Delay Optimal Partitioning Targeting Low Power VLSI Circuits

Hirendu Vaishnav
Synopsys Inc.
700, E. Middlefield Road
Mountain View, CA 94043

Massoud Pedram
University of Southern California
Department of Electrical Engineering - Systems
EEB 206, University Park
Los Angeles, CA 90089-2562

Contents

List of Figures

List of Tables

Abstract

In this paper, a delay optimal clustering/partitioning algorithm for minimizing the power dissipation of a circuit is proposed. Traditional approaches for delay optimal partitioning are based on Lawler's clustering algorithm that makes no attempt to explore alternative partitioning solutions that have the same delay but better power implementations. Our algorithm provides a formal mechanism which implicitly enumerates alternate partitionings and selects a partitioning that has the same delay but less power dissipation. For tree circuits, the proposed algorithm produces delay and power optimal partitioning whereas for non-tree circuits it produces delay optimal partitioning with significantly improved power dissipation.

1 Introduction

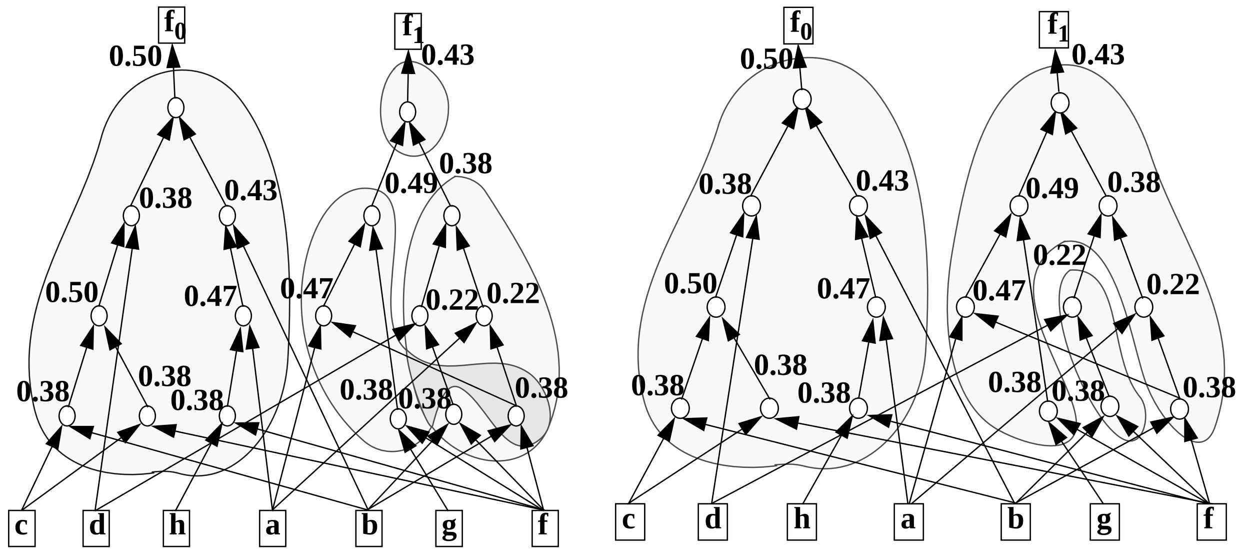
In recent years, the focus of portable devices has shifted from low throughput devices (e.g., watches, calculators) to high performance devices like notebook computers, PDAs, cellular phones, etc.. Minimizing power is the primary concern for these battery-powered products as for such products longer battery life translates to extended use and better marketability. Other low power applications include solar powered projects like communication satellites and applications emerging in the field of biomedical engineering. With the convergence of telecommunications, computers, consumer electronics, and biomedical technologies, the number of such low power and high performance applications is expected to grow rapidly. Furthermore, with the advent of deep submicron devices, excessive power consumption is becoming the limiting factor in integrating more transistors on a single chip or a multi-chip module. Exploring the trade offs between power, performance and area during synthesis and physical design is thus demanding more attention.

For low power physical design, a performance-driven placement tool [16], a floorplanner [3], and a performance-driven wire/driver sizing technique [5] have been proposed. This paper presents a low power delay optimal circuit partitioning algorithm.

Circuit partitioning is a technique to divide large circuits into smaller physical or logical subcomponents. Partitioning is necessary during synthesis and/or layout due to either of the following reasons: physical limitations on the number of transistors a chip or module can accommodate (e.g., to implement a large design on multiple chip modules or on fixed size PLA/FPGAs), for performing circuit restructuring during synthesis, for reducing complexity of synthesis or layout procedures by reducing the problem size, etc.. In most of these applications, a net that is external to a part drives a significantly larger load than an internal net. This implies that such inter-part nets contribute significantly to the circuit delay. An improper assignment of gates to parts can thus seriously degrade the performance of the circuit. Specifically, if tightly connected cells are put on different parts, paths may cross a partition boundary many times resulting in significant degradation of performance. Thus, it is important to partition a circuit using a performance-driven partitioning technique.

Initial work on performance-driven circuit partitioning was presented by Lawler et al. in [8]. His approach was a bottom-up approach which is usually referred to as "clustering" in contrast to a top-down approach which is referred to as "partitioning". Given a constraint M on the size of parts, Lawler's algorithm produces a delay optimal partitioning of the circuit by labeling gates based on the capacity of partial clusters at the input of a gate during a postorder traversal of the circuit. The partitioning produced by Lawler's algorithm is delay optimal under the assumption that internal delays within a cluster are zero and the external delay from one cluster to the other is one.

The main disadvantage of clustering algorithms based on Lawler's algorithm is that given a cluster size M, it arbitrarily selects one delay optimal clustering without providing the flexibility of exploring alternative delay optimal clustering solutions. Indeed, a large number of clustering solutions may exist with the same delay but different amount of power dissipation. For example, consider the benchmark circuit *con1* in Figure 1 where the values at the output of each gate corresponds to the switching activity as calculated by the symbolic simulation mechanism [7]. Square boxes in the figure correspond to the circuit inputs and outputs. Each circuit input is assumed to have a switching activity of 0.5. Figure 1(a) shows the delay optimal clustering solution generated by the Lawler's algorithm. Figure 1(b) corresponds to the power optimal clustering



Visible Power : 0.5 + 0.43 + 0.49 + 0.38 = 1.8
Total Power : 6.77 + 0.38 = 7.15
Gate replication ratio = 18/17 = 1.059
(a) Lawler's Algorithm

Visible Power: 0.5 + 0.43 + 0.22 = 1.15
Total Power : 6.77
Gate replication ratio = 17/17 = 1.0
(b) Our Algorithm

Figure 1: Two delay optimal clusterings under size constraint 8 for circuit *con1*.

solution generated by our algorithm that also has optimal depth. By selecting an alternative solution, the visible power dissipation of *con1* is reduced by 36% whereas the total power dissipation is reduced by about 5%. Hence, if the objective is to minimize the power dissipation during circuit clustering, it is necessary to explore alternative clustering solutions.

We propose a systematic approach for delay optimal clustering that implicitly enumerates all clustering solutions for a tree and selects a power optimal clustering from all delay optimal clustering solutions. The main idea is to maintain only the power optimal clustering solution for different delay values at every gate. This is achieved by enumerating all cluster patterns of size M at every gate starting from the primary inputs in a postorder fashion and selecting the power optimal cluster pattern for each delay value[1]. It is shown here that the problem of enumeration of all cluster patterns of size M at a gate i is related to the problem of enumerating all alphabetic trees of size m rooted at i where $m = \lceil \frac{M}{w_{min}} \rceil$ and w_{min} is the size of the smallest gate. This approach is similar in concept to the approach proposed for technology mapping in [4] with two significant differences: 1) technology mapping works within the context of available library patterns whereas in this case, all cluster patterns need to be enumerated thereby requiring an algorithm to efficiently enumerate all cluster patterns; 2) In this case, cluster enumeration needs to be performed on non-binary trees whereas technology mapping procedures operate only on binary trees. When applied to tree structures, the algorithm proposed here produces delay and power optimal clustering whereas Lawler's algorithm produces only delay optimal clustering.

