Optimal Clock Period Clustering for Sequential Circuits with Retiming

Peichen Pan, Arvind K. Karandikar, and C. L. Liu, *Fellow, IEEE*

***Abstract*— In this paper we consider the problem of clustering sequential circuits subject to a bound on the area of each cluster, with the objective of minimizing the clock period. Current algorithms address combinational circuits only, and treat a sequential circuit as a special case, by removing all flip-flops (FF's) and clustering the combinational part of the sequential circuit. This approach breaks the signal dependencies and assumes the positions of FF's are fixed. The positions of the FF's in a sequential circuit are in fact dynamic, because of retiming. As a result, current algorithms can only consider a small portion of the whole solution space. In this paper, we present a clustering algorithm that does not segment circuits by removing FF's. In additional, it considers the effect of retiming. The algorithm can produce clustering solutions with the optimal clock period under the unit delay model. For the general delay model, it can produce clustering solutions with a clock period provably close to optimal.**

I. Introduction

CIRCUIT partitioning/clustering is an important aspect of VLSI design [1]–[3]. It consists of dividing a circuit into parts, each of which can be implemented as a separate component (e.g., a chip) that satisfies certain design constraints. One such constraint is the area of the component. The limited area of a component forces the designer to lay out a circuit on several components. Since crossing components incurs relatively large delay, such a partitioning could greatly degrade the performance of a design if not done properly.

There has been a large amount of work done in the area of circuit partitioning and clustering [4]–[18]. In circuit partitioning, the circuit is divided into two (bipartitioning) or more (multiway partitioning) parts. In circuit clustering, the circuit is built up cluster by cluster. To partition designs of large size, bottom-up clustering is often combined with top-down partitioning. The classical objective of partitioning is to minimize the cut-size, i.e., the number of nets spanning two or more parts.

Manuscript received June 19, 1997. This work was supported in part by a Design Automation Scholarship Award. The work of C. L. Liu was supported in part by the National Science Foundation under Grant MIP-9612184. This paper was recommended by Associate Editor K. Cheng.

P. Pan is with the Department of Electrical and Computer Engineering, Clarkson University, Potsdam, NY 13699 USA (e-mail: panp@sun.soe.clarkson.edu).

A. K. Karandikar was with the Department of Electrical and Computer Engineering, Clarkson University, Potsdam, NY 13699 USA. He is now with the Validation Technology Group, Intel Corporation, Folsom, CA 95630 USA.

C. L. Liu was with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA. He is now with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan, R.O.C.

Publisher Item Identifier S 0278-0070(98)05020-9.

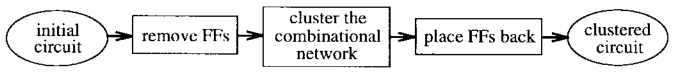


Fig. 1. Conventional clustering approach.

We consider circuit clustering subject to a bound on the area of each component, with possible replication of logic, i.e., a gate may be assigned to more than one component. Following [19], we refer to each such component as a cluster, and to the problem as the circuit clustering problem. Our objective is to minimize the clock period of the clustered circuit. The clock period of a sequential circuit is defined as the maximum delay between the FF's.

Previous work on clustering has mainly focused on combinational circuits. Effective heuristics and optimal algorithms have been proposed [19]–[21]. However, most circuits in practice are sequential. Clustering algorithms for combinational circuits can be applied to sequential circuits, by clustering the combinational logic between the FF's as was done in the past [19]. In other words, the FF's in a sequential circuit are simply removed to obtain a combinational network. Then the combinational network is clustered. Finally, the FF's are placed back, as indicated in Fig. 1.

For sequential circuits, there is a functionality preserving transformation known as retiming [22]. Retiming allows a designer to move the FF's around within a circuit. The original purpose of retiming is to minimize the clock period, the number of FF's or both, without modifying the structure of the circuit. Polynomial algorithms for doing so and their efficient implementations have been proposed in the literature [22]–[25].

The conventional clustering approach greatly restricts the solution space that can be explored, since it ignores other FF configurations that can be obtained using retiming. It also fails to consider signal dependencies across FF boundaries, since the logic is segmented into independent pieces after the removal of the FF's.

To illustrate the limitations of the conventional approach, consider the trivial example shown in Fig. 2. For the circuit in Fig. 2(a), assume the delay of each gate is 1, inter-cluster delay is 2, and each cluster can accommodate three gates.

Using the conventional method, i.e., removing all FF's and then clustering the remaining logic, the solution shown in Fig. 2(b) is obtained. The dot represents the inter-cluster delay. As can be seen, the parts of the circuit separated by the FF's are clustered individually, and the clustered circuit has a clock period of 2. Retiming at this stage cannot reduce the clock

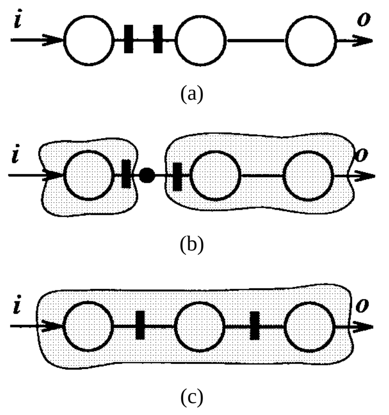


Fig. 2. Disadvantage of removing FF's during clustering.

