple-power-distribution grids, efficient-level converters, etc. However, the viability of this method has been successfully demonstrated in the design of a MPEG-4 video codec in [8].

In this paper, we address the problem of scheduling a data-flow graph (DFG) under resource *and* latency constraint for the case when the resources operate at multiple voltages. We assume that the resources respect different voltage-delay curves. The scheduling algorithm minimizes power/energy consumption for the case when the resources operate at multiple voltages (5 V, 3.3 V, 2.4 V, and 1.5 V). It is worth mentioning that the operating voltages do not have to be restricted to 5 V, 3.3 V, 2.4 V, and 1.5 V. In fact, the resources can operate on any voltage for which the corresponding delay is known.

The proposed algorithm operates in two passes. In the first pass, minimum-time, resource-constrained scheduling is done. In the second pass, the difference between the given latency and the time needed by the resource-constrained schedule (obtained in the first pass) is distributed among the nodes in a way that minimizes the total power/energy consumption. The distribution procedure (derived using the Lagrange multiplier method) uses the energy/delay (E/D) ratio of the nodes to distribute the slack. The procedure is implemented by an iterative algorithm, where in each iteration, increasing number of resources with high E/D ratio are disabled. The iterations continue till there is a timing violation. Two algorithms are proposed, 1)$O(n^2)$ algorithm and 2) a more complex $O(n^2 \log L)$ algorithm, where n is the number of nodes and L is the latency. The $O(n^2 \log L)$ algorithm gives a schedule with lower energy compared to the $O(n^2)$ algorithm. Experiments with some HLS benchmarks (DIFFEQ, AR lattice, EW filter, FIR Filter, FFT4 and DCT) show the effectiveness of these approaches in reducing power/energy. When the latency constraint is tight,[1] the average reduction is 17.5%, when the latency constraint is 1.5 times the critical path delay, the average reduction is 39% and when the latency constraint is two times the critical path delay, the average reduction is 58.5%.

There are several scheduling algorithms for multiple-voltage resources in the literature today [2]–[7]. These algorithms can be classified into i) *only* latency-constrained (i.e., latency is a hard constraint and resources are minimized) [3], [4], ii) *only* resource-constrained (i.e., resource is a hard constraint) [5], and iii) latency *and* resource constrained (i.e., both latency and resource are hard constraints) [6], [7]. While the (only) latency-constrained and (only) resource-constrained algorithms have polynomial or pseudopolynomial time complexity, the latency *and* resource-constrained algorithms are based on integer linear programming and have (worst case) exponential time complexity. In this paper, we propose a heuristic algorithm for latency *and* resource-constrained scheduling that produces comparable results with only polynomial time complexity.

The rest of the paper is organized as follows. Section II presents the definitions and the Lagrange formulation. Section III describes the algorithms and illustrates them with

Manuscript received January 9, 2001. This work was supported in part by the Center for Low Power Electronics (CLPE).

A. Manzak is with Lattice Semiconductor Corporation, San Jose, CA 95134-210 USA (e-mail: amanzak@latticesemi.com).

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

Publisher Item Identifier S 1063-8210(02)00487-0.

[1]Tight latency constraint corresponds to the minimum time required to schedule all the nodes of the DFG.

examples. Section IV includes the results on some HLS benchmark examples. Section V concludes the paper.

II. Preliminaries

A. Definitions

The input to our algorithm is a data flow graph, a timing constraint, and a resource constraint.

Timing Constraint: This is the time available to execute the operations in the data flow graph. It is also referred to as the latency constraint.

Resource Constraint: This is specified by the number of resources for each type, where each type of resource can only be operated at a specific voltage. The corresponding energy and delay values are also given. Examples of resources include adder/subtractor (denoted by A), multiplier (denoted by M), etc.

In addition, we assume that a level shifter is needed between resource A and resource B only if resource A operates at a lower voltage compared to resource B. The number of level shifters is not user defined. In fact, the proposed algorithm tries to reduce the number of level shifters.

B. Slack Distribution Using the Lagrange Multiplier Method

In the proposed algorithm, the Lagrange multiplier method is used to distribute the slack among the nodes in the critical path. The relation between the voltages of the nodes in the critical path is derived in the following way.

The delay of a resource is determined by the delay of the gates on the critical path. Let τ_i be the delay of gate i operating at voltage V and let C_{L_i} be the load capacitance of gate i

$$\tau_i = k' C_{L_i} \frac{V}{(V - V_t)^2}. \tag{1}$$

Then the delay of the resource j operating at voltage V_j is equal to sum of the delay of the gates on the critical path

$$D_j = \sum_{i=1}^{n} \tau_i = \frac{k' V_j}{(V_j - V_t)^2} \sum_{i=1}^{n} C_{L_i} = \frac{k' C_{c_j} V_j}{(V_j - V_t)^2} \tag{2}$$

where C_{c_j} is the sum of the capacitances of the gates on the critical path of resource j.

The energy of resource j operating at V_j volts is given by

$$E_j = \alpha_j C_j V_j^2 \tag{3}$$

where α_j is the average switching activity of the resource, and C_j is the total load capacitance of the resource. Note that C_j is typically much larger than C_{c_j}.