Apart from allowing a simultaneous optimization of power dissipation of the circuit, our al-

[1]An approach to enumerate different cluster patterns in the context of FPGA mapping was proposed by Murgai et al. [10]. However, they are enumerating clusters under a constraint on the number of inputs to a cluster whereas we enumerate clusters under a constraint on the size of the cluster.

gorithm has the following advantages as compared to traditional delay optimal partitioning algorithms.

- It produces a delay optimal clustering under a generalized load independent delay model.
- The proposed algorithm can be used to minimize objective functions other than power dissipation.
- The proposed algorithm produces a set of non-inferior power-delay solutions, allowing for an informed trade-off between delay and power dissipation of the circuit.
- If required, minimum cluster size constraints can be easily incorporated in the proposed algorithm, thereby allowing a partitioning solution with approximately equal size partitions.
- The method can be easily adapted to perform partitioning in a hierarchical manner thereby allowing larger partition sizes at little gate replication penalty.

The rest of the paper is organized as follows. In the next section, we present the power model used for power estimation/minimization in this paper. Section 1.2 presents a brief description of Lawler's algorithm and describes other work in performance-driven clustering. In Section 2, our approach for power optimal clustering for tree structures is presented. Generalization of this approach for DAGs and the consequences are described in Section 3. Section 5 presents experimental results and some empirical studies that characterize the impact of maximum cluster size M on the algorithm. Concluding remarks are presented in Section 6.

1.1 The Power Model

Power consumption in CMOS circuits is caused by three sources: the charging and discharging of capacitive loads during output switchings, the short circuit current which flows during output transitions and leakage current. The last two sources can be made small with proper device and circuit design techniques. CAD tools have thus concentrated on the dynamic power consumption.

The average dynamic power consumed by a CMOS logic gate i (if the gate is a part of a synchronous digital system controlled by a global clock) is given by

$$P_i^{average} = 0.5 \frac{V_{dd}^2}{T_{cycle}} C_i^{load} N_i \tag{1}$$

where C_i^{load} is the load capacitance, V_{dd} is the supply voltage, T_{cycle} is the global clock period, and N_i is the number of gate output transitions per clock cycle (i.e., *switching activity* of gate i).

The fanout load C^{load} in equation (1) consists of two components: the gate load C^{gate} which accounts for the input capacitances of fanout gates and the drain to ground capacitance of the driver, and the wire load C^{wire} which accounts for the load due to the interconnection tree formed between the driver and its fanout gates. Logic synthesis for low-power attempts to minimize $\sum_i C_i^{gate} N_i$, whereas physical design for low-power should minimize $\sum_i C_i^{wire} N_i$.

Estimating Switching Activities

The number of transitions N a gate makes during a clock cycle is a complex function of its global logic function, delays in the circuit, and the input sequences applied. Given a set of input vector

Figure 2: An illu

A Simplified Power Model for Circuit Partitioning

The fanout load C_i^{load} in equation (1) consists of two components: the basic net load C_i^{basic} which accounts for the load capacitances seen by gates in absence of any partitioning, and the extra load C_i^{extra} which accounts for the additional load capacitance due to the external connections of the net. Note that for a net with no external connections $C_i^{extra} = 0$.

Then, the total power dissipation of the circuit $\mathcal{G}$ is given by

$$P_{\mathcal{G}} = 0.5\frac{V_{dd}^2}{T_{cycle}} \sum_{i \in \mathcal{G}} (C_i^{basic} + C_i^{extra}) N_i \tag{2}$$

When a circuit partitioning corresponds to a physical partitioning, the additional load driven by gate i driving an external net, namely, C_i^{extra}, is one to three orders of magnitude larger than that of an internal net (i.e., $C_i^{basic} << C_i^{extra}$). In this case, the power model given in (2) can be simplified further as follows. First, it can be assumed that the power dissipation contribution that can be attributed to variations of C_i^{basic} under different partitioning solutions is negligible, i.e., $C_i^{basic} = 0 \;\; \forall i \in \mathcal{G}$. Furthermore, considering that the fixed overhead capacitance for an external net is dominant within C_i^{extra}, it can be assumed that C_i^{extra} is identical for each net. Under these assumptions, the objective function to be minimized during partitioning is given by

$$O_{\mathcal{G}} = \sum_{i \in \mathcal{G}_v} N_i \tag{3}$$

where $\mathcal{G}_v$ corresponds to the set of gates that are visible, i.e., the set of gates that drive a load external to the partition. As described in Section 5, the algorithm proposed here can minimize power dissipation under both the basic power dissipation model given in equation (2) and the simplified power dissipation model given in equation (3) (i.e., when $C_i^{basic} << C_i^{extra}$).

Note that depending on the application, i.e., whether targeting a physical partitioning or a logical partitioning, the proposed algorithm can be used to optimize either equation (3) or equation (2).

1.2 Background

Given a directed acyclic combinatorial circuit $\mathcal{G}$, the set of inputs of a gate $i \in \mathcal{G}$ are denoted by $\mathcal{I}_i$. The weight (i.e., area) of gate i is denoted by w_i and the label of the gate is denoted by l_i. Let $W_i(l_i)$ denote the total weight of all the gates with label l_i in the transitive fanin of gate i.

A **cluster** is defined as a connected rooted DAG where the vertices correspond to the gates belonging to the cluster, edges correspond to the connections between the gates and the root of the cluster is the gate reachable by all the gates in the cluster (i.e., all the gates in the cluster are in the transitive fanin of the root of the cluster).

Let the constraint on the size of the cluster (i.e., sum of the weights of gates belonging to that cluster) be M. Then, Lawler's algorithm [8] produces a depth-optimal partitioning as follows:

1. Gates with in-degree 0 (i.e., circuit inputs) are given the label 0.

2. Find any unlabeled gate i such that all gates belonging to $\mathcal{I}_i$ have been labeled. Let k be the largest label applied to any of these gates. If $w_i + W_i(k) \leq M$ then $l_i = k$. Otherwise, $l_i = k + 1$.

3. After the gates are labeled, locate all gates that have a label distinct from all of its outputs. Such gates correspond to the roots of a cluster. The root gate and all the gates in the transitive fanin of the root with the same label constitute a cluster.

Figure 1(a) illustrates the partitioning obtained by applying Lawler's algorithm to the benchmark circuit *con1* with a size restriction 8.

As a consequence of the labeling scheme, when a gate has multiple fanouts, it is likely that the gate is replicated in clusters containing each of its fanouts. Thus, the delay optimality is achieved at the cost of an increase in the gate area. This gate area can be recovered by some post-clustering operation during which clusters are merged into larger clusters such that the size constraint is not violated and the gate replication is reduced. Lawler indicates that the problem of merging to minimize the gate replication is similar to that of rectangle covering which is shown to be NP-hard. Lawler also proposes a heuristic to minimize gate replication. Murgai et al. [9] have proposed some heuristics to reduce the number of clusters and to reduce the gate replication. Touati [13] has also developed a gate area recovery mechanism based on a relabeling scheme.

