Area/Pin-Constrained Circuit Clustering for Delay Minimization

Honghua Yang and D. F. Wong

Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712

Abstract

We consider the problem of circuit partitioning for multiple-chip implementation. One motivation for studying this problem is the current needs of good partitioning tools for implementing a circuit on multiple FPGA chips. We allow replication of logic gates as it would reduce circuit delay. Circuit partitioning with replication of logic gates is also called circuit clustering. In this paper, we present a circuit clustering algorithm that minimizes circuit delay subject to area and pin constraints on each chip, under the general delay model. Our algorithm is optimal under either the area constraint only or the pin constraint only. We tested our algorithm on a set of benchmark circuits and consistently obtained optimal or near-optimal results.

1 Introduction

Circuit partitioning [PL88] for multiple-chip design is an important CAD problem of current industrial interest. On the one hand, complexity of current circuit and system designs often require multiple-chip implementation independent of the technology being used. On the other hand, new ASIC technology such as Field Programmable Gate Arrays (FPGAs), due to its more restricted logic and routing resources, often requires more chips to realize a circuit design as compared with traditional ASIC technologies. Most of the existing partitioning algorithms [PL88] primarily focus on wiring minimization and ignore circuit performance issues. In this paper, we consider the problem of partitioning a given circuit into chips to minimize circuit delay subject to area and pin constraints on each chip. We allow replication of logic gates as it would reduce circuit delay. Circuit partitioning with replication of logic gates is also referred to as circuit clustering.

Previous research on circuit clustering for delay minimization focused on either area-constrained clustering or pin-constrained clustering. In 1966, Lawler, Levitt and Turner [LLT66] presented a polynomial time optimal algorithm for area-constrained circuit clustering assuming a unit delay model. In the unit delay model, a constant delay (of one unit) is associated with every interconnection connecting two gates in different chips and no delay is associated within a chip itself. Recently, Cong and Ding [CD92] presented an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs. Although their algorithm was developed for a different application, it can be shown that the algorithm in fact is an optimal algorithm for pin-constrained clustering under the unit delay model.

The unit delay model is not very realistic, as it assumes that the inter-chip delay totally outweighs any delay within a chip. In general, a signal path may passes through many gates inside a chip, hence incurring a substantial delay within the chip itself. In 1991, Murgai, Brayton and Sangiovanni-Vincentelli [MBS91] proposed a general delay model, a more realistic extension of the
unit delay model. In this model, we have (1) each gate \( v \) of the circuit has a gate delay given by \( \delta(v) \); (2) no delay is encountered on an interconnection linking two gates within a chip; (3) a delay of \( D \) time units (\( D \) is a given constant) is encountered on every interconnection linking two gates in different chips. The authors of [MBS91] considered the area-constrained clustering problem under the general delay model and presented a greedy heuristic algorithm for solving the problem. Their algorithm produces reasonable solutions but in general not optimal. In 1993, Rajaraman and Wong [RW93] presented an optimal algorithm for area-constrained clustering under the general delay model. A major shortcoming of the algorithms in [MBS91] and [RW93] is that they do not consider pin constraints. We note that the algorithm of Woo and Kim in [WK93] explicitly considers area and pin constraints. However, the optimization objective of their algorithm is wiring rather than circuit delay.

Delay minimization under both area and pin constraints is a difficult problem because of the totally different nature of the two types of constraints (e.g. area constraints is monotone and pin constraints is not [LLT66]). In this paper, we present an efficient circuit clustering algorithm to minimize delay under the general delay model subject to both area and pin constraints. This is an extension of the previous work done in [RW93], together with applying maximum network flow techniques. Our algorithm is optimal under either the area constraint only or the pin constraint only. We tested our algorithm on a set of benchmark circuits and consistently obtained optimal or near-optimal results.

The rest of the paper is organized as follows. In section 2, we give a formal definition to our problem. Section 3 presents an algorithm for solving the problem. The algorithm relies on a procedure that finds an area/pin-constrained sub-circuit rooted at a particular node. Section 4 presents an algorithm which is based on maximum network flow techniques to implement such procedure, and analyzes the complexity of our algorithm. Section 5 discusses the post-processing step of our algorithm. Finally we show some experimental results in section 6. The full version of this paper is in [YW94].