Our aim is to minimize E_{total} subject to the time constraint T_{total}. We use the Lagrange multiplier method to determine the supply voltage of each node

$$\begin{aligned} \frac{\partial E_{\text{total}}}{\partial V_1} &= \lambda \frac{\partial T_{\text{total}}}{\partial V_1}, \\ &\vdots \\ \frac{\partial E_{\text{total}}}{\partial V_n} &= \lambda \frac{\partial T_{\text{total}}}{\partial V_n} \\ T_{\text{total}} &= \sum_{j=1}^{n} \tau_j = \text{const}. \end{aligned}$$

We find that E_{total} is minimum when the following condition is satisfied among the nodes in the critical path

$$\frac{\alpha_1 C_1 V_1 (V_1 - V_t)^3}{C_{c_1}(V_1 + V_t)} = \cdots = \frac{\alpha_n C_n V_n (V_n - V_t)^3}{C_{c_n}(V_n + V_t)}.$$

Since computing the supply voltage for the above equation can be a computational burden, we simplified the equation. For $V_j > 3V_t$, this is approximated to

$$\frac{\alpha_1 C_1 (V_1 - V_t)^3}{C_{c_1}} = \cdots = \frac{\alpha_n C_n (V_n - V_t)^3}{C_{c_n}}. \tag{4}$$

The error as a result of approximation is ≈0.1% [10].

Since $\alpha_j C_j / C_{c_j}$ ratio is a constant for each resource, from (2) and (3) we have

$$\frac{\alpha_j C_j}{C_{c_j}} = \frac{E_j}{D_j V_j (V_j - V_t)^2}.$$

Let V_{ref} be the reference voltage. Then

$$\left[\frac{E_1}{D_1}\right]_{V_{\text{ref}}} (V_1 - V_t)^3 = \cdots = \left[\frac{E_n}{D_n}\right]_{V_{\text{ref}}} (V_n - V_t)^3. \tag{5}$$

To satisfy (5), nodes with high E/D (such as wide multipliers) have to be assigned to lower-voltage resources. This is the basis of our slack distribution algorithm.

III. Resource and Latency-Constrained Scheduling: Algorithm and Examples

Algorithm Overview: In a nutshell, the proposed resource and latency-constrained algorithm operates in two passes. In the first pass, resource-constrained scheduling is done in a way that minimizes the computation time. In the second pass, the slack between the given latency and the computation time obtained by the scheduling algorithm in the first pass is distributed to the nodes such that the total power/energy consumption is minimum. The Lagrange multiplier method described in Section II-B, is used to find the optimal distribution of slack between the nodes.

A. First Pass: Minimum-Time Resource-Constrained Algorithm

The minimum-time resource-constrained algorithm schedules the nodes such that the computation time is minimum. To ensure that a feasible schedule is obtained, it calculates the number of ready nodes in each cycle in a special way. Define:

- $C_j(i)$: the minimum number of control cycles required to schedule all the nodes of type j in the ready set corresponding to the ith control cycle
- $A_{j,\,v}(i)$: the number of ready nodes of type j that will be scheduled in control cycle i to the resources operating at $v$volts.

In this scheme, $C_j(i)$ is first calculated and then $C_j(i)$ is used to calculate $A_{j,\,v}(i)$. Use of $C_j(i)$ results in the ready nodes being scheduled in a way that takes minimum time, thereby increasing the feasibility of the algorithm. Power consumption reduction is not considered in this step of the algorithm.

The proposed algorithm is list based [9]. The nodes in a ready set are prioritized based on the freedom: $A_{j,\,v}(i)$ nodes with the lowest freedom are chosen among the ready nodes. Then the

nodes are scheduled such that if the freedom of a node is low, it is assigned to a high-voltage resource and if the freedom of a node is high, it is assigned to a low-voltage resource. This is implemented in two different ways. Let $R_{j,v}(i)$ be the number of available resources of type j operating at v volts in cycle i. Then in the first scheme, $A_{j,v}(i)$ nodes with higher freedom are scheduled to the $R_{j,v}(i)$ resources with lower voltage, while in the second scheme $A_{j,v}(i)$ nodes with lower freedom are scheduled to the $R_{j,v}(i)$ resources with higher voltage. The first scheme has the advantage of lower-power consumption since more low-voltage resources are utilized. However, it can result in an infeasible solution when the given latency is tight. The second scheme has higher feasibility of scheduling but possibly higher-power consumption. In our procedure, both the schemes are implemented, and the schedule which is feasible and takes lower time is chosen for the minimum-time algorithm (first pass) and the schedule which is feasible and consumes lower power is chosen for the low power algorithm (second pass).

Finally, if the freedom of a node is higher than the delay of the resource to which it is assigned, the node can be scheduled to a lower voltage resource, if available. While this improves the power consumption, it can result in increased number of level shifters, since the extra freedom allows the node to be scheduled to a lower voltage than its children.

The computation time obtained by application of the minimum-time resource-constrained algorithm is referred to as T_{low}. Thus if the latency constraint $L < T_{\text{low}}$, a feasible solution cannot be obtained.

Algorithm *Schedule*

1. Make/update the ready table.
2. Calculate $A_{j,v}(i)$.
3. Schedule $A_{j,v}(i)$ nodes to $R_{j,v}(i)$ resources.

(a) Choose $A_{j,v}(i)$ nodes with the lowest freedom.

i) Start scheduling from the highest-freedom node to the resources with the lowest voltage.

ii) Start scheduling from the lowest-freedom node to the resources with the highest voltage.

iii) For the minimum time algorithm, choose the schedule [among i) and ii)] that is feasible and results in lower time. For the low-power algorithm, choose the schedule [among i) and ii)] that is feasible and results in lower power.

b) If the delay of the scheduled node is less than the freedom of the node, reschedule the node to the lower available voltage resource.

Calculation of Parameter $A_{j,v}(i)$: Recall that $A_{j,v}(i)$ is the number of ready nodes of type j that will be scheduled in control cycle i to the resources operating at v volts. Define $N_{j,v}(i)$ as the number of nodes of type j that can be scheduled to the resource of type j operating at v volts during the period cycle i to cycle $i + C_j(i) - 1$. To calculate $A_{j,v}(i)$, we need to calculate $C_j(i)$ and $N_{j,v}(i)$. Since $R_{j,v}(i)$ is the number of available resources of type j in control cycle i, $A_{j,v}(i) = \min\{N_{j,v}(i), R_{j,v}(i)\}$.