Murgai et al. [9] have modified Lawler's algorithm to allow a generalized load independent delay model, i.e., a delay model in which each gate i has delay of $\delta(i)$ time units and each inter-cluster net has delay of d time units[2]. However, they show that the modified Lawler's algorithm they propose is delay optimal only under certain special conditions. Recently, a clustering algorithm that produces delay optimal clustering under a load independent delay model has been proposed in [12]. Both these algorithms still rely on the basic framework of Lawler's algorithm and hence, have no flexibility to explore alternative clustering solutions.

2 Power Optimal Partitioning for Trees

2.1 Definitions and Notations

A **Cluster Pattern** at a gate i, denoted P_i, is defined as a cluster of size less than or equal to M with gate i as the root of the cluster. The concept of cluster patterns is illustrated in Figure 3. Note that each cluster pattern has a set of leaf nodes associated with it. The set of leaf nodes associated with a cluster pattern P_i is referred to as its **leaf set** $\mathcal{L}_i$.

A **clustering solution** at a gate i is characterized by a **PD-point** (power-delay point) which is a 3-tuple $\{a_i, p_i, \mathcal{LP}_i\}$ where a_i gives the delay value (i.e., arrival time) associated with the PD-point, p_i gives the corresponding power cost of the clustered sub-circuit rooted at gate i and $\mathcal{LP}_i$ denotes the corresponding set of PD-points at the leaf nodes of the cluster pattern. It should be noted that a clustering solution at gate i determines the clustering solution of the entire sub-circuit rooted at gate i.

Given $\mathcal{LP}_i$, the power cost p_i and the arrival time a_i are calculated as follows.

$$p_i = N_i + \sum_{l \in \mathcal{LP}_i} p_l \tag{4}$$

$$a_i = max_{l \in \mathcal{LP}_i}\{a_l + d_l\} + d \tag{5}$$

[2]To emphasize that in spite of being more general than unit delay model, this model still assumes that gate delays are independent of the load driven by gates, we refer to this delay model as a "generalized load independent delay model" in contrast to the term "generalized delay model" used in [9].

2.2 An Enumerative Clustering Algorithm

A power optimal clustering can be obtained by exhaustively enumerating all possible clustering solutions of the circuit. However, such an approach is computationally infeasible. Instead, we propose a dynamic programming approach that generates power optimal clustering solutions at a gate given the power optimal clustering solution at each gate in its transitive fanin.

In order to apply a dynamic programming approach to explore alternative clustering solutions and to identify the delay and power optimal clustering solution for the circuit, the clustering mechanism should satisfy the following two conditions: 1) It should provide the freedom to explore all cluster patterns at a gate; 2) For each pair of such patterns, it should be able to correctly characterize the relative delay and power cost of each pattern and select the best cluster pattern which guarantees delay and power optimality of the entire circuit.

Assuming that mechanisms that satisfy both of these conditions (i.e., mechanisms that enumerate and characterize cluster patterns at each gate) can be provided, the outline of our algorithm is as follows. POWER_CLUSTER visits each gate in a postorder (i.e., all inputs of a gate are visited before the gate is visited) and generates a power optimal clustering at each gate. In fact, the algorithm maintains a set of non-inferior power-delay solutions at each gate. The inherent benefit of maintaining such non-inferior solutions during a dynamic programming algorithm is that if the cost of the solution to a problem is monotone in the cost of the solutions to its subproblems, then these non-inferior solutions are sufficient to guarantee optimality of the main problem.

```
**Algorithm 1** POWER_CLUSTER ($\mathcal{G}$, $M_{min}$, $M_{max}$)
01  **for** each gate $i \in \mathcal{G}$ (in postorder) **do**
02    **for** $size = M_{min}$ to $M_{max}$
03      $\mathcal{L}$ = first_leaf_set($i$, children($i$), $size$)
04      **do**
05        **for** each set of arrival times $a_{\mathcal{L}}$ for $\mathcal{L}$
06          $a$ = output_arrival_time($\mathcal{L}, a_{\mathcal{L}}, i$)
07          $p$ = power_dissipation_after_merge($\mathcal{L}, a_{\mathcal{L}}$)
08          update_PD-set($a, p$)
09        $\mathcal{L}$ = next_leaf_set($i$, children($i$), $\mathcal{L}$, $size$)
10      **while** ($\mathcal{L} \neq NULL$)
11  $\mathcal{L}$ = set of primary outputs
12  **for** each set of arrival times $a_{\mathcal{L}}$ for $\mathcal{L}$
13    $a_o$ = max($a_{\mathcal{L}}$)
14    $p_o$ = power_dissipation_after_merge($\mathcal{L}, a_{\mathcal{L}}$)
15    update_PD-set($a_o, p_o$)
16  Select_a_non-inferior_solution
```

In the following, we describe the operation of POWER_CLUSTER in the context of tree circuits[3]. Modification of this algorithm to handle general DAGs will be presented in Section 3.

[3]The same analysis is also applicable to leaf-DAGs, i.e., circuits that have multiple fanouts only at the circuit inputs.

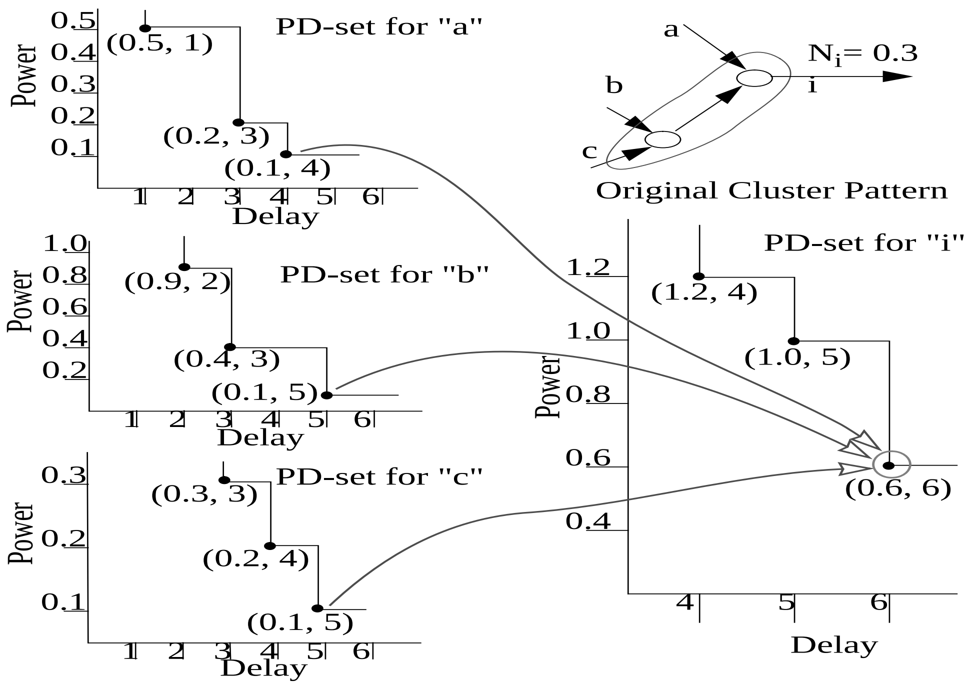


Figure 4: An illustration of PD-set.

The routines *first_leaf_set* and *next_leaf_set* on Line 3 and Line 9, respectively, identify each cluster pattern as will be described in Section 2.4. Once the leaf set $\mathcal{L}$ corresponding to a cluster pattern P_i is available, the sets of arrival times at gate i are determined. Since the worst size of $[A_{min}, A_{max}]$ is D, the maximum number of arrival time sets and the maximum number of execution of Lines 5-8 for each cluster pattern is D. Thus, maintaining a PD-set as opposed to maintaining a single delay optimal clustering at each gate increases the time complexity of the algorithm by a multiplicative factor D.