2 Problem Formulation and Preliminaries

A combinational circuit is represented by a directed acyclic graph (DAG) \( N = (V, E) \) where \( V \) is the set of nodes representing gates\(^1\) and input nodes, and \( E \) is the set of directed edges representing interconnections between gates. Each node \( v \) in \( N \) is assigned a positive weight \( w(v) \) (representing the area taken by the gate, for example). The indegree of a node is the number of edges incoming to the node, and the outdegree of a node is the number of edges outgoing from the node. A primary input (PI) node is a node with indegree 0, and a primary output (PO) node is a node with outdegree 0. A cluster is a subset of nodes in \( V \).

A solution to the circuit clustering problem of \( N \) is a logically equivalent circuit \( N_C \) with a node partition, such that, (1) \( N_C \) consists of \( N \) with some replicated nodes and edges, (2) each set in the node partition is a cluster in \( N \), and (3) a PI node is not included in any cluster\(^2\).

An inter-cluster edge is an edge in \( N_C \) that connects gates in two clusters, or a PI node and a gate in a cluster. The constant delay \( D \) is also called an inter-cluster delay. The input nodes of a cluster \( C \) in \( N_C \), denoted by \( Input(C) \), are the nodes outside of \( C \) that have wires incoming to nodes

\(^1\)A gate is either a simple gate or a complex gate such as a lookup-table in an FPGA.

\(^2\)This is because a PI node represents a signal from the outside world. We assume that the gate delay \( \delta(u) \) for a PI node \( u \) is 0.
in $C$. The output nodes of $C$ in $N_C$, denoted by $Output(C)$, are the nodes in $C$ with wires outgoing from nodes in $C$. The area of $C$, denoted by $Area(C)$, is the sum of the gate weights in $C$. Let $A$ and $P$ be two given constants denoting the area constraint and the pin constraint, respectively. A cluster $C$ in a solution is feasible if $Area(C) \leq A$, and $|Input(C)| + |Output(C)| \leq P$. A solution $N_C$ is feasible if every cluster in $N_C$ is feasible. Given a solution $N_C$, the delay along a path in $N_C$ is the sum of the gate delays and inter-cluster delays on the path. The delay of a node $v$ in $N_C$ is the maximum delay along any path from a PI node to $v$. The delay through the circuit is the maximum delay among all PO nodes. A feasible solution $N_C$ is optimal if the delay through $N_C$ is the minimum among all feasible solutions for the circuit clustering problem of $N$.

Figure 1 shows a circuit clustering example. We use the following notations throughout the paper. Given a node $v$ in a circuit $N$, $N_v$ denotes the subgraph of $N$ induced by $v$ and all its predecessors, with $v$ being the only output node; a cluster rooted at $v$, denoted by $C_v$, is a subgraph of $N_v$ containing $v$ as the only output node. For a node $u \in N_v$, $\Delta(u, v)$ denotes the maximum total gate delay along any path in $N$ from the output of $u$ to the output of $v$. We also let $d(v)$ denote the maximum delay of node $v$ in an optimal clustering solution of $N_v$ and let $d'(u, v) = d(u) + \Delta(u, v)$ for a node $u \in N_v$.

**Lemma 2.1** Let $N_C$ be a delay optimal solution to the circuit clustering problem of a circuit $N$. Then there is another optimal solution $N'_C$ such that every cluster in $N'_C$ is a rooted cluster.

Lemma 2.1 suggests that we can replace the pin constraint of a feasible cluster $C$ in a solution by $|Input(C)| \leq P - 1$.