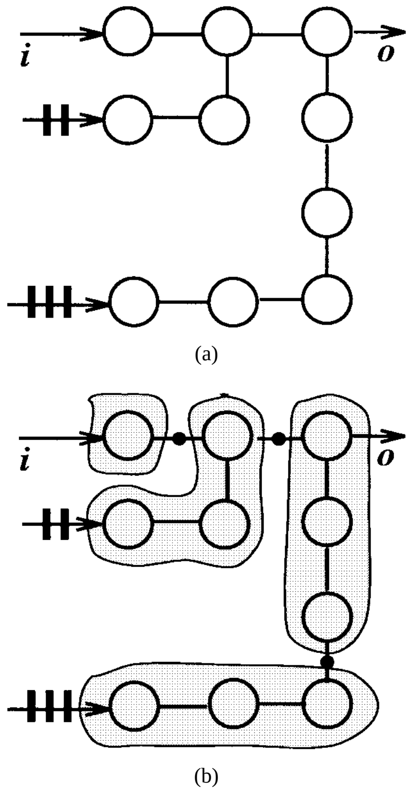


Fig. 3. Clustering while ignoring the effect of FF's.

period below this value. However, if the circuit is clustered with the FF's and then retimed, the solution shown in Fig. 2(c) can be obtained. This solution has a clock period of 1.

Ignoring the effect of FF's while trying to optimally cluster the combinational logic of a sequential circuit, as the conventional approach does, may not result in the best solution for the sequential circuit. Consider the circuit shown in Fig. 3(a). Assuming the delay of each gate is 1, the inter-cluster delay is 2, and each cluster can accommodate three gates, after clustering the combinational logic optimally, we obtain the solution shown in Fig. 3(b). This solution has a clock period of 8. Retiming can only reduce the clock period to 7, due to the seven units of delay on the combinational path from i to o.

On the other hand, consider the clustering solution shown in Fig. 4(a), which, by the way, is NOT an optimal clustering of the combinational logic. This solution, however has a clock period of 3, when retiming is applied to it, as shown in Fig. 4(b).

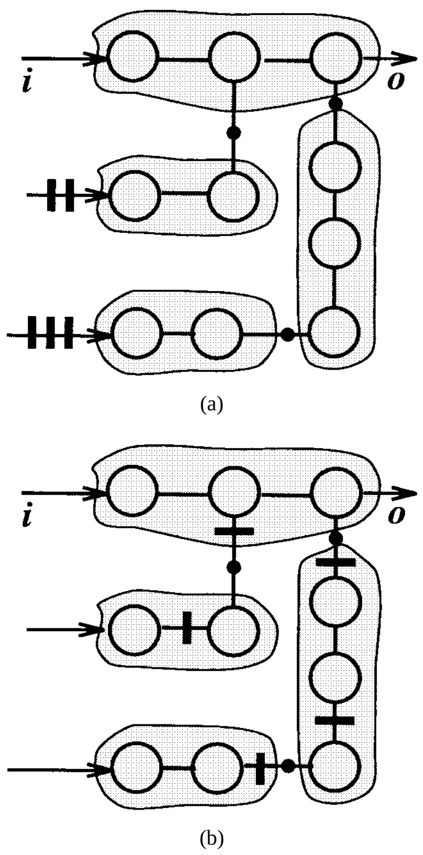


Fig. 4. Clustering while considering the effect of FF's.

Thus, the shortfalls of the conventional approach, when applied to sequential circuits, are obvious. As can be seen from these examples, if a circuit is segmented by removing the FF's, or if the effect of FF's is ignored while clustering, an optimal solution may not result.

Observing the shortcomings of the conventional approach, we propose a sequential clustering approach which does not segment a circuit by removing FF's. The approach also takes into account the effect of FF's when forming clusters to avoid cutting critical sequential paths. We further present an efficient clustering algorithm for the new approach. The clustering solution produced by the algorithm has the minimum clock period under the unit delay model, and a clock period provably close to minimum under the general delay model.

We point out that there are efforts to integrate retiming into the classical bipartitioning problem [26], [27]. The resulting problem is obviously NP-hard. Our problem is different in nature. In fact, under the assumption that delay values are constants, the algorithm proposed in this paper for the clustering problem has polynomial time complexity.

The rest of this paper is organized as follows. Section II contains some preliminaries and the problem statement. In Section III, we introduce the strong clustering problem which is closely related to the clustering problem. In fact, we will solve the strong clustering problem to obtain a solution to the clustering problem. Our algorithm to the strong clustering

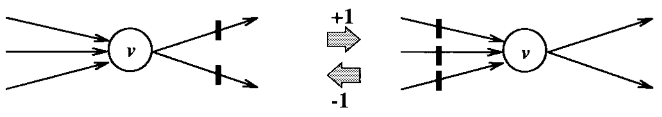


Fig. 5. Retiming a node.

problem is described in details in Section IV. We present some experimental results in Section V, and conclude with a few remarks and future research directions in Section VI.

II. Preliminaries and Problem Formulation

We represent a circuit as a directed graph. A node in the graph represents either a primary input (PI), a primary output (PO) or a gate, and an edge $u \overset{e}{\rightarrow} v$ represents an interconnection from node u to node v, in the circuit. An edge e has a weight, $w(e)$, which is the number of FF's on the interconnection. A node v has an area and a delay associated with it. In this paper, we assume delay values are integers. (If they are not, we can perform scaling and rounding to obtain integer values arbitrarily close to them.) The clock period of a circuit is the maximum total delay on the combinational paths (paths without FF's) in the circuit.

Retiming is a technique of repositioning the FF's in a circuit without altering its functionality. Retiming a node by a value i is an operation of removing i FF's from each fanout edge, and in the same time adding i FF's to each fanin edge of the node. Fig. 5 shows the cases in which $i = 1$ and -1. In general, all nodes except the PI's and the PO's can be retimed collectively to arrive at a retiming of the circuit.