Calculation of $C_j(i)$ and $N_{j,v}(i)$: Let $D_{j,v}$ be the delay of a resource of type j operating at v volts. Let $R_{j,v,l}(i) = 1$ if the lth resources of type j operating at v volts is available in control cycle i, otherwise $R_{j,v,l}(i) = 0$. Let $N_j(i)$ be the number of ready nodes of type j in cycle i. Then $N_j(i)$, $C_j(i)$ and $N_{j,v}(i)$ are linked as follows

$$N_{j,v}(i) = \sum_{l=1}^{R_{j,v}(i)} \left\lfloor \frac{\sum_{k=i}^{C_j(i)+i-1} R_{j,v,l}(k)}{D_{j,v}} \right\rfloor \tag{6}$$

$$N_j(i) \le \sum_{v=v_{\max}}^{v_{\min}} N_{j,v}(i). \tag{7}$$

So, given $N_j(i)$, $C_j(i)$, and $N_{j,v}(i)$ can be calculated. Note that these parameters have to be calculated for each resource j. This will be explained with the following example.

Example 1: Let the resources consist of a 5 V multiplier, a 3.3 V multiplier, and a 2.4 V multiplier. Thus, $R_{M,5} = R_{M,3.3} = R_{M,2.4} = 1$. Assume that all the resources are available ($R_{M,v,1}(i) = 1$ for all v). The delays are related by $D_{M,5} = D$, $D_{M,3.3} = 2D$, and $D_{M,2.4} = 3D$, where D is the delay of a multiplier relative to an adder operating at 5 V. In $C_M(i) = 3$ control cycles, three nodes can be assigned to a 5 V multiplier, one node to a 3.3 V multiplier and one node to a 2.4 V multiplier. Thus $N_M(i) = N_{M,5}(i) + N_{M,3.3}(i) + N_{M,2.4}(i) = 3+1+1 = 5$. Thus, in three control cycles, a maximum of five multiplication nodes can be scheduled under the given resource constraint. So, given $C_M(i)$, we can easily calculate $N_M(i)$.

Now consider calculating $C_M(i)$ given $N_M(i)$. Let $N_M(1) = 7$. Then under the same resource constraint, $C_M(1)$ can be calculated using (1) and (2)

$$7 \le \left\lfloor \frac{C_M(1)}{3} \right\rfloor + \left\lfloor \frac{C_M(1)}{2} \right\rfloor + C_M(1).$$

$C_M(1) = 4$ satisfies the above equation, resulting in $N_{M,5}(1) = 4$, $N_{M,3.3}(1) = 2$, and $N_{M,2.4}(1) = 1$. □

B. Second Pass: Low-Power Algorithm

At the end of the first pass, if $L > T_{\text{low}}$, each node has a nonzero slack. The objective of the low-power algorithm is to distribute the available slack between the nodes (i.e., determine the voltage assignment of the nodes) such that the latency constraint is satisfied and the minimum energy condition (5) is satisfied as much as possible. Recall that the voltage assignment is directly related to the operating voltage of the corresponding hardware resource. The procedure is iterative; in each iteration, increasing number of resources with high E/D values are disabled and the nodes are scheduled. The order in which the resources are disabled is determined using (5). After each iteration, the minimum energy condition (5) is better satisfied. The iterations continue till there is a timing violation. We motivate our procedure with the help of the following example.

Example 2: Consider a DFG with an addition node being fed to a multiplication node. The resource constraint consists of multipliers and adders operating at 5 V, 3.3 V, 2.4 V, and

1.5 V, and the latency is 17 control cycles. The energy-delay ratio of the multiplier at $V_{\rm ref} = 5$ V is $[E_M/D_M]_{5\,\rm V} = 8$, and that of the adder is $[E_A/D_A]_{5\,\rm V} = 1$. For other values of $V_{\rm ref}$, $[(E_M/D_M)/(E_A/D_A)]_{V_{\rm ref}}$ is still 8. From (5), the condition for minimum energy is $8(V_M - V_t)^3 = (V_A - V_t)^3$ or

$$2(V_M - V_t) = (V_A - V_t) \tag{8}$$

where $V_M(V_A)$ is the voltage that is assigned to the multiplication (addition) node. Our aim is to schedule the nodes without violating the timing constraint such that the minimum energy equation is satisfied as much as possible.

Assume that multipliers and adders operating at 5 V, 3.3 V, and 2.4 V are available. In the beginning, let both the multiplication and the addition node be assigned to the 5 V resources. For this assignment, the computation time is 6, which is less than the latency L. Furthermore, for $V_t = 1$, the left-hand side of (8) is 8, which is larger than the right-hand side of (8) which is 4. In order to satisfy (4) better, we choose to assign the multiplication node to the 3.3 V multiplier. Choosing the 3.3 V multiplier is equivalent to disabling the 5 V multiplier from the set of resources. For this assignment, the left-hand side is 4.6 which is closer to 4 but still higher. The corresponding computation time of 10 is still less than L and so we assign the multiplication node to the 2.4 multiplier. Choosing the 2.4 multiplier is equivalent to disabling the 5 V and the 3.3 V multiplier from the set of resources. The computation time is now 16 and the left-hand side is 2.8, which is less than the right-hand side. So it is now the addition node that has to be assigned to a lower-voltage resource, namely, a 3.3 V adder. With this assignment, the latency is 17 and the right-hand side is 2.3 which is closer to the left-hand side. Note that while the minimum energy condition is not satisfied (the left-hand side and right-hand side of (8) are close but not equal), this assignment is the closest to the minimum-energy solution. We cannot lower the voltages any further since that would cause a latency violation. From this example, we see how (8) helps us to determine the order in which the resources should be disabled so that the minimum energy condition is satisfied as much as possible. □

1) Priority Assignment: The priority of which resources to disable first is determined using (5) and the energy values of [4] that are quoted in Table I. The priorities are given in Table II. According to this table, the multipliers operating at 5 V have the highest chance of being disabled followed by multipliers operating at 3.3 V, followed by adders operating at 5 V, followed by adders operating at 3.3 V, etc.