## 3 An Algorithm for Optimal Clustering

Similar to the optimal algorithm given in [RW93], our algorithm CLUSTER contains two phases: labeling and clustering. In the labeling phase, for each node $v$ in $N$ in topological order, we compute $d(v)$, which is the delay of node $v$ in an optimal clustering solution of $N_v$. In the clustering phase, clusters in an optimal solution are generated starting from the PO nodes in a bottom-up fashion, based on the information obtained in the labeling phase. We described in section 5 a post-processing phase to reduce the number of clusters without affecting the delay of the circuit computed in the labeling phase. The CLUSTER algorithm is listed in Algorithm 1.
Algorithm 1 CLUSTER
Input: A combinational circuit $N = (V, E)$, $A$, $P$ and $D$.
Output: The delay of $N$ and the corresponding cluster for each node in $N$.
begings
/* The labeling phase */
0. Assign 0 to $d(u)$ for each PI node $u$;
1. Sort the non-PI nodes in $V$ by topological order into a list $V_T$;
2. for each node $v$ in $V_T$ in topological order
3. Construct $N_v$;
4. Calculate $\Delta(u, v)$ for every $u \in N_v$ to $v$;
5. Calculate $d'(u, v) = d(u) + \Delta(u, v)$ for every $u \in N_v$ except $v$;
6. Sort all the different values of $d'(u, v)$ in strictly decreasing order $d'_1 > d'_2 \ldots > d'_m$;
7. $k_1 = \max\{i \mid \sum_{u \in V} d'_u \geq d'_i \};$ if $u \leq A$;
8. $k_2 = \max\{i \mid \exists$ PI node $u \in N_v$, s.t. $d'(u, v) \geq d'_i \};$
9. $k = \min\{k_1, k_2\};$
10. $done = false;$
11. while not done
12. Find a feasible rooted cluster $C_v$ containing all nodes $u$ with $d'(u, v) \geq d'_k$;
13. if such $C_v$ exists
14. $d(v) = d'_{k+1} + D$; $done = true;$
15. else $k = k - 1;$
endif;
endwhile;
endfor
/* The clustering phase */
16. $S = \phi$; $L$ = the list of PO nodes in $N$;
17. while $L$ is not empty
18. Remove a node $v$ from $L$;
19. $S = S \cup \{C_v\}$; $L = L \cup INPUT(C_v)$;
endwhile
end.

In the labeling phase, we observe that $d'(u, v)$ is a lower bound on the maximum delay along any path from a PI node to $v$ that passes through $u$. The greater the value of $d'(u, v)$, the more need to include $u$ in $C_v$. Therefore we first try to include all high $d'$-valued nodes $u$ (with $d'(u, v) \geq d'_k$) into $C_v$ that satisfies the area constraint, and then repeatedly find a superset of $C_v$ that would satisfy both the area and the pin constraints. If we cannot find a feasible $C_v$ containing all nodes $u$ with $d'(u, v) \geq d'_k$ in step 12, then we remove the smallest $d'$-valued nodes from $C_v$ (in step 15), and try to find a feasible $C_v$ that contains all nodes $u$ with $d'(u, v) \geq d'_{(k-1)}$ in the cluster. If we do find a feasible $C_v$ in step 12, then we assign $d(v)$ to be the largest $d'$ value outside $C_v$ plus the inter-cluster delay $D$. Note that we need to compute $k_2$ in step 8, since by our definition a cluster does not contain any PI node. If there is a PI node $u$ with $d'(u, v) \geq d'_{k_1}$, then $d'(u, v)$ dominates the $d'$ values outside the cluster which consists of all nodes $w$ such that $d'(w) \geq d'_{k_1}$. The clustering phase is straightforward. We will discuss the post-processing phase in detail in section 5. Figure 2 shows a labeling process of node $v$.

Since the delay of a node $v$ in a cluster $C$ depends on the delay of the input nodes of $C$ and the total gate delay along every path from an input node of $C$ to $v$, it is intuitive to see that algorithm CLUSTER gives us an optimal solution to the circuit clustering problem. The correctness proof of our algorithm is similar to the proof for the area-constrained case in [RW93], and therefore we suggest that interesting readers read [RW93] for reference.

Clearly, the optimality of the solutions produced by our algorithm depends on how to implement step 12. A straightforward implementation would be to enumerate every possible input node
Figure 2: A labeling process for node \( v \), assuming \( A = 4, P = 6, D = 1, w(v) = 1, \delta(v) = 1 \) for each non-PI node \( v \), and the \( d(u)'s \) for all predecessors \( u \) of \( v \) are already computed.

Set with size \( \leq P - 1 \) and test whether the node set can be used as the input nodes of a cluster rooted at \( v \). If the answer is yes, we examine the corresponding cluster to see whether it is feasible and whether it contains all nodes \( u \) with \( d'(u, v) \geq d'_k \). For convenience, we refer the latter condition as the node constraint. Although this implementation has a polynomial-time complexity \( O(|N|^{P+c}) \) for some nonnegative constant \( c \), it is far from realistic since the constant \( P \) is usually very large. Therefore, in the next section, we will instead propose an efficient polynomial time algorithm using maximum network flow techniques to implement step 12. The algorithm can find a node-constrained feasible cluster rooted at \( v \) in most cases when there indeed exists one. Consequently, our algorithm always finds optimal or near optimal solutions to the circuit clustering problem.