A retiming r can be represented as a mapping from the nodes to integers where $r(v)$ is the retiming value for node v. Note that if v is a PI or PO, $r(v) = 0$. After applying retiming r, the number of FF's on edge $u \overset{e}{\rightarrow} v$ becomes $w(e) + r(v) - r(u)$.

Given a sequential circuit, a clustered circuit is an equivalent circuit formed by *legal clusters.* A (legal) cluster is a subcircuit with a total area at most M which is a given parameter. The clustered circuit may contain more than one copy of a node in the original circuit and the same is true for edges. In other words, logic may be replicated when forming the clustered circuit. Retiming is also allowed during clustering to exploit the whole solution space. It should be noted that a node in a clustered circuit may differ from the node in the original circuit in clock cycles due to retiming.

In a clustered circuit, an edge from a node outside a cluster to a node inside the cluster incurs an inter-cluster delay D which, as M, is also a given parameter. *The clustering problem addressed in this paper is to find a clustered circuit with the minimum clock period.* We assume each PI or PO forms a singleton cluster of its own and no inter-cluster delay is incurred on edges involving its cluster.

In the description above, the underlying delay model is usually called the *general delay model* as the delay values of the gates can be arbitrary. When all gate delays are zero and $D = 1$, we have the so-called *unit delay model* [19].

In practice, a target clock period is usually given and the objective is to find a clustered circuit with the target clock period. This leads to the decision version of the clustering problem:

Problem 1: Given a sequential circuit, and a target clock period ϕ, find a clustered circuit with a clock period of ϕ, if such a circuit exists.

With a method to solve the decision problem, we can easily find a clustered circuit with the minimum clock period if it is so desired, by carrying out binary search on the target clock period within estimated lower and upper bounds. For example, the optimal clock period of the original circuit is a lower bound, and the optimal clock period plus nD, where n is the number of nodes in the original circuit, is an upper bound as there are at most n clusters in the clustered circuit.

To consider the effect of retiming during clustering, we generalize the *concept of l-values* introduced in [28] and use l-values as an indirect way to consider retiming. For this we introduce another set of weights on the edges in the circuit (graph). For each edge $u \overset{e}{\rightarrow} v$, the second weight $w_1(e) = -\phi w(e) + d(v)$, where $w(e)$ is the number of FF's on e, and $d(v)$ is the delay of v. *The l-value of a node is defined as the weight of the longest paths from the PI's to the node using w_1 weight.* One useful property of l-values is that the l-values of the PO's are an invariant with respect to retiming, as stated in the following result:

Lemma 1: If a circuit is obtained from another one by retiming, then the l-values of the corresponding PO's in both circuits are the same.

Proof: Actually the following stronger statement holds: Suppose circuit N_2 is obtained by applying retiming r to N_1. For any path p from a node x to a node y, the difference of the w_1 weights of p in N_2 and N_1 is $\phi(r(x) - r(y))$.

For each edge $u \overset{e}{\rightarrow} v$, its w_1 weight in N_2 is $-\phi w_r(e) + d(v) = -\phi w(e) + d(v) + \phi(r(u) - r(v))$. With this formula, the above statement is obvious. Since $r(v) = 0$ if v is a PI or PO, the lemma follows from the statement. □

Finally, we list a few additional notations. We will use N to denote the circuit to be clustered throughout the rest of this paper. A node is an input to a cluster if the node is not in the cluster, but is connected to at least one node in the cluster. For each node u in N, we use $\delta(u)$ to denote the delay incurred on all edges starting at u and going into the same cluster, so it is zero if u is a PI or the cluster is a PO, otherwise it is D. For any two nodes u and v in N, we use $\Delta(u, v)$ to denote the weight of the longest paths from u to v using w_1 weights.

III. Transforming the Clustering Problem

Dealing directly with the clock period during clustering proves to be difficult since we do not have the clustered circuit before clustering is complete. In this section, we introduce the so-called *strong clustering problem* whose objective is stated in terms of l-values. Our approach to the clustering problem is to solve the strong clustering problem. We also identify a subset of clustered circuits such that it suffices for us to consider only clustered circuits in the subset to find a solution to the strong clustering problem.

First we state an important property of l-values whose proof is given in the appendix.

Theorem 1: In a sequential circuit, if there is a PO whose l-value is greater than ϕ, the circuit cannot be retimed to a clock period of ϕ. If, on the other hand, the l-values of all PO's are less than or equal to ϕ, the circuit can be retimed to a clock period less than $\phi + K$ where K is the maximum gate delay in the circuit.

This result shows a close tie between retiming and l-values. In fact, l-values can be viewed as a sequential version of arrival times commonly used in combinational circuits. To check whether the target clock period ϕ is achievable, we can simply examine the l-values of the PO's. This leads us to the following problem:

Problem 2 (Strong Clustering Problem): Given a positive integer ϕ, find a clustered circuit such that the l-values of the PO's in the clustered circuit are less than or equal to ϕ.

If there is no solution to the strong clustering problem, from Theorem 1 there is no clustered circuit with a clock period of ϕ. On the other hand, if a clustered circuit is a solution to the strong clustering problem, the circuit can be retimed to a clock period less than ϕ plus the maximum of D and the largest gate delay in N. In the special case of unit-delay model, the clustered circuit can actually be retimed to a clock period of ϕ. Therefore, instead of studying Problem 1, we study the strong clustering problem. We will present an efficient algorithm to the strong clustering problem.