The proposed algorithm disables resources in the order of their priority. For each configuration, it schedules the nodes and checks if its computation time $T_{\rm alg} < L$. If it is true, then it disables the resources with the next highest priority and reschedules the nodes. If it is not true ($T_{\rm alg} > L$), then the specific resource cannot be disabled for all the control cycles. We next describe two algorithms which differ in determining until what control cycle the specific resource can be disabled.

2) Description of Algorithms 1 and 2: Assume that a feasible solution exists when resources with priorities 1 through k are disabled and that there is a timing violation when resources with priorities 1 through $k+1$ are disabled. For the feasible schedule, let T_K be the computation time and let the finish time of node j be f_j. Then the slack of each node is $t_{\rm slack} = L - T_K$ and node j can finish as late as $f_j + t_{\rm slack}$. Algorithms 1 and 2 are used to determine until what control cycle the resource with priority $k+1$ can be disabled without violating the timing constraint.

TABLE I
ENERGY E (IN pJ) AND NORMALIZED DELAY, D (FOR $t_c = 20$ ns) FOR THE MODULES IN [4]

	5.0V		3.3V		2.4V		1.5V	
Module	D	E	D	E	D	E	D	En.
mult16	5	2504	9	1090.7	15	576.9	36	225.3
add16	1	118	2	51.4	3	27.2	8	10.6
sub16	1	118	2	51.4	3	27.2	8	10.6

TABLE II
ORDER IN WHICH THE RESOURCES ARE DISABLED

Priority	Resource
1	5V multiplier
2	3.3V multiplier
3	5V adder
4	3.3V adder
5	2.4V multiplier
6	2.4V adder

Algorithm 1: This is a one-pass algorithm. Here we start scheduling the nodes with the resource with priority $k+1$ disabled, and after an assignment check if the f_j is greater than $f_j + t_{\rm slack}$. If it is greater, a conflict has occurred, and node j and the remaining unassigned nodes are scheduled assuming that the resource with priority $k+1$ has not been disabled.

Algorithm 2: This is an iterative algorithm, and consequently more complex compared to Algorithm 1. The procedure consists of first disabling the resource with priority $k+1$ from control cycle 1 to $T_K/2$ and checking for timing violations. If $L > T_{\rm alg}$, then the resource is disabled from control cycle 1 to $3T_K/4$ $(=T_K/2 + T_K/4)$. If the new $T_{alg} > L$, then the resource is disabled from control cycle 1 to $5T_K/8$ $(=3T_K/4 - T_K/8)$, and so on. After $\log(vfL)$ steps, where v is the number of voltage levels, f is the number of resources and L is the latency, the algorithm determines which resources to disable such that the computation time is less than L but as close to L as possible. Thus, Algorithm 2 iteratively schedules the nodes until unused slack is minimum. As a result, its complexity is $O(\log L)$ times larger than that of Algorithm 1. We explain the operation of Algorithm 2 with the help of the following example.

Example 3: Let the latency $L = 30$. Let the computation time when only the 5 V multiplier is disabled is 16, and when both the 5 V and 3.3 V multipliers are disabled is 35. Thus, while the 5 V multiplier can be disabled for the whole time, the 3.3 V multiplier can only be disabled for part of the time. Algorithm 2 determines until what control cycle the 3.3 V multiplier can be disabled in the following way: in the first iteration, the algorithm disables the 3.3 V multiplier from 1 to 8 (=16/2) cycles and completes the schedule in 24 cycles. Since there is a positive slack, the algorithm disables the 3.3 V multiplier from 1 to 12 (=8 + 16/4) cycles. The latency is now 31 which is greater than L. So the algorithm disables the multiplier from 1

to 10 (=12 − 16/8) cycles and completes the scheduling in 28 cycles. Finally, the algorithm disables the multiplier from 1 to 11 (=10 + 16/16) cycles and completes in 29 cycles, which is as close as to $L = 30$ as possible. □

The low-power algorithm is summarized as follows.

Algorithm *Low Power*
1. Make the priority table for resource disabling.
2. Disable resources with the highest priority.
3 Schedule all the nodes using **Algorithm** *Schedule*.
4. If $T_{\text{alg}} > L$
   invoke Algorithm 1 or Algorithm 2
   else
   disable the resources with the next highest priority.
   Go to Step 3.

Summary: The proposed resource *and* latency constrained scheduling algorithm operates in two passes in the following way.

First Pass: Algorithm *Schedule* (minimum-time version)

Second Pass: Algorithm *Low Power* (invokes Algorithm 1 or 2)

We refer to the version when Algorithm 1 is invoked in Algorithm *Low Power* as Algorithm **LC** (or the low-complexity algorithm), and the version when Algorithm 2 is invoked as Algorithm **HC** (or the high-complexity algorithm).

C. Complexity Analysis

In this section, we show that the worst case complexity of Algorithm **LC** is $O(n^2)$ and of Algorithm **HC** is $O(n^2 \log_2(L))$, where L is the latency and n is the number of nodes. The complexity of the Algorithm *Schedule* is dominated by the step where the nodes are ordered with respect to their freedom. Specifically, in this step, $A_j(i)$ nodes are chosen among $N_j(i)$ ready nodes in control cycle i. In the worst case, all the nodes are ready in each control cycle ($N_j(i) = n$) and the complexity for that control cycle is $A_j(i)n$. Note that $\sum_{i=1}^{L} A_j(i) = n$. Thus, in the worst case, the complexity of Algorithm *Schedule* is $O(n^2)$.