4 Finding a Node-Constrained Feasible Cluster Rooted at \( v \)

Our strategy is to first eliminate the node constraint (i.e., the cluster contains all nodes \( u \) with \( d'(u, v) \geq d'_k \)), and then deal with the other two constraints by repeatedly applying maximum flow minimum cut (max-flow min-cut) techniques.

In order to guarantee that every rooted cluster we consider satisfies the node constraint, we collapse all the nodes \( u \) in \( N_v \) with \( d'(u, v) \geq d'_k \) to node \( v \) and add weights of these collapsed nodes to the weight of \( v \).

4.1 Finding a Feasible Cluster Rooted at \( v \)

Let us first introduce some terminologies on a single-source \( s \) and single-sink \( t \) directed graph \( N \). We define a cut \((X, \bar{X})\) of \( N \) to be a partition of nodes in \( N \) such that \( s \in X \) and \( t \in \bar{X} \). The edge cut \( e(X, \bar{X}) \) induced by the cut \((X, \bar{X})\) is the set of edges of \( N \) whose starting node is in \( X \) and ending node is in \( \bar{X} \). The node cut \( n(X, \bar{X}) \) induced by the cut \((X, \bar{X})\) is the set of nodes of \( X \) who are the starting nodes of the edges in \( e(X, \bar{X}) \). A node cut is a flat node cut if every node in the node cut can be reached by \( s \) without going through other nodes in the node cut.

A cut \((X, \bar{X})\) of \( N \) is a min-cut if its induced node cut \( n(X, \bar{X}) \) contains the minimum number of nodes among all cuts of \( N \), i.e. \(|n(X, \bar{X})|\) is minimum. (It is easy to see that the node cut induced by a min-cut must be a flat node cut.) A min-cut \((X, \bar{X})\) of \( N \) is a min-area min-cut if Area\((\bar{X})\) is
Algorithm 2 Finding a Feasible Rooted Cluster $C_v$ Containing all Nodes $u$ with $d'(u, v) \geq d_x^*$

Input: A network $N_v$, $A$, $P$, $d_x^*$, and $d'(u, v)$ for each $u$ in $N_v$.
Output: A feasible cluster $C_v$ rooted at $v$.
begin
0. Collapse all the nodes $u$ with $d'(u, v) \geq d_x^*$ to $v$;
    Add weights of these collapsed nodes to the weight of $v$;
    Delete all edges outgoing from $v$;
    Add the source $s$ to $N_v$ and identify $v$ as the sink;
1. Find a min-area min-cut $(X, \bar{X})$ in $N_v$;
2. If $|n(X, \bar{X})| > P - 1$
3. return(NULL);
4. endif;
5. while $\text{Area}(X) > A$
6. Construct the cut induced subcircuit $N_X$; $P' = P$;
    Add the source $s$ to $N_X$ and identify $v$ as the sink;
7. for each node $u$ in $n(X, \bar{X})$
8. if $u$ is a fanin of the node $v$
9. Mark $u$ as a cut node;
endfor;
10. if all nodes in $n(X, \bar{X})$ are marked as cut nodes
11. Remove the nodes in $n(X, \bar{X})$ from $N_X$;
12. $P'' = P'' - |n(X, \bar{X})|$
13. Find a new min-area min-cut $(\bar{X}, \bar{X})$ in the remaining $N_X$;
14. else
15. for each node $u$ in $n(X, \bar{X})$ that is not marked as a cut node
16. Merge $u$ with the source $s$ in $N_X$;
17. Find a new min-area min-cut $(X_u, \bar{X}_u)$ in the resulting $N_X$;
endfor
18. $(X, \bar{X}) = \text{the cut with the minimum area among the cuts with}$
    the minimum node cut size in $\{(X_u, \bar{X}_u)\}$;
endfor;
19. if $|n(X, \bar{X})| > P' - 1$
20. return(NULL);
endif;
endwhile;
21. Mark the nodes in $n(X, \bar{X})$ as cut nodes;
22. return($\bar{X}$);
end.

the minimum among all min-cuts of $N$. A cut $(Y, \bar{Y})$ of $N$ is an undercut of another cut $(X, \bar{X})$ of $N$ if $|n(Y, \bar{Y})| \leq |n(X, \bar{X})|$ and $\bar{Y} \subseteq \bar{X}$. (It is easy to see that if $\bar{X}$ is a feasible cluster rooted at $t$, then $\bar{Y}$ is also a feasible cluster rooted at $t$.)