In solving the strong clustering problem, we restrict the search space while making sure at least one of the solutions to the problem is in the restricted space. We focus on clustered circuits that satisfy the following four conditions:

1) *Each cluster has one output.* If there is a cluster with more than one output, we can replicate the cluster enough times so that each copy of the cluster has only one output. This transformation does not change the l-values of any node in the clustered circuit. It, of course, may increase the number of clusters and the amount of logic replication. We can overcome this problem by a post-processing step to merge the clusters. We will refer to a cluster with the output coming from node v, *a cluster at* v.

2) *The retiming value at each node in the clustered circuit is zero.* That is, we only consider the clustered circuits that do not involve retiming. This requirement does not exclude all solutions to the strong clustering problem. Consider an arbitrary clustered circuit. If there is a node in the clustered circuit with a retiming value $t \neq 0$, we retime the node one more time by a value equal to $-t$. This additional retiming step cancels out the existing retiming at the node and makes the net retiming value at the node zero. From Lemma 1, the l-values of the PO's of the clustered circuit are kept the same after the additional retiming. Therefore, if the initial clustered circuit is a solution to the strong clustering problem, so is the retimed one. Thus, if the strong clustering problem has a solution, it has one that satisfies this requirement. Since there is no retiming at each node, in such a clustered circuit, each node is functionally the same as the corresponding node in N and each edge has the same number of FF's as the corresponding edge in N. As a result, all copies of the same node in N are functionally equivalent in the clustered circuit, and the edges fanning out of these copies can be redistributed among the copies without changing the functionality of the clustered circuit. This property will be utilized in discussing the next two requirements.

3) *For each node in N, there is at most one cluster at the node.* Suppose u in N has several clusters in the clustered circuit. We modify the clustered circuit by removing all such clusters except one with the smallest l-value at the output. Then we redirect all edges coming from these clusters so that they come from the output of the only remaining cluster at u. This modification preserves functionality as the outputs of all clusters at u are equivalent. Moreover, it does not increase the l-value of any node. Thus, if the clustered circuit is a solution to the strong clustering problem, so is the modified one.

4) *If the cluster at u is connected to nodes in the cluster at v, then the cluster at v does not contain a copy of u.* Suppose the cluster at v contains a copy of u, u'. If in the clustered circuit, the l-value of u' exceeds the l-value of u by an amount larger than or equal to $\delta(u)$, we remove u' and redirect all edges coming out of u' so that they come from u. On the other hand, if the l-value of u' is less than the l-value of u plus $\delta(u)$, we redirect all edges from u to nodes in the cluster at v so that they come from u'. In either case, none of the l-values in the clustered circuit is increased.

A clustered circuit that satisfies all the above four requirements will be referred to as a *simple clustered circuit.* Based on the above analysis, we have the following result:

Theorem 2: If the strong clustering problem has a solution, it has a simple solution. □

Because of this result, we only consider simple clustered circuits in solving the strong clustering problem. Simple clustered circuits are well-behaved and have the following useful property which may not be true for nonsimple ones:

Lemma 2: Let S be a simple clustered circuit. If C is a cluster at a node v in S, then

$$l_v = \max\{l_u + \Delta(u,v) + \delta(u) | u \text{ is an input to } C\}$$

where l_x denotes the l-value of node x in S.

Proof: Let p be a path from a PI to v in S, and u be the node in p that is an input to C. u separates p into two segments p_1 and p_2, where p_2 is the segment from u to v and p_1 is the rest. Then, $w_1(p) = w_1(p_1) + w_1(p_2) + \delta(u) \leq l_u + \Delta(u,v) + \delta(u)$. Thus, $l_v = \max_p w_1(p) \leq \max\{l_u + \Delta(u,v) + \delta(u) | u$ is an input to $C\}$.

We next show $l_v \geq l_u + \Delta(u,v) + \delta(u)$ for any input u to C. For a path from u to v in the original circuit with w_1 weight equal to $\Delta(u,v)$, consider the corresponding path in S. Since S is simple, the path must originate from u. Thus, the path must cross C. Hence, $l_v \geq l_u + \Delta(u,v) + \delta(u)$. □

IV. An Algorithm for the Strong Clustering Problem

In this section, we present an efficient algorithm for the strong clustering problem. The algorithm has two phases. In

```
LABEL(N, $\phi$)
  for each $v$ in the circuit $N$ do
    if ($v$ is a PI) then $l(v) = 0$
    else $l(v) = -\infty$
  for $i = 1$ to $B$ do /* $B$, number of iterations */
    changed = FALSE
    for each non-PI node $v$ in $N$ do
      $(l_{new}, C_{new})$ = TIGHTENBOUND($v$)
      if ($l_{new} > l(v)$) then
        $l(v) = l_{new}$
        changed = TRUE
      $C_v = C_{new}$
      if ($v$ is a PO and $l(v) > \phi$) then
        return FAILURE /* no solution */
    if (changed = FALSE) then
      return SUCCESS /* Bounds have settled */
```

Fig. 6. Outline of the labeling procedure.

the first phase, for each node in N a label and a corresponding cluster are computed. *The label we wish to compute is the minimum l-value of the node in all simple clustered circuits of* N. If the labeling phase stops successfully, the algorithm goes to the second phase to generate a solution by connecting the clusters computed in the first phase. In this section, we first describe the two phases. Then we analyze the time complexity of the algorithm and point out a few enhancements to the algorithm.

A. Computing the Labels

Due to the presence of loops, there are cyclic dependencies among the labels. Our approach to overcome this difficulty is to maintain a lower bound on the label of each node and successively tighten the lower bound.