Given the PD-sets at each element in leaf set $\mathcal{L}$, Line 5 in POWER_CLUSTER enumerates all arrival time set $a_{\mathcal{L}}$ for the leaf set $\mathcal{L}$. For example, if the leaf set $\mathcal{L} = \{a, b, c\}$ and if

$$\begin{aligned}
\mathcal{S}_a &= \{\{1, 0.5, \mathcal{LP}_a{}^1\}, \{3, 0.2, \mathcal{LP}_a{}^2\}, \{4, 0.1, \mathcal{LP}_a{}^3\}\} \\
\mathcal{S}_b &= \{\{2, 0.7, \mathcal{LP}_b{}^1\}, \{3, 0.4, \mathcal{LP}_b{}^2\}, \{5, 0.1, \mathcal{LP}_b{}^3\}\} \\
\mathcal{S}_c &= \{\{3, 0.3, \mathcal{LP}_c{}^1\}, \{4, 0.2, \mathcal{LP}_c{}^2\}, \{5, 0.1, \mathcal{LP}_c{}^3\}\}
\end{aligned}$$

then $A_{min} = 3, A_{max} = 5$ and the arrival time sets are given by $\{3, 3, 3\}, \{4, 3, 4\}$ and $\{4, 5, 5\}$ where the first, second and the third elements correspond to the leaf nodes a, b and c respectively. The output PD-set corresponding to the above example is shown in Figure 4.

For generalized load independent delay model, the delay from the leaf node to the root is first added to the value of each PD-point of the leaf before identifying A_{min} and A_{max}. For example, if the delays from leaf nodes a, b and c to the root of the cluster are $3, 1$ and 2, respectively, then the modified PD-sets (denoted by $\mathcal{S}_a{}', \mathcal{S}_b{}'$ and $\mathcal{S}_c{}'$, respectively) are given by

$$\begin{aligned}
\mathcal{S}_a{}' &= \{\{4, 0.5, \mathcal{LP}_a{}^1\}, \{6, 0.2, \mathcal{LP}_a{}^2\}, \{7, 0.1, \mathcal{LP}_a{}^3\}\} \\
\mathcal{S}_b{}' &= \{\{3, 0.7, \mathcal{LP}_b{}^1\}, \{4, 0.4, \mathcal{LP}_b{}^2\}, \{6, 0.1, \mathcal{LP}_b{}^3\}\}
\end{aligned}$$

$$\mathcal{S}_c{}' \quad = \quad \{\{5, 0.3, \mathcal{LP}_c{}^1\}, \{6, 0.2, \mathcal{LP}_c{}^2\}, \{7, 0.1, \mathcal{LP}_c{}^3\}\}.$$

Apart from this, the rest of the algorithm is independent of the delay model used. For each such arrival time set, Line 6 of the algorithm calculates the arrival time at the output of the cluster. This is simply computed by adding d to the maximum value in the arrival time set. Thus, the overall time complexity of Line 6 is $O(1)$ for unit delay model and $O(m)$ for the generalized load independent model.

Line 7 computes the power cost corresponding to this clustering solution for gate i using equation (4). For example, for arrival time set $\{3, 3, 3\}$ (under unit delay model) the power cost of the clustering solutions for the transitive cone corresponding to each leaf is given by $0.2 + 0.4 + 0.3 = 0.9$. As illustrated in Figure 4, if the switching activity of gate i is 0.3 then p is calculated as $0.9 + 0.3 = 1.2$. Thus, at the end of Line 7, we have identified a PD-point for gate i with $a = 4$ (i.e., $3 + 1$) and $p = 1.2$. The time complexity of *power_dissipation_after_merge* is $O(m)$.

Line 8 of the algorithm adds this PD-point to the PD-set of gate i if it has lower power dissipation with respect to the faster solutions. Whenever a PD-point is introduced to a PD-set, all the slower solutions that have become inferior (i.e., have worse power cost) to the newly introduced PD-point are removed. Since the size of the PD-set is limited by D, the time complexity of *update_PD-set* is $O(D)$.

Lines 11-15 apply a similar technique to the leaf set corresponding to the primary outputs of the circuit. This identifies non-inferior PD-points for the entire circuit in the context of the slowest primary output. In other words, if the earliest arrival time at an output is 4, then it is not necessary to select a clustering solution with arrival time 2 at another output when a clustering solution with lower power dissipation but with arrival time 3 is available at that output. Finally, Line 16 of the algorithm allows one to select the solution based on the delay and/or power constraints. This determines the selection of PD-points corresponding to each primary output. Using these PD-points a clustering solution is generated by traversing the circuit top-down (i.e., from primary outputs to primary inputs). Specifically, given that PD-point j is chosen at a gate i, the clustering solution at the corresponding inputs is generated by recursively generating a clustering solution for each element in $\mathcal{LP}_i{}^j$.

The time complexity of POWER_CLUSTER is given by $|\mathcal{G}|^2 \cdot m \cdot O(D) \cdot (O(m) + O(D)) \cdot number\_of\_patterns = O(|\mathcal{G}|^2 \cdot (Dm^2 + mD^2) \cdot number\_of\_patterns)$.

2.3 Number of Distinct Cluster Patterns

POWER_CLUSTER crucially depends on effective and efficient enumeration of cluster patterns. In this section, a cluster enumeration algorithm is described and it is shown that the number of cluster patterns with size M is limited by the number of alphabetic trees with $m(T-1)+1$ leaf nodes. Specifically, it is shown that if m and T are bounded by a constant, the runtime of POWER_CLUSTER, though exponential in m and T, is polynomial is the circuit size $|\mathcal{G}|$. Before relating the number of cluster patterns (and hence the time complexity of POWER_CLUSTER) to the number of alphabetic trees, we define alphabetic trees and some relevant notations. Further details on alphabetic trees can be found in [15].

An **alphabetic tree** is a tree generated on an ordered set of leaf nodes such that internal edges do not cross each other. The number of alphabetic trees on n leaf nodes are denoted by $\Phi_n{}^{|t|}$ where each internal node has arity t (i.e., number of children). Let Φ_n denote the number of alphabetic trees when there is no arity restriction. Likewise, let Π_M denote the number of cluster patterns

For non-binary tree structures, the worst case m is given by $\lceil \frac{M}{w_{min}} \rceil$, which leads to the following.

Theorem 2.3

$$\Pi_M \leq \Phi_{m(T-1)+1} = O(6^{m(T-1)+1}) \tag{7}$$

where Φ_n denotes the number of alphabetic trees on n leaf nodes without any restriction on the arity of the nodes.

Proof Since $m = \lceil \frac{M}{w_{min}} \rceil$, any cluster pattern generated under size constraint M can not have more than m gates. If the arity of each gate is at most T, the maximum number of leaf node that an m node tree may have is $m(T-1)+1$. From Lemma 2.1, each cluster pattern corresponds to an alphabetic tree. Thus, in the worst case, the number of cluster patterns can never be more than $\Phi_{m(T-1)+1}$. It has been shown [17] (see [15] for further details) that $\Phi_{m(T-1)+1} = O(6^{m(T-1)+1})$. ∎

This implies that the number of cluster patterns that need to be enumerated is independent of the circuit size and is exponential only in m and T which are usually fixed to be a small number. However, in the case where clusters of larger size are required, by using a hierarchical clustering approach, the number of cluster patterns can still be kept low. This and other extensions are discussed in Section 5.