The cut $(X, \bar{X})$ induced subnetwork, denoted by $N_X$, is the subgraph of $N$ induced by $n(X, \bar{X}) \cup \bar{X}$. (It is easy to see that the nodes in $n(X, \bar{X})$ are the PI nodes of $N_X$.) An example of $N_X$ is shown in Figure 4. The node cut $n(X, \bar{X})$ in Figure 4 is a flat node cut, and the node cut $n(X, \bar{X}) \cup \{u\}$ is not a flat node cut.

Our strategy is to satisfy the pin constraint first. Our algorithm works as follows. We first find a min-area min-cut $(X, \bar{X})$ in $N_v$. If the node cut size is greater than $P - 1$, then with the current node constraint no feasible cluster rooted at $v$ can be found. If the node cut size satisfies the pin constraint, but the $\text{Area}(\bar{X}) > A$, then we try to find a min-area min-cut $(X_u, \bar{X}_u)$ in $N_X$ that excludes at least one node $u$ in $n(X, \bar{X})$ ($(X_u, \bar{X}_u)$ is a cut with the minimum area among the cuts with the minimum node cut size of $N_X$ whose induced node cut does not contain $u$). We designate this cut as $(X, \bar{X})$ and repeat the above procedure. The iteration stops when either a feasible cut is
found or no new cut \((X, \bar{X})\) with \(|n(X, \bar{X})| \leq P - 1\) can be found. A similar strategy was mentioned in [CD92] for post-processing in their Flow-Map algorithm.

**Lemma 4.1** Let \(n(X_0, \bar{X}_0)\) be any flat node cut in \(N_v\). Let \((X, \bar{X})\) be a min-area min-cut in \(N_{X_0}\) that excludes some node in \(n(X_0, \bar{X}_0)\). Then any cut \((Y, \bar{Y})\) in \(N_{X_0}\) satisfying \(n(X_0, \bar{X}_0) \subseteq n(X, \bar{X}) \cup n(Y, \bar{Y})\) has an undercut in \(N_X\).

The above lemma suggests that when searching for a feasible cut in \(N_{X_0}\), we can always narrow down our search for a feasible cut to a smaller circuit \(N_X\), where \((X, \bar{X})\) is a min-area min-cut in \(N_{X_0}\) that excludes some node in \(n(X_0, \bar{X}_0)\). The chance of possibly missing a feasible cut in \(N_{X_0}\) is relatively small (i.e., when for every feasible cut \((Y, \bar{Y})\) in \(N_{X_0}\), \(n(X_0, \bar{X}_0) \subseteq n(X, \bar{X}) \cup n(Y, \bar{Y})\).)

The strategy described above works well except for the case when all nodes in the node cut of the latest min-area min-cut \((X, \bar{X})\) have edges going to node \(v\), as the example shown in Figure 4. Since we only find cuts \((X', \bar{X}')\) in \(N_X\) such that all PI nodes of \(N_X\) are in \(X'\), no more min-area min-cut excluding a node in \(n(X, \bar{X})\) can be found in \(N_X\). In the example, however, the non-flat node cut \(n(X, \bar{X}) \cup \{v\}\) is a feasible cut. To overcome this problem, we remove from \(N_X\) all nodes in \(n(X, \bar{X})\) that have edges going to \(v\). We then continue the algorithm to find the flat node cut \(\{v\}\) on the remaining \(N_X\). Evidently, a feasible node cut found after this step must be a non-flat node cut made up by the union of a node cut in the remaining \(N_X\) and \(n(X, \bar{X})\). The detailed algorithm for finding a feasible cluster rooted at \(v\) is listed in algorithm 2.