Initially, the lower bounds of all nodes are set to $-\infty$ except for the PI's, whose lower bounds are always zero. In general, a certain number of iterations are needed for the lower bounds to reach the respective labels. We use B to denote the number of iterations. For now, we assume B is large enough so that either all lower bounds settle down or one of the PO's is found to have a lower bound larger than ϕ. We will present an estimate for B later. If the lower bound for a PO exceeds ϕ, as will be shown later, there is no solution to the strong clustering problem and the labeling procedure terminates with a return value FAILURE. Fig. 6 shows an outline of the labeling procedure, where TIGHTENBOUND is the routine that tightens or improves the lower bound for a given node. Each time TIGHTENBOUND is called, it returns the improved lower bound and an associated legal cluster.

We now introduce the procedure TIGHTENBOUND. To make sure the lower bounds approach the labels from below, we increase each lower bound only by an amount that is absolutely necessary, based on the current lower bounds for all nodes. Namely, we minimize the new lower bound for each node. Suppose we are to improve the lower bound for v. For each node u in N, we calculate a value $l'(u)$ as follows:

$$l'(u) = l(u) + \Delta(u, v) + \delta(u) \tag{1}$$

where $l(u)$ is the current lower bound for node u.

```
TIGHTENBOUND($v$)
  for each node $u$ in $N$ do
      calculate $l'(u)$ according to Eq. 1
  sort all $l'$ values in increasing order $L_1, L_2, ..., L_t$
  $low = 1, high = t$
  while ($low \leq high$) do
      $mid = (low + high)/2$
      remove every node $u$ with $l'(u) \leq L_{mid}$
      form cluster $C_{v,L_{low}}$
      if (area of $C_{v,L_{low}} > M$) then
        $low = mid + 1$
      else
        $high = mid$
  return ($L_{low}, C_{v,L_{low}}$)
```

Fig. 7. Computing the improved lower bound.

If u is an input to a cluster at v, then the new bound for v is at least $l'(u)$, for this is the case in a simple clustered circuit (Lemma 2). Remember that we want to minimize the new lower bound. We set the new lower bound to be the value given by the following formula:

$$\min_{C \text{ a cluster at } v} \{\max\{l'(u) | u \text{ is an input to } C\}\}.$$

Obviously, the new bound is equal to one of the l' values. To find this l' value, we sort the l' values of the nodes in N into a list and do a binary search on the list. The main step is, then, solving the following problem:

Problem 3: Given an integer L, determine whether there is a legal cluster at v such that the l' value of each input to the cluster is at most L.

To solve Problem 3, we form a cluster $C_{v,L}$ at v as follows: All nodes having an l' value less than or equal to L are removed from N and $C_{v,L}$ is composed of all nodes that still have a path to v. Intuitively, $C_{v,L}$ consists of all gates that must be inside a cluster at v since they have a path to v that cannot cross clusters.

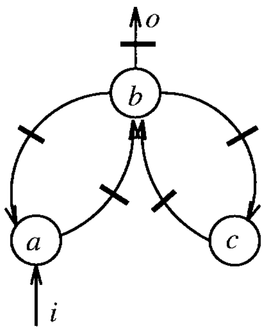
Lemma 3: Problem 3 has a positive answer iff $C_{v,L}$ is a legal cluster and does not contain a PI.

Proof: $\Leftarrow$ By construction, each input to $C_{v,L}$ is a removed node and has a l' value less than or equal to L. Since it is a legal cluster, $C_{v,L}$ is a solution to Problem 3.

$\Rightarrow$ It suffices to show that any cluster C that is a solution to Problem 3 must contain $C_{v,L}$. We prove this by contradiction. Suppose u is in $C_{v,L}$ but not in C. Since u is in $C_{v,L}$, there exists a path from u to v such that every node in p has a l' value larger than L. p must cross C for otherwise u would be in C. This means C has an input with an l' value larger than L, which contradicts the assumption that C is a solution to Problem 3. □

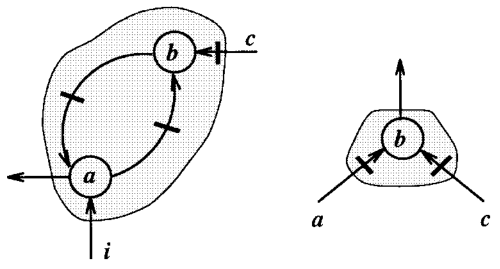
Based on Lemma 3 we can carry out binary search on the list of l' values to determine the improved lower bound for v. Fig. 7 summarizes the procedure TIGHTENBOUND

We now use the circuit in Fig. 8 to illustrate the labeling procedure. For this example, the cluster area, inter-cluster delay and target clock period are assumed to be 2 units, and the area and delay of each gate are assumed to be one unit. The table in Fig. 8 shows the all-pairs matrix Δ for the circuit.



node	a	b	c
i	1	0	-1
a	0	-1	-2
b	-1	0	-1
c	-2	-1	0

Fig. 8 A circuit and its all-pairs matrix for $\phi = 2$.



(a) (b)

Fig. 9. Final clusters for the example.

Initially, $l(i) = 0, l(a) = l(b) = l(c) = l(o) = -\infty$. Suppose we tighten the lower bounds for a, b, c in this order. After the second iteration, the lower bounds become $l(i) = 0, l(a) = l(c) = 1, l(b) = 2, l(o) = 0$. Consider the third iteration of the procedure LABEL.

For node a:

$$\begin{aligned} l'(i) &= l(i) + \Delta(i,a) + \delta(i) = 0 + 1 + 0 = 1 \\ l'(b) &= l(b) + \Delta(b,a) + \delta(b) = 2 + (-1) + 2 = 3 \\ l'(c) &= l(c) + \Delta(c,a) + \delta(c) = 1 + (-2) + 2 = 1. \end{aligned}$$

The new lower bound for a is still 1. To form the corresponding cluster, both i and c are removed, so the resulting cluster is $\{a, b\}$, as shown in Fig. 9(a).