2.4 Cluster Pattern Enumeration

Since different cluster patterns of size M are enumerated based on m which is the maximum number of gates that can be contained in a cluster of size M, it is likely that an infeasible cluster (i.e., a cluster with the actual size greater than M) is returned by the procedures enumerating clusters. However, enumerating based on m guarantees that all feasible clusters are enumerated. In the actual implementation, infeasible clusters can be easily avoided by keeping track of current size of clusters during the cluster pattern enumeration algorithm. Thus, the value of m has significance only in terms of deriving an upper bound on the runtime of the proposed algorithm.

Given the cluster size m in terms of the number of gates in a cluster, two routines are proposed to enumerate all cluster patterns rooted at a gate. The first routine, namely, *first_leaf_set* returns the leaf set corresponding to the first cluster pattern at a gate with a given size. Given the current leaf set, routine *next_leaf_set* returns the next leaf set of the same size. The topological order amongst the cluster pattern is formally defined as follows.

Let P_i and P'_i correspond to two cluster patterns of the same size m at gate i. Let i_1 through i_I correspond to the inputs of i ordered from left to right. The part of cluster P_i rooted at the inputs of gate i is denoted by P_{i,i_1} through P_{i,i_I}, and let W_1 through W_I correspond to the size of each such partial cluster, respectively. We refer to the set of these sizes as $\mathcal{W}_{1,I}$. We define $\mathcal{W}_{j,I} < \mathcal{W}'_{j,I}$ iff:

- $W_j < W'_j$ or
- $\mathcal{W}_{j+1,I} < \mathcal{W}'_{j+1,I}$

Then, we define $P_i > P'_i$ iff:

- $\mathcal{W}_{1,I} < \mathcal{W}'_{1,I}$ or

- $\mathcal{W}_{1,I} = \mathcal{W}'_{1,I}$ and $P_{i,i_{j-1}} = P'_{i,i_{j-1}} \forall j < k$ and $P_{i,i_k} > P'_{i,i_k}$ for any $k \in [1, I]$.

This concept of cluster pattern ordering is illustrated in Figure 3 and 5. Specifically, Figure 3 shows the order between the clusters and Figure 5 illustrates the corresponding tree structures. Based on the above order definition, cluster patterns are enumerated starting from the first cluster pattern to the last cluster pattern by routine *first_leaf_set* and routine *next_leaf_set*. We provide efficient implementations of these routines as follows.

In these routines we use "{ }" as a set forming operator. Procedure "max_size(gate)" returns the number of gates in the transitive fanin which is computed in a preprocessing step. It should be noted that these routines only give a general outline of the actual routines. Certain special considerations, e.g., handling of DAG circuits, are not included in these routines.

```
Routine 1 first_leaf_set (gate, children, size)
begin
  if (size == 1) return { gate }
  else (size = size - 1)
  currentSet = NULL
  forEach child starting from the right most of children
    children = children - { child }
    childSize = min(max_size({ child }), size)
    grandChildren = get_children(child)
    currentSet = currentSet ∪ first_leaf_set (child, grandChildren, childSize)
    size = size - childSize
    if (size == 0)
      currentSet = currentSet ∪  children
      break;
  return currentSet
end
```

The routine *first_leaf_set* generates the first topological pattern by generating a tree which is as much right-biased as possible. This is achieved by including as many gates as possible starting from the branch that is farthest right and including as many gates as possible before including the sibling to the left of the right most branch.

```
Routine 2 next_leaf_set (gate, children, leafSet, size)
begin
  if ((size == 1)||(size == 0)) return NULL
  if (|children| == 1)
    gate = left_most(children)
    children = get_children(gate)
    return next_leaf_set(gate, children, leafSet, size - 1)
  leftMostChild = left_most(children)
  children = children - {leftMostChild}
  leftLeafSet = left_leaf_set(leafSet)
  rightLeafSet = leafSet - leftLeafSet
  newRightSet = rightLeafSet
  if (size(rightLeafSet) > 0)
    newRightSet = next_leaf_set(gate, children,
            rightLeafSet, size(rightLeafSet))
    if (newRightSet ≠ NULL) return leftLeafSet ∪ newRightSet
    newRightSet = first_leaf_set(gate, children, size(rightLeafSet))
  if (size(leftLeafSet) > 0)
    newLeftSet = next_leaf_set(gate, {leftMostChild},
            leftLeafSet, size(leftLeafSet))
    if (newLeftSet ≠ NULL) return newLeftSet ∪ newRightSet
  if (size(rightLeafSet) == 0) return NULL
  size(rightLeafSet) = size(rightLeafSet) −1
  size(leftLeafSet) = size(leftLeafSet) +1
  if (max_size(leftMostChild) ¡ size(leftLeafSet)) return NULL
  if (size(rightLeafSet) > 0)
    newRightSet = first_leaf_set(gate, children, size(rightLeafSet))
  else newRightSet = NULL
  newLeftSet = first_leaf_set(gate, {leftMostChild}, size(leftLeafSet))
  return newLeftSet ∪ newRightSet
end
```

The routine *next_leaf_set* identifies leaf set associated with the next topological cluster pattern by recursively traversing to the right most gate j whose part of the original cluster P_i, namely $P_{i,j}$ can be restructured by removing a gate from one branch to add another gate to a branch on the left of it. In essence, this routine incrementally turns a right-biased clustering pattern to a left-biased clustering pattern.

Next, we show that this enumeration along with proper characterization of power is sufficient for power optimality of the clustering solution.

2.5 Optimality of POWER_CLUSTER

Theorem 2.4 *For tree structures,* POWER_CLUSTER *generates delay and power optimal clustering solution under the generalized load independent delay model.*

Proof This is shown by induction on the original depth (i.e., depth before clustering). The algorithm generates a power and delay optimal solution at original depth 1 as the only solution at depth 1 is to cluster each gate individually. Assume that all the solutions for original depths 1 to k are delay and power optimal. Now, to prove the theorem it needs to be shown that the solution generated by POWER_CLUSTER for a gate i at original circuit depth $k+1$ is delay and power optimal.

Delay Optimality: Since the algorithm evaluates every feasible cluster pattern under the size constraint M, it identifies every cluster pattern that leads to the optimal arrival time value at gate i.

Power Optimality: To guarantee delay and power optimality, POWER_CLUSTER chooses the minimal power solution from the solutions that result in optimal delay at gate i. Thus, to show the power and delay optimality of the algorithm, need to show that: 1) no other solution with less power for the same delay exists; and 2) a selection based on the power cost of each cluster pattern minimizes the power contribution of the clustering solution at gate i to the power dissipation of the entire circuit.

The first is true based on the assumption that clustering solution at each gate in the transitive fanin cone is delay and power optimal and the fact that all feasible cluster patterns at gate i are enumerated. 2) is true due to the tree structure of the circuits. By selecting power optimal cluster pattern from all cluster patterns at gate i, POWER_CLUSTER guarantees the power optimality of the sub-circuit rooted at gate i. However, for tree structures, power cost of the sub-circuit rooted at i also corresponds to the power contribution of the sub-circuit to the entire circuit.

Thus, POWER_CLUSTER generates delay and power optimal circuits at gates with original depth $k+1$. ∎

In should be noted that if area cost is measured in terms of minimal gate duplication, POWER_CLUSTER also generates area optimal circuits as any clustering on tree structures causes no gate duplication.

Taking into account the maximum number of cluster patterns the final time complexity of POWER_CLUSTER is given by $O(|\mathcal{G}|^2 \cdot (Dm^2 + mD^2) \cdot number\_of\_patterns) = O(|\mathcal{G}|^2 \cdot (Dm^2 + mD^2) \cdot 6^{m(T-1)+1})$. This complexity may appear prohibitive but it should be noted that the runtime is exponential only in terms of the cluster size and not in terms of the circuit size. By keeping the size of each cluster small relative to the size of individual gate sizes, the runtime penalty can be kept to a minimum. In cases where clusters of larger size are required, the clustering algorithm can be applied in a hierarchical fashion to achieve the larger size as described in Section 5

3 Low Power Partitioning for DAGs

To be able to handle general combinational circuits (i.e., DAGs) correctly, a few modifications must be made to POWER_CLUSTER.