Steps 1, 13 and 17 of algorithm 2 find a min-area min-cut in a single-source single-sink \(N_v\) with \(v\) being the sink and \(s\) being the source. Note that we define our min-cut as a cut with the minimum node cut size. In order to apply the traditional algorithms for finding maximum flow in \(N_v\), which by the max-flow min-cut theorem [ET79] also finds a cut with the minimum edge cut size, we apply the node-splitting technique ([ET79, CD92]) to reduce the minimum node cut size constraint to the minimum edge cut size constraint. We construct \(N_v'\) from \(N_v\) with the single source \(s\) and
single sink $v$ as follows. For each node $u$ in $N_v$ except $s$ and $v$, we define two nodes $u_1$ and $u_2$ with an edge $(u_1, u_2)$ in $N'_v$, which is called a bridging edge. For each edge $(u, w)$ ($u \neq s$ and $w \neq v$) in $N_v$, we define an edge $(u_2, w)$ in $N'_v$. For each edge $(s, u)$ and $(w, v)$ in $N_v$, we defined $(s, u_1)$ and $(u_2, v)$ in $N'_v$ respectively. Then we assign unit capacity to all bridging edges and infinite capacity to all other edges in $N'_v$.

We can show that the problem of finding a min-area min-cut (i.e. with the minimum node cut size) in $N_v$ is equivalent to finding a minimum edge cut with the minimum area in $N'_v$. The detailed steps on finding a min-area min-cut in $N_v$ is given in [YW94].

**Theorem 4.1** A feasible solution to the optimal circuit clustering problem can be found in $O(P^3 m n \log(\min\{n, A\}))$ where $n, m$ are the numbers of nodes and edges in the circuit, respectively.

**Theorem 4.2** By appropriately implementing step 12, the algorithm CLUSTER can always find an optimal solution efficiently for the circuit clustering problem under either the area constraint only or the pin constraint only.

**Proof:** Under either condition, the implementation of step 12 of algorithm 1 is simplified. Under the area constraint only, step 12 simply returns the cluster consisting of node $v$ and all nodes $u$ with $d'(u, v) \geq d'_v$, and our algorithm degenerates to the algorithm in [RW93].

Under the pin constraint only, step 12 can be implemented by applying steps 0-4 followed by steps 21-22 of algorithm 2 with the following modification: in step 0, we not only collapse the nodes $u$ with $d'(u, v) \geq d'_v$ to $v$, but also collapse to $v$ the set of nodes $L_v$ that can be reached from PI nodes only by going through edges outgoing from the collapsed $v$. Let us call the circuit $N'_v$ after collapsing nodes $u$ with $d'(u, v) \geq d'_v$ only as $N''_v$ and call the circuit obtained after the modified step 0 as $N'_v$. To prove the optimality of our algorithm under the pin constraint only, we show that the cut $(X, \bar{X})$ of $N''_v$ found in step 21 is indeed a min cut in $N'_v$. Suppose it is not the case. Let $(Y, \bar{Y})$ be a cut in $N'_v$ such that $|n(Y, \bar{Y})| < |n(X, \bar{X})|$. Since the nodes in $L_v$ can only have incoming edges from $v$ or other nodes in $L_v$, by moving the nodes in $L_v \cap Y$ to $\bar{Y}$ we will not introduce new edges from $Y$ to $\bar{Y}$. Hence the cut $(Z, \bar{Z})$ in $N'_v$ such that $Z = Y - L_v$, $\bar{Z} = \bar{Y} \cup L_v$ satisfies $|n(Z, \bar{Z})| \leq |n(Y, \bar{Y})|$. Clearly, $n(Z, \bar{Z})$ is also a node cut in $N''_v$. However, $|n(Z, \bar{Z})| < |n(X, \bar{X})|$, a contradiction. $\square$

We note that the above theorem gives the first polynomial time (delay) optimal algorithm for pin-constraint clustering under the general delay model. In a previous work, [CD92] gave a (delay) optimal algorithm for pin-constraint clustering under the unit delay model.