In Algorithm *Low Power*, the priority table can be searched in $\log_2(vf)$ passes, where v is the number of voltage levels and f is the number of resources. Since the number of voltage levels and number of resources are limited, the $\log_2(vf)$ term can be ignored for asymptotic analysis.

Algorithm **LC** consists of Algorithm *Schedule* and Algorithm *Low Power* with Algorithm 1 invoked and thus, has a complexity of $O(n^2)$. Algorithm **HC**, on the other hand, includes Algorithm *Low Power* with Algorithm 2 invoked. Since Algorithm 2 requires additional $\log_2 L$ passes to find the exact control cycle where disabling priority changes, the complexity of Algorithm **HC** is $O(n^2 \log_2 L)$.

D. Illustrative Example

Example 4: The resource constraint is $R_{M,5} = 1$, $R_{M,3.3} = 1$, $R_{M,2.4} = 1$, $R_{A,5} = 1$, $R_{A,3.3} = 1$ and $R_{A,2.4} = 1$ for the DFG in Fig. 1. The timing constraint is $L = 24$. Switching activities at all nodes are assumed to be 0.5. Algorithm 1 is invoked in case of timing violation.

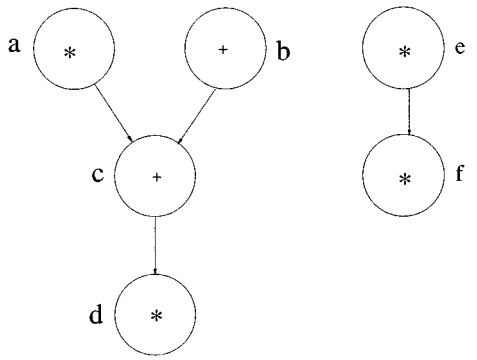

Fig. 1. Data-flow graph of *Example 4*.

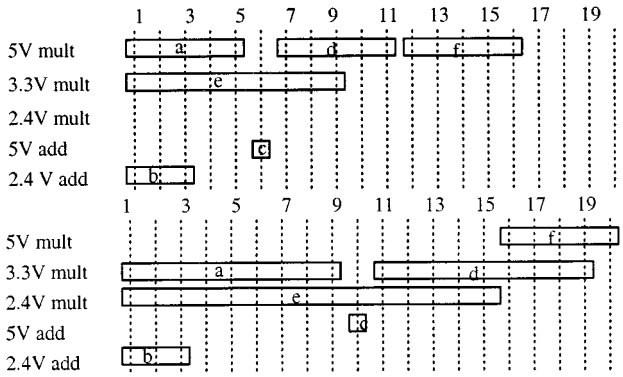

Fig. 2. Schedule for example 4. (a) After the first pass. (b) Final assignment.

First Pass: In control cycle $i = 1$, the ready sets are {a, e} and {b}. For nodes a and e, $C_M(1) = 9$, $A_{M,5}(1) = 1$, $A_{M,3.3}(1) = 1$. Node a is assigned to the 5 V multiplier since it has lower freedom, and node e is assigned to the 3.3 V multiplier [Step 3a of *Schedule*].

$C_A(1) = 1$, $A_{A,5}(1) = 1$, and node b is assigned to the 5 V adder. However, since the children of node b (node c) will be available at control cycle 6, we assign node b to the lower-voltage resource (2.4 V).

In control cycle $i = 6$, the ready set is {c}. $C_A(6) = 1$, $A_{A,5}(6) = 1$, and node c is assigned to the 5 V adder. *In control cycle* $i = 7$, the ready set is {d}. $C_M(7) = 5$, $A_{M,5}(7) = 1$, and node d is assigned to the 5 V multiplier. *In control cycle* $i = 10$, the ready set is {f}. $C_M(10) = 7$, $A_{M,5}(10) = 1$. But since 5 V multiplier is not available in control cycle 10, no scheduling is done. Node f is assigned to the 5 V multiplier in control cycle 12.

The schedule at the end of the first pass is shown in Fig. 2(a). $T_{\text{alg}} = T_{\text{low}} = 16$. Since $L > T_{\text{alg}}$, the multipliers operating at 5 V are disabled first (according to the priority table). $t_{\text{slack}} = L - T_{\text{alg}} = 24 - 16 = 8$.

Second Pass: In control cycle $i = 1$, the ready sets are {a, e} and {b}. For multiplication nodes a and e, $C_M(1) = 15$, $A_{M,3.3}(1) = 1$, $A_{M,2.4}(1) = 1$. So, node a is assigned to the 3.3 V multiplier since it has lower freedom and node e is assigned to the 2.4 V multiplier.

$C_A(1) = 1$, $A_{A,5}(1) = 1$, and node b should be assigned to the 5 V adder. However the children of node b (node c), will be available at control cycle 10. So we assign node b to the lower voltage resource (2.4 V) [Step 3b of *Schedule*].

In control cycle $i = 10$, the ready set is $\{c\}$. $C_A(10) = 1$, $A_{A,5}(10) = 1$, and node c is assigned to the 5 V adder. *In control cycle* $i = 11$, the ready set is $\{d\}$. $C_M(11) = 9$, $A_{M,3.3}(11) = 1$, $f_A = 19$ and node d is assigned to the 3.3 V multiplier. *In control cycle* $i = 16$, the ready set is $\{f\}$. $C_M(16) = 13$, $A_{M,2.4}(16) = 1$. Scheduling node f to 3.3 V violates its finish time ($f_F = 24 < i + C_M(16) = 29$). Thus, the algorithm shift backs to the previous schedule (no resources are disabled) and node f is assigned to the 5 V multiplier.

The final schedule is given in Fig. 2(b). Note that the computation finishes in 20 cycles, which is less than L.