For node b:

$$\begin{aligned} l'(i) &= l(i) + \Delta(i,b) + \delta(i) = 0 + 0 + 0 = 0 \\ l'(a) &= l(a) + \Delta(a,b) + \delta(a) = 1 + (-1) + 2 = 2 \\ l'(c) &= l(c) + \Delta(c,b) + \delta(c) = 1 + (-1) + 2 = 2. \end{aligned}$$

The new lower bound for b is still 2, and the corresponding cluster is $\{b\}$ formed by removing i, a and c, as shown in Fig. 9(b).

For node c:

$$\begin{aligned} l'(i) &= l(i) + \Delta(i,c) + \delta(i) = 0 + (-1) + 0 = -1 \\ l'(a) &= l(a) + \Delta(a,c) + \delta(a) = 1 + (-2) + 2 = 1 \\ l'(b) &= l(b) + \Delta(b,c) + \delta(b) = 2 + (-1) + 2 = 3. \end{aligned}$$

The new lower bound for c is still 1, and the corresponding cluster is $\{c, b\}$ formed by removing i and a, as shown in Fig. 9(c).

For the PO $o, l(o) = l(b) - \phi w(b, o) = 0$. (Note that since each PO forms a cluster by itself, its lower bound is always that of the gate that generates the output, plus the w_1 weight of the edge leading to the PO.) Since no change in the bounds is encountered, the labeling procedure comes to a halt.

Lemma 4: (1) For any node $v, l(v)$ is nondecreasing from iteration to iteration. (2) After the labeling procedure stops with SUCCESS, $l(v) = \max\{l(u) + \Delta(u,v) + \delta(u) | u$ is an input to $C_v\}$.

Proof: Using induction on the number of calls to TIGHTENBOUND, it is easy to show $l(v) \leq l_{\text{new}}$, which obviously implies statement (1).

Suppose the labeling procedure stops with SUCCESS. Consider the last iteration in the procedure. For each node v, since $l(v) \leq l_{\text{new}}$ and changed = FALSE, we have, $l(v) = l_{\text{new}}$. Right after the call to TIGHTENBOUND$(v), l_{\text{new}} = \max\{l(u) + \Delta(u,v) + \delta(u) | u$ is an input to $C_v\} = l(v)$. Since no changes in any lower bounds thereafter, this relation holds when the procedure stops. □

Lemma 5: Let l_v be the l-value of a node v in a simple clustered circuit. Then, $l(v) \leq l_v$.

Proof: We prove the statement by induction on the number of times TIGHTENBOUND has been called. At the beginning of the algorithm, with the initial values of the lower bounds the statement is obviously true. Suppose it is true right before a call to TIGHTENBOUND(v). After the call only the lower bound for v may be changed, so all we need is to show the new lower bound for v is still less than or equal to l_v.

Let C_1 be the cluster at v in the simple clustered circuit. From Lemma 2, we have $l_v = \max\{l_u + \Delta(u,v) + \delta(u) | u$ is an input to $C_1\}$. Thus

$$\begin{aligned} l_{\text{new}} = & \min_C \{\max\{l(u) + \Delta(u,v) \\ & + \delta(u) | u \text{ is an input to } C\}\} \\ \leq & \max\{l(u) + \Delta(u,v) + \delta(u) | u \text{ is an input to } C_1\} \\ \leq & \max\{l_u + \Delta(u,v) + \delta(u) | u \text{ is an input to } C_1\} \\ = & l_v. \end{aligned}$$

□

The labeling procedure returns FAILURE if there is a PO whose lower bound exceeds ϕ, which according to Lemma 5, implies that the l-value of that PO in any simple clustered circuit exceeds ϕ. This means the strong clustering problem has no solution. Thus, we have the following result:

Lemma 6: If the labeling procedure returns FAILURE, the strong clustering problem has no solution. □

B. Generating a Clustered Circuit

After the successful completion of the labeling procedure, we also have a cluster for each node, in addition to the label, i.e., the final lower bound. In this phase of the algorithm, we construct a solution for the strong clustering problem. This is done by connecting the clusters together. If node u is an input to the cluster at node v, the output of the cluster at u is connected to each node that u is supposed to be connected to in the cluster at v, and the number of FF's on each connection is kept the same as that on the corresponding connection in N. In general, some clusters are redundant in that their outputs have no path to the PO's. After all connections are made, we remove the redundant clusters by tracing the clustered circuit backward

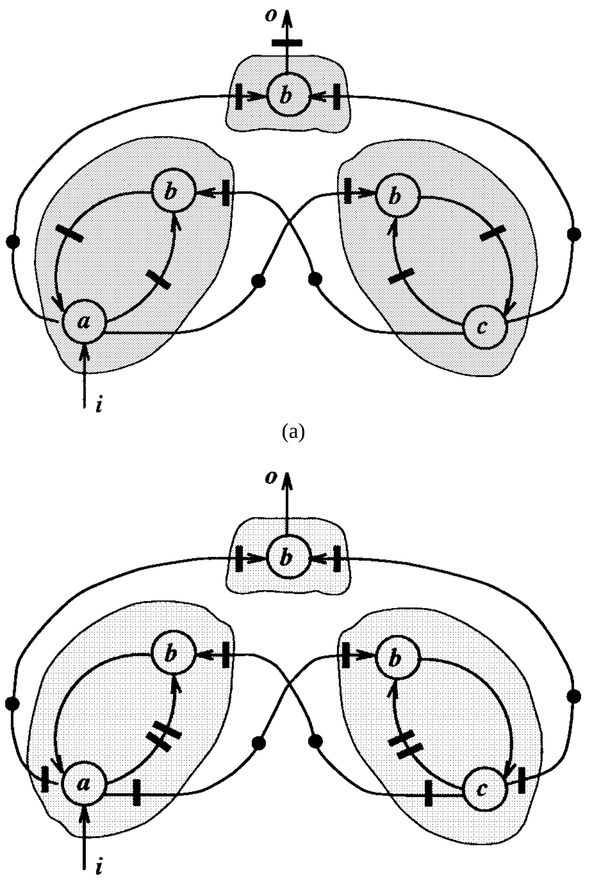


Fig. 10. (a) Initial clustered circuit; (b) retimed one with a clock period of 2.