Cluster Pattern Enumeration: Consider a gate i which fans out to gates x and y that are input to another gate z. Due to the re-convergent fanout at gate z, when enumerating cluster patterns at z, gate i will be encountered in two subtrees of z, namely, the subtree rooted at x

gates x and y are in the transitive fanout of gate i, it may not be possible to make correct decisions for power and area optimality at gates x and y. Specifically, since the sub-circuit rooted at gate i is shared by the sub-circuits rooted at x and y, it is likely that a sub-optimal solution in terms of individual power costs of x and y may be optimal when the combined power cost of x and y is considered. This implies that, to guarantee optimality, all clustering solutions at a gate (not only the non-inferior solutions) must be maintained, which grows exponentially with the circuit size. However, the delay optimality of POWER_CLUSTER under a load independent delay model remains valid.

Theorem 3.1 POWER_CLUSTER *generates a delay optimal clustering under a generalized load independent delay model.*

Proof The proof of the theorem follows from the observation that the only impact of generalization to DAGs from trees on the delay calculation is due to the effect of multiple fanout nodes. However, under a generalized load independent delay model, the delay is independent of fanout characteristics of gates. Hence, the claim for delay optimality of POWER_CLUSTER remains valid for general DAGs. ■

4 Post-Clustering Area Recovery

The gate replication introduced by POWER_CLUSTER can be recovered by employing area recovery mechanism that reduce gate replication at no loss in power dissipation. However, even greater reduction in gate replication can be obtained by allowing some loss in power dissipation. In this section, three techniques for area recovery are described.

4.1 Area Recovery by Forcing a Single Clustering Solution

As shown in Figure 7(a), due to multiple outputs with different criticality or due to reconvergent fanouts, it is likely that in the final solution more than one clustering solution is required at a gate. In such a situation, both area and power can be recovered without any loss in delay by always selecting the fastest clustering solution required at that gate as shown in Figure 7(b). In POWER_CLUSTER, the concept of membership set automatically ensures this type of area recovery by selecting only the best clustering solution at each gate.

4.2 Area Recovery by Relabeling Based on Unique Label Assignment

An important characteristic of clustering algorithms based on Lawler's clustering algorithm is that each gate in the circuit has exactly one label. With POWER_CLUSTER, however, a gate can have multiple labels as shown in Figure 8(a). By forcing each gate to have only one label, the area of the clustered circuit can never get worse. However, the exact impact on the power dissipation needs to be analyzed. For each case where a gate has more than one label, by forcing the fastest label on that gate, a new cluster is introduced as shown in Figure 8(b). Since a new visible gate is introduced, the power dissipation of the circuit is likely to increase. However, by giving a unique label some power may also be saved in the following two cases:

- Since different clustering solutions corresponding to other labels for the root of new cluster are not required, the overall power dissipation may decrease.
- When C^{basic} can not be ignored compared to C^{extra}, such unique labeling may reduce the power dissipation due to reduced replication of internal gates.

Thus, for each gate that has multiple labels and has at least one fanout with each label, the impact on power dissipation and gate replication due to unique label assignment should be analyzed. Once this impact on power dissipation and gate replication is available, an informed decision can be made based on the power-area trade-offs desired. Based on the values of power gain or loss and area gain, the following three scenarios may exist.

1. There is no area gain. In this case, it can be shown that no power gain is possible and hence the labels of the gates should not be changed.
2. An area gain is accompanied either by a power gain or by no change in power. In this case, changing the gate label is desirable and hence should be performed. It should be noted that when there is no additional power dissipated on an external net (i.e., $C^{extra} = 0$), an area gain will always be accompanied by a power gain.
3. An area gain is accompanied by a loss in power. In this case, the decision must be made based on the relative importance of area and power costs.

Unfortunately, experimental results indicate that most of the area recovery achievable by this technique is accompanied by a loss in power when C^{basic} is negligible compared to C^{extra}. In this case, if power dissipation is of prime concern, this technique for area recovery should not be used.

4.3 Area Recovery by Relabeling Based on Required Times

Once all the gates in the circuit have been assigned unique labels, the relabeling technique proposed by Touati [13] can be applied to recover the area even further. This technique traverses the circuit in a preorder fashion from primary outputs to primary input (processing all the outputs of a gate before processing the gate itself) setting the required time at the gates based on the delay criticality of the gates. When two or more fanouts of a replicated gate have greater required time then the arrival time at the gate, the sub-cluster rooted at that gate can be implemented as separate cluster to reduce the replication. This is shown in Figure 9.

Once again, for each case of relabeling, the power cost/gain needs to be analyzed before relabeling the gates. Fortunately, in this case the power impact can be captured easily as it will depend only on the new cluster under consideration (and not on the transitive fanin cone of the cluster as is the case in relabeling by unique label assignment). Depending on the relative values of C^{extra} and C^{basic}, such relabeling may improve the power dissipation or degrade the power dissipation. However, if C^{basic} is negligible compared to C^{extra}, this technique will also always lead to increased power dissipation and hence should not be used. It should also be noted that this technique always increases the number of clusters.

5 Experimental Results and Extensions

We implemented POWER_CLUSTER in the SIS environment and compared our results with the SIS *reduce_depth* command that implements the original Lawler's algorithm [13]. Since *reduce_depth*