E. Other Issues

1) Switching Activity Consideration: Switching activity of the resources depends on the switching probability of the input data and the circuit structure. The energy values of the resources and the priority table for disabling resources, assumed that the input switching activity is the same for all the resources ($\alpha = 0.5$). However, when the correlation of the multiplexed input data streams is high, the input switching activity is low and the energy consumption of the resource is low. Our analysis in [10] shows that if the switching activity of the resources vary by a factor of 2, then using average switching activity value results in a 3% error if the voltages are assigned according the minimum energy equation (derived using the Lagrange multiplier method). Thus, assuming that the switching activity of the resources varies by a factor less than 2, using $\alpha = 0.5$ introduces an error of at most 3% into our results.

2) Feasibility of the Algorithm: Let T_{crit} be the optimum-minimum-computation time (critical-path delay) under the given resource constraint. Finding a feasible solution for T_{crit} is equivalent to finding the optimal solution. The proposed algorithm is a list-based algorithm and does not guarantee an optimal solution. However, the proposed algorithm guarantees a feasible solution if $L = T_{\text{crit}} + \Delta$, where $\Delta = 1$ or 2 for the benchmarks that we have considered. The results are shown in Table III. Here $res = 1$ implies 1 multiplier and 1 adder operating at each of the following voltages: 5 V, 3.3 V, 2.4 V, 1.5 V, and $res = 2$ implies two multipliers and two adders operating at the same set of voltages. For this set of benchmarks, the average error is 3.9%. Thus, our list-based algorithm generates a feasible solution almost all the time.

TABLE III
FEASIBILITY ANALYSIS

benchmark	res.	L	T_{low}	Δ	% error
Diff-Eq	1	19	21	2	10.5
	2	15	15	0	0
AR-Lattice	1	47	48	1	2.1
	2	30	32	2	6.6
Elliptic Wave	1	36	37	1	2.7
	2	29	31	2	6.9
FFT4	1	9	9	0	0
	2	5	5	0	0
7th order FIR	1	21	21	0	0
	2	16	17	1	6.2

IV. RESULTS

In this section, we present the results obtained by running our algorithm on some high-level synthesis benchmarks (DIFFEQ, AR lattice, EW filter, FIR Filter, FFT4, and DCT). We present the results when actual energy consumption values in [4] are used. The switching activity of the nodes is assumed to be 0.5. The results for $res = 1$ (i.e., 1 multiplier and 1 adder operating at 5 V, 3.3 V, 2.4 V, and 1.5 V) have been tabulated in Table IV. In this table, E_5 is the energy dissipation corresponding to the supply voltage of 5 V. E_{alg} is the average energy obtained by our algorithm. Timing constraints are given for three different values: T_{low}, $1.5T_{\text{crit}}$, and $2T_{\text{crit}}$, where T_{crit} is the critical path delay. If $L = T_{\text{low}}$, then the average energy reduction is 17.5% compared to E_5 for Algorithm **HC**. Similarly, if $L = 1.5T_{\text{crit}}$, then the average energy reduction is 39% and for $L = 2T_{\text{crit}}$, the average energy reduction is 58.5% for Algorithm **HC**.

Table IV also demonstrates how the level-shifter-power consumption varies with the latency. If the given latency is tight, the majority of the nodes are assigned to the high-voltage resources to satisfy the timing constraint and consequently, the number of level shifters is low. When the given latency is high, the number of nodes assigned to the lower-voltage resources increases and the number of level shifters increases. The ratio of level-shifter energy to the total energy is 10.1% when the latency is $1.5T_{\text{crit}}$ and 11.4% when the latency is $2T_{\text{crit}}$ for Algorithm HC.

Fig. 3 graphically illustrates the energy reduction when Algorithm **LC** is used on the benchmarks for the case when $res = 1$ and when $res = 2$. Note, that since the values of T_{crit} and T_{low} are different for the two cases, the energy reductions for the two cases should not be directly compared.

Fig. 4 plots the reduction in energy when the latency varies from T_{low} to $2T_{\text{crit}}$ for the EW filter in Fig. 4(a), and the average case (the average of DIFFEQ, AR lattice, EW filter, FIR filter, FFT4 and DCT reductions) in Fig. 4(b). Here E_{HC} is the ratio of the energy of the assignment using Algorithm **HC** to the case when all the resources are assigned to 5 V, and E_{LC} is the ratio of the assignment using Algorithm **LC** to the case when all the resources are assigned to 5 V. We assume that $res = 1$ and that the latency is varied from T_{low} to $2T_{\text{crit}}$, where T_{low} is 3.9% higher than the critical path length (see Table III). The changes in the energy reduction for the EW filter graph [Fig. 4(a)] occur in steps. This is very typical of all the benchmarks that occur because increasing the latency can enable an extra addition or multiplication node to be scheduled to a lower-voltage resource. Similar step changes are not visible in Fig. 4(b) since this is an average energy-reduction plot, where the step changes have been smoothened.

From the plots, we see that the energy reduction increases with increase in latency. This is to be expected since an increase