from the PO's, and deleting all clusters not encountered. Let the resulting circuit be S. For example, Fig. 10(a) shows the solution for the circuit in Fig. 8 constructed using the clusters in Fig. 9.

Lemma 7: If v is the output of a cluster in S, the l-value of v in S is less than or equal to the final lower bound for v.

Proof: Let $l(v)$ be the final lower bound for v, and C_v be the corresponding cluster at v. It suffices to show $l(v)$ is larger than or equal to $w_1(p)$, the total w_1 weight of p, for any path p from a PI to v in S. We prove this by induction on the number of cut edges on p.

For the base case that p contains no cut edge, v must be a PI (since we assume each PI forms a cluster of its own), so the statement is trivially true.

Now assume the statement is true for any path with $k-1$ cut edges. Suppose there are k cut edges on p. Let u be the node in p that is an input to the cluster at v, C_v. u separates p into two segments p_1 and p_2, where p_2 is the segment from u to v and p_1 is the rest. Then, $w_1(p) = w_1(p_1) + w_1(p_2) + \delta(u) \leq w_1(p_1) + \Delta(u,v) + \delta(u)$. From the induction assumption, $w_1(p_1) \leq l(u)$. Therefore,

$$\begin{aligned} l(v) &= \max\{l(u_1) + \Delta(u_1, v) + \delta(u_1) | u_1 \text{ is an input to} C_v\} \\ &\quad \text{(Lemma 4)} \\ &\geq l(u) + \Delta(u,v) + \delta(u) \\ &\geq w_1(p_1) + \Delta(u,v) + \delta(u) \geq w_1(p). \end{aligned}$$

□

Note that the final lower bound for each PO is less than or equal to ϕ. Hence the l-value of each PO in S is less than or equal to ϕ according to Lemma 7. Combining this with Lemma 6, we have have the following result which asserts the correctness of our algorithm.

Theorem 3: If the labeling procedure returns FAILURE, there is no solution to the strong clustering problem. If, on the other hand, the procedure return SUCCESS, S is a solution to the strong clustering problem. □

To obtain a solution to Problem 1, we optimally retime S. Let S_r denote the retimed circuit. Combining Theorems 1 and 3, we have the following result:

Theorem 4: If the labeling procedure returns FAILURE, there is no clustered circuit with a clock period of ϕ. On the other hand, if the procedure return SUCCESS, S_r has a clock period less than $\phi + K$, where K is the maximum of the gate delays and D. □

For the unit delay model, where $D = 1$, and each gate delay is zero [20], we have that the clock period of S_r is less than $\phi + 1$, which implies the clock period is less than or equal to ϕ, since ϕ is an integer. Thus,

Corollary 1: For the unit delay model, S_r has a clock period less than or equal to ϕ if the labeling procedure return SUCCESS. □

For the clustered circuit in Fig. 10(a), after retiming we obtain the clustered circuit in Fig. 10(b), which meets the target clock period 2.

C. Time Complexity and Improvements

In this section, we determine the time complexity of the labeling procedure as it is the most time consuming step in the algorithm. In the following, we will use n and m to denote the number of nodes and number of edges in N, respectively.

We first determine B the number of iterations needed in the labeling procedure. For this, we modify the labeling procedure by setting the initial lower bound for each node to its l-value in N, since the l-value of any node in any simple clustered circuit is obviously larger than or equal to the l-value of the node in N.

Consider the clustered circuit S generated by our algorithm. Any simple path can cross clusters at most $n - 1$ times as S has at most n clusters. As a result, the l-value of any node in S is larger than the l-value of the node in N by at most $(n - 1)D$. During each iteration before the labeling procedure stops, at least one of the lower bounds is increased. After at most $n(n - 1)D$ iterations, all lower bounds must reach their maximum possible values. Thus, we can set B to $n(n - 1)D$ in the labeling procedure. If there are still unsettled lower bounds after $n(n - 1)D$ iterations, S does not exist, which means the target clock period is not achievable. In this case, the procedure simply stops by returning FAILURE.

Given ϕ, the l-values of the nodes in N can be determined in time $O(nm)$ using Bellman-Ford algorithm, and the all-pairs matrix Δ can be calculated in time $O(n^2 \log n + nm)$ [29]. (Note that longest paths become shortest paths if we negate the edge weights.) In TIGHTENBOUND, searching the

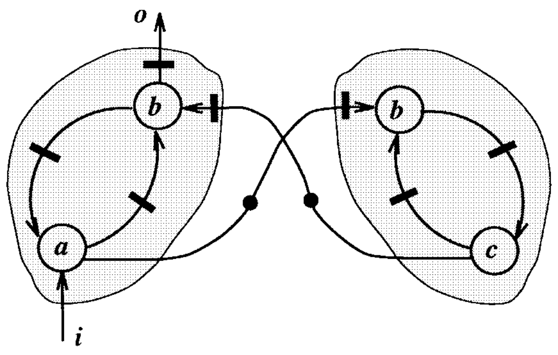


Fig. 11. Cluster reduction.

list of l' values and determining the corresponding cluster take time $O((n+m)\log n)$. TIGHTENBOUND is called at most B times at each node. Hence the time cost of the labeling procedure is $O(n^3 mD\log n)$ in the worst case. This worst-case scenario is unlikely to occur in practice as the estimate of B is obviously very loose. We also have several methods to improve the running time, as discussed next.