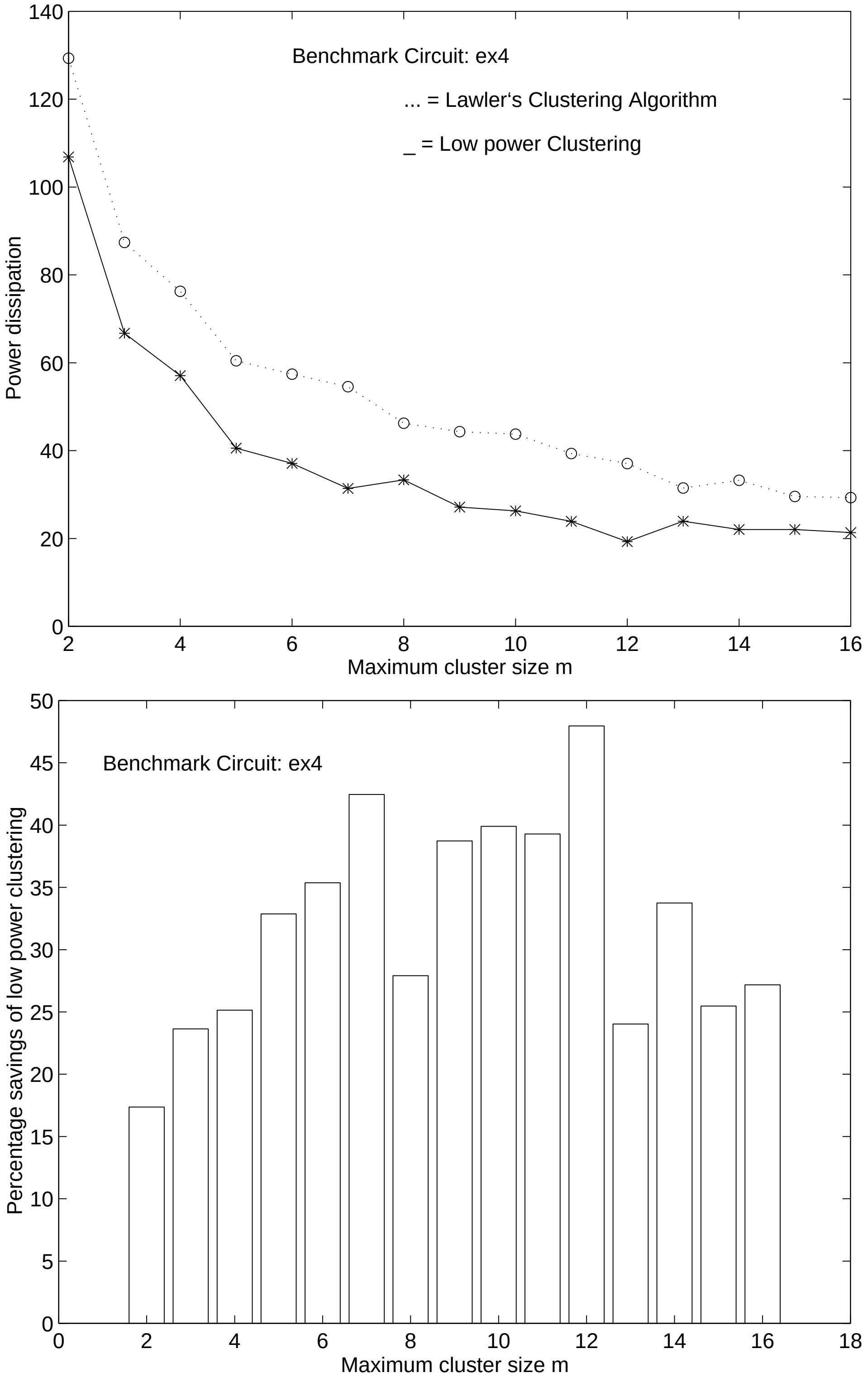


Figure 10: Effect of varying the maximum cluster size.

clustering at the next level of hierarchy. When minimizing equation (3), this guarantees that power dissipation will monotonically decrease at each level of hierarchy, resulting in an effective performance-driven partitioning strategy for low power. Based on this analysis, for rest of the experiments a cluster size of 8 was selected.

5.2 Results without Area Recovery

First, we report results of POWER_CLUSTER while minimizing power dissipation under the simplified power model of equation (3). In Table 1, the results of applying Lawler's clustering algorithm on some two-level and multi-level benchmark circuits are presented. Rest of the results presented in this section are normalized with respect to the values in Table 1. For this experiment, the maximum cluster size was set to be 8, i.e., $m = 8$. However, it was observed that the results show improvements for other values of m as well. Gate replication ratio reported is the ratio of gate count after clustering to the gate count before clustering. The main reason for choosing cluster size of 8 was to control the run time of power clustering algorithm. As mentioned above, if clusters of larger size are required, then the clustering algorithm can be applied in a hierarchical fashion.

As can be seen from Table 2, on average, for two-level benchmark circuits (first set of circuits in Table 2), an improvement of about 41% improvement in power dissipation and about 24% in cluster count was obtained. However, the gate replication ratio increased by an average of 11%. This degradation was expected as for this set of results, POWER_CLUSTER does not apply any post-clustering area recovery heuristics while *reduce_depth* does use these heuristics. The reduction in the cluster count is a byproduct of power minimization during clustering, i.e., minimizing power not only amounts to selecting low switching activity external nets, but also to minimizing the number of external nets, thereby reducing the cluster count. For the multi-level benchmark circuits, however, the power dissipation is reduced by 29% but the gate replication has increased by 23%. The higher rate of gate replication can be attributed to high degree of reconvergent fanouts in these circuits. Since reconvergent fanouts are the cause of the area and power non-optimality of POWER_CLUSTER, and since for these results, POWER_CLUSTER is targeting only on minimizing the power, it is quite natural that the area cost is higher on circuits with high reconvergent fanouts. For all circuits, on average, the power dissipation was improved by 35% and the number of clusters was improved by about 19%. The gate replication ratio increased by an average of 18%.

5.3 Results with Area Recovery

In the cases where area cost is also of concern, area can be recovered as mentioned in Section 4. Since, the area recovery by forcing single solution at each gate is an integral part of POWER_CLUSTER, to achieve further area recovery, we need to apply the relabeling techniques described in Section 4. As mentioned in Section 4, these techniques are accompanied by an increase in power dissipation and cluster count in most cases. The results obtained by integrating the area recovery mechanisms in POWER_CLUSTER are shown in Table 3. As can be seen, by applying relabeling techniques, the replication ratio was in fact improved by 6% while improving the power dissipation by 11%.

Circuit	Cluster Count	Repl. Ratio	Power
Two-level Benchmarks			
b12	26	1.14	9.71
cordic	31	1.45	10.58
cps	631	1.21	140.45
duke2	201	1.28	44.44
ex1010	845	1.77	163.92
ex4	126	1.12	46.25
misex2	40	1.30	5.27
misex3c	211	1.22	60.42
pdc	624	1.37	161.26
rd84	58	1.84	18.40
spla	464	1.40	114.59
Multi-level Benchmarks			
C1355	116	1.18	53.68
C2670	312	1.50	119.76
C432	104	1.53	36.57
C499	136	1.76	55.50
C880	153	1.71	48.72
apex6	250	1.52	75.00
apex7	105	1.64	35.18
b9	38	1.25	11.35
dalu	547	1.71	123.92
des	1396	1.93	366.04
k2	632	1.58	33.71
rot	278	1.48	100.61
t481	572	2.20	18.86

Table 1: Results using Lawler's clustering algorithm.

Circuit	Cluster Count	Repl. Ratio	Power
b12	0.77	1.09	0.61
cordic	0.90	1.01	0.82
cps	0.72	1.30	0.45
duke2	0.68	1.08	0.44
ex1010	0.89	1.02	0.75
ex4	0.77	1.00	0.72
misex2	0.68	1.03	0.43
misex3c	0.62	1.09	0.46
pdc	0.75	1.32	0.54
rd84	0.83	1.10	0.79
spla	0.75	1.21	0.52
Average	0.76	1.11	0.59
C1355	0.86	1.15	0.82
C2670	0.93	1.34	0.81
C432	0.59	1.24	0.51
C499	0.75	0.99	0.79
C880	0.84	1.36	0.73
apex6	0.74	1.08	0.70
apex7	0.77	1.15	0.62
b9	0.82	1.14	0.71
dalu	0.98	1.28	0.73
des	0.95	1.26	0.85
k2	1.00	1.55	0.72
rot	0.74	1.34	0.69
t481	1.15	1.12	0.53
Average	0.86	1.23	0.71
ALL Average	0.81	1.18	0.65

Table 2: POWER_CLUSTER without area recovery.

Circuit	Cluster Count	Repl. Ratio	Power
b12	0.92	0.96	0.77
cordic	1.06	0.98	1.10
cps	0.96	1.01	0.81
duke2	0.92	0.98	0.82
ex1010	1.06	0.93	1.09
ex4	0.92	0.98	0.89
misex2	0.82	0.98	0.76
misex3c	0.93	0.99	0.85
pdc	0.99	0.98	0.88
rd84	0.97	1.02	0.95
spla	0.98	0.98	0.86
Average	0.96	0.98	0.89
C1355	0.86	1.08	0.82
C2670	0.84	0.87	0.83
C432	0.74	0.98	0.74
C499	0.71	0.82	0.77
C880	0.83	0.94	0.81
apex6	0.82	0.95	0.79
apex7	0.82	0.94	0.72
b9	0.97	1.03	0.90
dalu	0.87	0.76	0.79
des	0.91	0.98	0.87
k2	1.12	0.96	1.18
rot	0.81	0.90	0.82
t481	1.30	0.96	1.61
Average	0.89	0.94	0.90
ALL Average	0.92	0.96	0.89

Table 3: POWER_CLUSTER with area recovery.