TABLE IV
ENERGY REDUCTION FOR THE SET OF BENCHMARKS WHEN $res = 1$

Example	Latency	Algorithm LC			Algorithm HC		
		E_{alg} (pJ)	% redn	$\frac{E_{LS}}{E_{alg}}$%	E_{alg} (pJ)	% redn	$\frac{E_{LS}}{E_{alg}}$%
Diffeq	T_{low}	12112	22.3	1.9	12112	22.3	1.9
$E_5 = 15590pJ$	$1.5T_{crit}$	9275	40.5	2.7	7326	53	4.1
	$2T_{crit}$	5771	62.9	2.7	5743	63.1	2.7
AR-Lattice	T_{low}	31045	25	3	31016	25.1	2.9
$E_5 = 41416pJ$	$1.5T_{crit}$	23120	44.1	4.4	23120	44.1	4.4
	$2T_{crit}$	16073	61.2	8.6	15502	62.5	8.9
Elliptic Wave	T_{low}	20644	10.5	5.3	20644	10.5	5.3
$E_5 = 23068pJ$	$1.5T_{crit}$	12937	43.9	13.6	12865	44.2	11.4
	$2T_{crit}$	10946	52.5	15.8	10946	52.5	15.8
FFT4	T_{low}	1670	11.5	19.7	1670	11.5	19.7
$E_5 = 1888pJ$	$1.5T_{crit}$	1662	11.9	30	1662	11.9	30
	$2T_{crit}$	842	55.4	12.6	781	58.6	26.3
FIR	T_{low}	13593	25.3	2.5	13593	25.3	2.5
$E_5 = 18208pJ$	$1.5T_{crit}$	13445	26.1	1.1	12164	33.2	4
	$2T_{crit}$	8441	36.3	1.7	6301	65.4	2.7
DCT	T_{low}	12866	19.3	11.5	12551	21.2	10.5
$E_5 = 15942pJ$	$1.5T_{crit}$	11456	28.1	15.2	11085	30.4	11.9
	$2T_{crit}$	8003	50	22	7833	50.8	18.1
Average	T_{low}		18.9	7.3		19.3	7.1
	$1.5T_{crit}$		32.4	11.1		36.1	10.9
	$2T_{crit}$		51.3	10.4		58.8	12.6

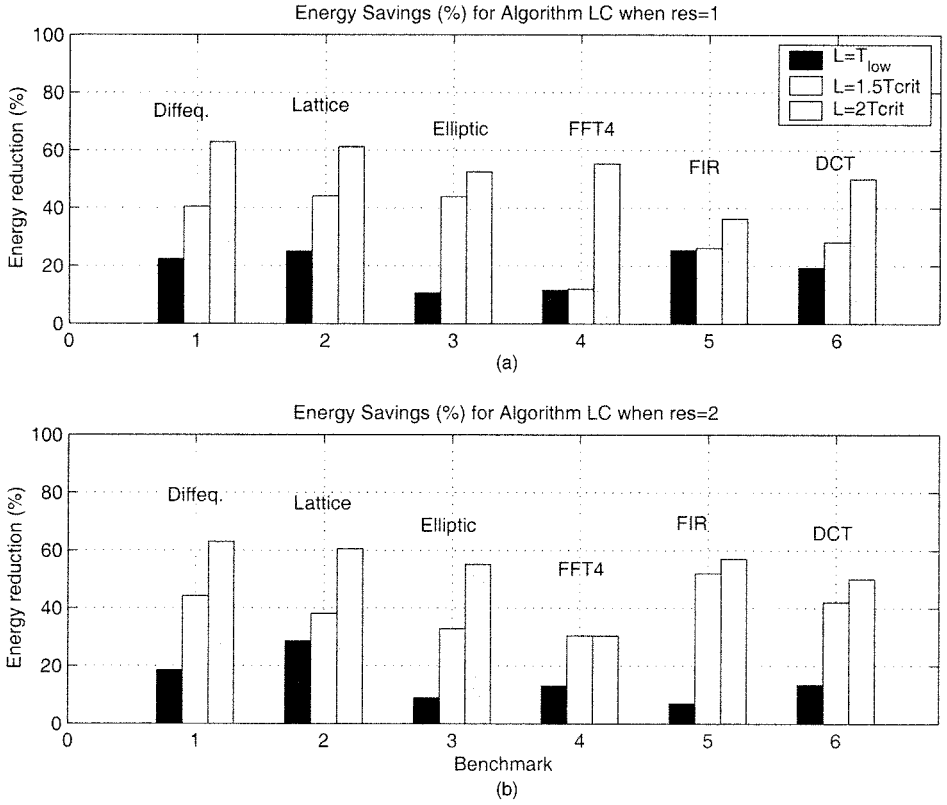

Fig. 3. Energy reduction for the set of benchmarks. (a) $res = 1$. (b) $res = 2$.

in latency facilitates more nodes being assigned to lower voltages. Second, the energy reduction occurs in steps. This is because a reduction occurs only when an additional node can be assigned to a lower-voltage resource as a result of the increase in latency. So, if the increase in the delay of a node (as a result of the lower-voltage assignment) is less than the increase in latency, there is no energy reduction. Third, for the same latency, the energy reduction obtained using Algorithm **HC**, is larger compared to Algorithm **LC**. Thus, the use of a more complex algorithm results in higher-energy savings. Fourth, for small