5 Post-Processing for Reducing the Number of Clusters

After performing the clustering phase for a circuit, the number of clusters in the resulting circuit is usually large, since many clusters are not to their full capacity and there are gate duplications. To find a feasible solution with the same optimal or near optimal delay computed in the labeling phase, but with less duplication and fewer number of clusters, we present the following simple two-step cluster reduction method: rooted cluster elimination and cluster packing. In the rooted cluster elimination step, we try to eliminate as many as possible rooted clusters by substituting the roots (i.e. the only output node) of these clusters with their replica in other clusters, provided that both pin constraint and the optimal network delay are not violated. In the cluster packing step, we try to pack as many as possible clusters into a single cluster using a heuristic algorithm that favors the packing of two clusters sharing the largest number of gates. The experimental results in the next section will show that the number of clusters and the gate duplications in the solution are dramatically reduced after applying the two-step cluster reduction method.
Table 1: Comparison of the delay obtained by CLUSTER-RW and CLUSTER, and the number of clusters generated by CLUSTER without post-processing and with post-processing, under area constraint \( A = 100 \), pin constraint \( P = 40 \), inter-cluster delay \( D = 2 \), and \( w(v) = 1, \delta(v) = 1 \) for each non-PI node \( v \).

As the experimental results in section 6 show, the number of clusters after cluster packing in the post-processing phase is dramatically reduced.

6 Experimental Results
We have implemented our CLUSTER algorithm and the CLUSTER-RW algorithm given in [RW93]\(^3\) using the C language on Sun SPARC workstations. We used input/output routines and general utility functions provided by MIS [BrRS87] in our implementation. Our program runs on any circuit whose maximum gate indegree does not exceed the pin constraint \( P \). (If the maximum gate indegree of a circuit exceeds \( P \), we can decompose the circuit using functions provided by MIS.) We tested CLUSTER and CLUSTER-RW on a set of MCNC benchmark combinational circuits, and a number of look-up table (LUT) based multi-FPGA circuits obtained from Flow-Map [CD92]. We assumed that the inter-cluster delay \( D = 2 \). For ease of testing, we assigned unit gate weight and unit gate delay to each non-PI node (gate or LUT) in a circuit. Our algorithm and program can deal with non-uniform gate weight and gate delay.

We set the area constraint per cluster to \( A = 100 \) and tested the following (see Table 1):

\(^3\)We re-implemented the CLUSTER-RW algorithm for comparison purposes, since the original implementation does not accept the BLIF net-list description input format [BrRS87].
CLUSTER-RW with no pin constraint, recording the number of clusters (#clu > 40 pins) that violate the pin constraint, and the optimal delay (opt del) for area constraint only clustering; CLUSTER with pin constraint P = 40 per cluster but without the post-processing step, recording the delay (del) for area/pin-constrained clustering, the number of clusters (#clu) in the solution, and the number of non-PI gates (non-PI gates) in the replicated circuit; and CLUSTER with P = 40 with the post-processing step, recording the number of clusters (#clu) in the solution after cluster packing, the number of non-PI gates (non-PI gates) in the resulting circuit, and the ratio between the number of non-PI gates in the final solution and the number of non-PI gates in the initial circuit.

Our results show that clustering without consideration of pin constraint, as in the case of CLUSTER-RW, often results in many clusters with unacceptably large pin numbers that violate the pin constraint. Our results also show that CLUSTER generates clusters that satisfy both area and pin constraints, and still consistently keeps the optimality or near optimality of the delay of the circuit obtained in clustering with the same area constraint but without pin constraint. In most cases, the delay generated by CLUSTER is the same as that generated by CLUSTER-RW, which is optimal. For the rest, the delay is off the optimal delay with pin constraint by one or two inter-cluster delay (D = 2). (Note that this does not mean that the delay is not optimal with the given pin constraint.) On the average, the delay generated by CLUSTER increases by 1.9% compared with the delay generated by CLUSTER-RW. The post-processing step proves to be effective, reducing the number of clusters in the solution (without increasing the delay) by 87% on the average.

It is also interesting to notice that for circuits with smaller delay, and hence smaller depth, CLUSTER-RW generates fewer clusters that violate the pin constraint. This is because in a circuit with small depth, the subcircuits Nq are small and therefore the clusters are small. In fact CLUSTER shows its strength in dealing with large and complicated circuits.

The running times of most examples are less than 1 minute on a Sun SPARC 10 workstation. A couple of large circuits take 15-30 minutes, with more than half of the time spent on the post-processing step.

References