5.4 Results Considering Internal Power Dissipation

As mentioned earlier, algorithm POWER_CLUSTER has the capability of performing low power clustering even in the cases when the internal power dissipation (i.e., value of C^{basic}) is not negligible as compared to the extra power dissipation between clusters (i.e., value of C^{extra}). To show the effectiveness of POWER_CLUSTER in such a case, we report the results obtained for the other extreme case, i.e., when C^{extra} is negligible ($C^{extra} = 0$). It is interesting to note that just by changing the relative importance of C^{extra} with respect to C^{basic} we get quite a different problem. Specifically, in this case, since power dissipation of a net does not differ whether it is an internal net or an external net, the problem is to achieve delay optimality by replicating as many low power consuming gates as possible and by avoiding replication of high power consuming gates. The results obtained in this case is shown in Table 4. As can be seen on average, an improvement of about 11% in total power dissipation was obtained, at the same time improving the gate area by 9%.

6 Conclusion

In this paper, a delay optimal clustering algorithm for low power was proposed. The proposed algorithm is delay and power optimal for tree structures whereas it is delay optimal for general DAGs under a generalized load independent delay model. The optimality claims for this algorithm stem from the underlying clustering mechanism that enumerates all feasible cluster patterns at each gate in the circuit and maintains only the power optimal solutions at each gate for each arrival time value. In the process, it has been shown that enumeration of cluster patterns relates to enumeration of alphabetic trees and provided upper-bounds on the number of cluster pattern that need to be identified at each gate. Indeed, it is shown that if the cluster size and the maximum number of inputs to any gate are bounded by a constant number, the runtime of the proposed algorithm is polynomial in the circuit size. Also, formal methods were proposed to enumerate all cluster patterns efficiently, specifically with respect to non-binary tree structures.

Experimental results indicate that the proposed algorithm generates circuits with exactly the same delay as traditional delay optimal clustering mechanisms but at a substantial saving in power dissipation and number of clusters. Also, the results indicate that in the case where gate replication needs to be controlled, this algorithm can in fact produce circuits with better gate replication while improving the power dissipation.

The only disadvantage of this algorithm is the increased runtime which may be prohibitive in some cases. Though the time complexity is polynomial in the size of the circuit, it is exponential in the maximum size of the partition due to the pattern enumeration. To speed up the runtime, some quick method of identifying "good" patterns should be used. Indeed, an algorithm that needs to enumerate only $O(m^T)$ clusters at each gate based on the cluster size distribution at immediate fanins of gates can be developed. However, since switching activities at gates do not follow a predictable pattern in terms of circuit structure, any method that does not consider all cluster patterns is likely to be sub-optimal for DAG circuits.

Apart from the power minimization during clustering, this algorithm contributes to the general field of delay optimal clustering. Also, it has the capability of tracking the gate replication during clustering, which can be used to optimize area during clustering. In fact, experimental results indicate that when applied purely in an area driven mode, this algorithm can improve area by about 10% for the same delay and about the same cluster count.

Circuit	Cluster Count	Repl. Ratio	Power
b12	1.08	0.95	0.95
cordic	1.19	0.86	0.85
cps	1.01	0.98	0.98
duke2	1.05	0.99	0.98
ex1010	1.05	0.90	0.89
ex4	1.07	0.99	0.99
misex2	1.07	1.00	1.00
misex3c	1.08	0.94	0.94
pdc	0.97	0.90	0.88
rd84	1.00	0.92	0.91
spla	1.00	0.90	0.88
Average	1.05	0.94	0.93
C1355	1.00	1.00	1.00
C2670	0.90	0.80	0.79
C432	0.92	1.06	1.07
C499	0.84	0.79	0.74
C880	0.92	0.77	0.79
apex6	0.99	0.95	0.95
apex7	1.01	0.94	0.94
b9	1.13	1.03	1.02
dalu	0.91	0.79	0.83
des	0.96	0.83	0.80
k2	1.28	1.04	0.89
rot	0.95	0.84	0.83
t481	0.94	0.64	0.38
Average	0.98	0.88	0.85
ALL Average	1.01	0.91	0.89

Table 4: POWER_CLUSTER minimizing total power.

References

[1] L. Benini, M. Favalli, and B. Ricco. Analysis of hazard contribution to power dissipation in CMOS IC's. In *Proc. 1994 International Workshop on Logic Power Design*, pages 27–32, April 1994.

[2] R. Burch, F. Najm, P. Yang, and D. Hocevar. Pattern-independent current estimation for reliability analysis of CMOS circuits. In *Proceedings of the 25th Design Automation Conference*, pages 294–299, June 1988.

[3] K-Y. Chao and D. F. Wong. Low power considerations in floorplan design. In *Proc. 1994 International Workshop on Logic Power Design*, pages 45–50, April 1994.

[4] K. Chaudhary and M. Pedram. A near-optimal algorithm for technology mapping minimizing area under delay constraints. In *Proceedings of the 29th Design Automation Conference*, pages 492–498, June 1992.

[5] J. Cong, C-K. Koh, and K-S. Leung. Wire sizing with driver sizing for performance and power optimization. In *Proc. 1994 International Workshop on Logic Power Design*, pages 81–86, April 1994.

[6] A. D. Friedman and P. R. Menon. *Theory and Design of Switching Circuits.* Computer Science Press, 1975.

[7] A. Ghosh, S. Devadas, K. Kuetzer, and J. White. Estimation of average switching activity in combinational and sequential circuits. In *Proceedings of the 29th Design Automation Conference*, pages 253–259, June 1992.

[8] E. L. Lawler, K. N. Levitt, and J. Turner. Module clustering to minimize delay in digital networks. In *IEEE Transactions on Computers*, volume C-18, pages 47–57, January 1969.

[9] R. Murgai, R. K. Brayton, and A. Sangiovanni-Vincentelli. On clustering for minimum delay/area. In *Proceedings of the IEEE International Conference on Computer Aided Design*, pages 6–9, November 1991.

[10] R. Murgai, Y. Nishizaki, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli. Logic synthesis for programmable gate arrays. In *Proceedings of the 27th Design Automation Conference*, pages 620–625, June 1990.

[11] F. N. Najm. Transition density, a stochastic measure of activity in digital circuits. In *Proceedings of the 28th Design Automation Conference*, pages 518–521, June 1991.

[12] R. Rajmohan and D. F. Wong. Optimal clustering for delay minimization. In *Proceedings of the 30th Design Automation Conference*, pages 309–314, June 1993.

[13] Herve Touati. *Performance Oriented Technology Mapping.* PhD thesis, University of California, Berkeley, 1990.

[14] C. Tsui, M. Pedram, and A. Despain. Efficient estimation of dynamic power dissipation under a real delay model. In *Proceedings of the IEEE International Conference on Computer Aided Design*, November 1993.

[15] H. Vaishnav. *Optimization of Post-Layout Area, Delay and Power Dissipation.* PhD thesis, University of Southern California, Los Angeles, 1995.

[16] H. Vaishnav and M. Pedram. PCUBE: a performance-driven placement algorithm for low power designs. In *Proceedings of the European Design Automation Conference*, September 1993.

[17] H. Vaishnav and M. Pedram. Routability-driven fanout optimization. In *Proceedings of the 30th Design Automation Conference*, pages 230–236, June 1993.