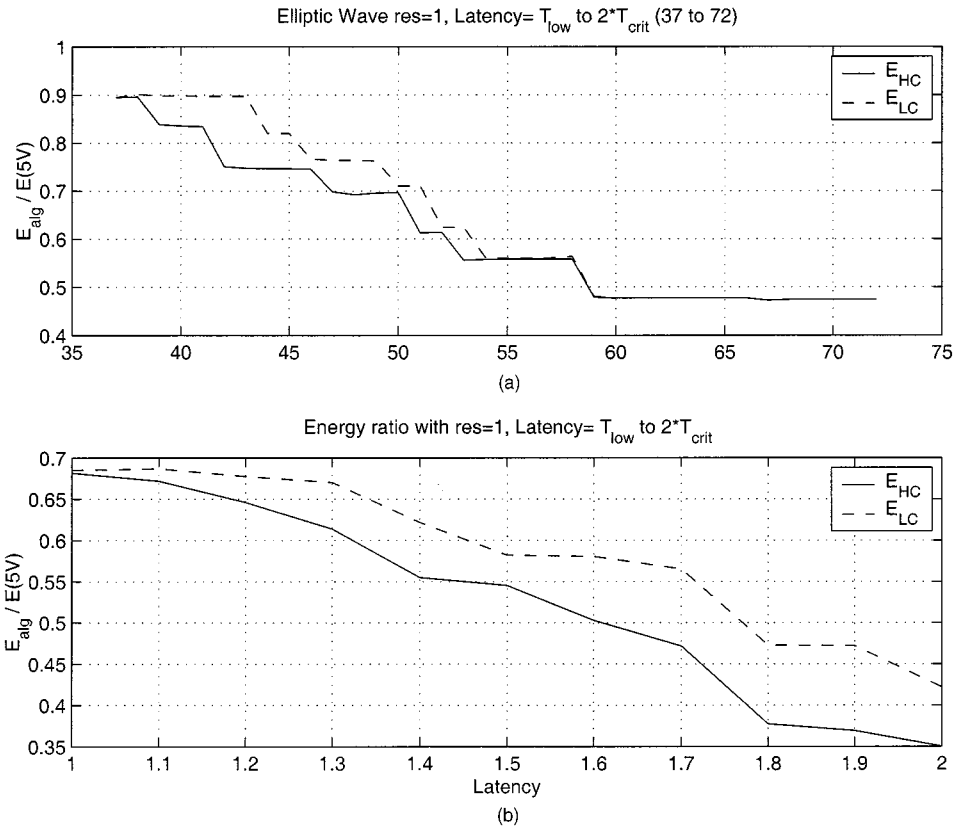

Fig. 4. Energy reduction with $res = 1$. (a) Elliptic wave filter. (b) Average over all benchmarks.

values of L, where $L = T_{\text{low}}$, E_{HC} is very close to E_{LC}. This is expected too, since when the latency is tight, most of the nodes are assigned to high-voltage resources.

V. Conclusion

In this paper, we present a new scheduling scheme under resource *and* latency constraint that minimizes power/energy consumption for the case when the resources operate at multiple voltages. The proposed scheme minimizes the power/energy consumption by distributing the slack among the nodes according to the condition derived using the Lagrange multiplier method. The scheme is implemented using an iterative algorithm, where in each iteration, increasing number of resources with high-energy-delay ratio are disabled and the nodes scheduled using a list-based algorithm. We propose two algorithms: 1) a simpler $O(n^2)$ algorithm and 2) a more complex $O(n^2 \log L)$ algorithm, where L is the latency and n is the number of nodes. The average reduction obtained by the more complex algorithm is 17.5% when the latency constraint is tight and is 39% when the latency constraint is 1.5 times the critical-path delay. The results obtained by the simpler algorithm are on the average 9% less compared to those obtained by the more complex algorithm. This is the expected tradeoff between the complexity of the algorithm and power/energy savings.

Acknowledgment

The authors would like to thank M. Erdem of the Department of Mathematics, Arizona State University, Tempe, AZ, for his help in the Lagrange formulation.

References

[1] A. P. Chandrakasan and R. W. Brodersen, "Minimizing power consumption in digital CMOS circuits," *Proc. IEEE*, pp. 498–523, Apr. 1995.
[2] K. Usami and M. Horowitz, "Clustered voltage scaling technique for low power," in *Proc. Int. Symp. Low Power Electronics Design*, Dana pt., CA, 1995, pp. 3–8.
[3] S. Raje and M. Sarrafzadeh, "Scheduling with multiple voltages," *Integr. VLSI J.*, pp. 37–60, Oct. 1997.
[4] J.-M. Chang and M. Pedram, "Energy minimization using multiple supply voltages," *IEEE Trans. VLSI Syst.*, pp. 436–443, Dec. 1997.
[5] W.-T. Shiue and C. Chakrabarti, "Low power scheduling with resources operating at multiple voltages," *IEEE Trans. Circuits Syst. II*, vol. 47, pp. 536–543, June 2000.
[6] M. C. Johnson and K. Roy, "Datapath scheduling with multiple supply voltages and level converters," *ACM Trans. Design Automation Electronic Syst.*, pp. 227–248, July 1997.
[7] Y.-R. Lin, C.-T. Hwang, and A. C.-H. Wu, "Scheduling techniques for variable voltage low power design," *ACM Trans. Design Automation Electronic Syst.*, pp. 81–97, Apr. 1997.
[8] M. Takahashi *et al.*, "A 60mW MPEG-4 video codec using clustered voltage scaling with variable supply voltage scheme," *IEEE J. Solid-State Circuits*, vol. 33, pp. 1772–1780, Nov. 1998.
[9] G. De Micheli, *Synthesis and Optimization of Digital Circuits*. New York: McGraw-Hill, 1994.
[10] A. Manzak and C. Chakrabarti, "Variable voltage task scheduling algorithms for minimizing energy," in *Proc. Int. Symp. Low Power Electronics and Design*, Huntington Beach, CA, Aug. 2001, pp. 279–283.

Ali Manzak received the B.S. degree in electronics and communication engineering from the Istanbul Technical University, Turkey, the M.S. degree in electrical engineering from the University of Colorado at Boulder, and the Ph.D. degree in electrical engineering from Arizona State University, Tempe, in 1991, 1996, and 2001, respectively.

Currently, he is with Lattice Semiconductor Corporation, San Jose, CA.

Chaitali Chakrabarti (S'86–M'89) received the B.Tech. degree in electronics and electrical communications engineering from the Indian Institute of Technology, Kharagpur, India, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland at College Park, in 1984, 1986, and 1990, respectively.

Since August 1990, she has been with the Department of Electrical Engineering, Arizona State University, Tempe, where she is currently an Associate Professor. Her research interests include low power systems design, memory optimization, high level synthesis and compilations, and VLSI architectures and algorithms for signal processing, image processing and communications.

Dr. Chakrabarti is a member of the Center of Low Power Electronics and the Telecommunications Research Center. She received the Research Initiation Award from the National Science Foundation in 1993, a Best Teacher Award from the College of Engineering and Applied Sciences, ASU, in 1994, and the Outstanding Educator Award from the IEEE Phoenix section in 2001. She has served on the program committees of ICASSP, ISCAS, SIPS, ISLPED, and DAC. She is currently an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the *Journal of VLSI Processing Systems*.