One area for improvement is in TIGHTENBOUND, which tightens the lower bound for v. Noticing that only those nodes that have a path to v can be in the cluster $C_{v,L}$, we only consider those nodes in determining the improved bound.

If TIGHTENBOUND is called with a particular order of the nodes, the number of iterations can be reduced significantly. An effective ordering is obtained as follows. A set of nodes are selected to break all cycles in N. A topological order is obtained for the remaining nodes. We then append the removed nodes at the end of the topological order to generate an order of all nodes in N. The intuition here is to tighten the lower bounds of the fanins of a node before the node itself. Obviously, this is not possible for all nodes in the circuit if there are feedback loops. The proposed order makes sure that this is the case for most of the nodes. In our experiment, we observed that either the lower bounds settle down, or a PO with a lower bound larger than ϕ is found in a very small number of iterations when TIGHTENBOUND is called in this order.

D. Cluster and Replication Reduction

In the previous discussion, we assume each cluster has one output. If this assumption is relaxed, a post-processing step can be added to reduce the number of clusters, without increasing the l-values of the PO's. For this, techniques similar to those in [19] can be used. For example, if the l-value of a node v in its own cluster is equal to the l-value of a copy of the node in another cluster, the entire cluster at v can be removed, and replaced by the copy. As an example, for the circuit in Fig. 10(a), the l-value of the b in the cluster at a is 1, the same as the l-value of the b in its own cluster. Hence, we can remove the cluster at b and replace it with the b in the cluster at a. After the reduction, we obtain the clustered circuit in Fig. 11. This circuit can also be retimed to the target clock period 2.

Replicated nodes can also be reduced, as follows. If the l-values of two copies of the node differ by an amount greater than or equal to the inter-cluster delay D, then the output of the copy with the smaller l-value can replace the copy with the larger l-value. This does not increase the l-values of any nodes. Also, we have observed that there are only a few "critical paths" that make one or more of the PO's with l-values close or equal to ϕ. For nodes on noncritical paths, there are slacks available and their l-values need not be minimum. This property can also be used to further remove replicated nodes while keeping the l-values of the PO's to be less than or equal to ϕ. Once this reduction of replicated nodes is carried out, there may be clusters that are not completely filled. We can merge some of the clusters, provided that the area bound is not exceeded.

Several techniques were proposed to guide the cluster merging for combinational circuits in [19]. Similar ideas apply here except that l-values are used in place of combinational path delays. We follow a two-stage strategy. In the first stage, we search for clusters which contain copies of the same node. If these clusters are merged, all such copies in the resulting cluster could potentially be replaced by the copy with the smallest l-value. Note that the previous condition that require the l-values to differ by an amount greater than or equal to the inter-cluster delay can now be relaxed, since the nodes being replaced are now in the same cluster. Two clusters are mergeable if their combined area (after the replicated copies have been removed) does not exceed the area bound.

In the next stage, the remaining clusters are simply merged such that the area bound is not violated. When there are choices, we select "strongly connected" clusters to merge. The reason for doing this is as follows. Merging clusters that have edges between them results in removing inter-cluster delays between the clusters. This could potentially reduce the l-values of all nodes downstream of these nodes. It may now be possible to replace nodes that previously could not be replaced. Hence, we follow this merging step by recalculating the l-values of the nodes. We then iterate between replacing copies and merging clusters till no further reduction is possible.

V. Experimental Results

We implemented the clustering algorithm proposed in this paper, and carried out some experiments. In this section, we describe our experiments and summarize the results.

The test examples come from the sequential circuits in the ISCAS 89 suite. In our experiments, we set the area and delay of each gate to one unit and the inter-cluster delay to two units.

For each circuit, we tested three area bounds: M is 5, 10, and 20% of the number of gates. The results obtained are summarized in Table I, where column ϕ lists the minimum clock period, column $\#g$ lists the number of gates, and column $\#c$ lists the number of clusters in the clustered circuit. The final column lists the largest CPU time among the three area bounds for a given target clock period on an UltraSPARC 2 workstation.

From the table it can be seen that overall our algorithm obtains optimal clock period clustering solutions with reasonable area overhead. As the cluster size increases, there is more freedom in assigning gates to clusters. The results in the table

TABLE I
EXPERIMENTAL RESULTS

test circuit			5%			10%			20%			CPU
name	ϕ	#g	ϕ	#g	#c	ϕ	#g	#c	ϕ	#g	#c	time (s)
s208.1	10	104	13	107	22	11	109	12	10	106	6	0.05
s349	14	161	18	162	21	17	163	11	15	163	6	1.48
s420.1	12	218	14	228	25	13	227	11	12	219	6	0.25
s635	66	286	75	298	22	70	286	11	68	286	6	0.76
s838.1	16	446	17	455	21	16	451	11	16	446	6	1.06
s1196	24	529	26	558	22	25	570	12	24	572	6	0.65
s1423	53	657	55	673	22	53	657	11	53	657	6	5.20
s1512	23	780	24	839	24	23	826	13	23	780	6	9.75
s3330	14	1789	15	1795	21	14	1874	11	14	1839	6	8.52
s4863	30	2342	31	2351	21	30	2348	11	30	2352	6	658.37
s5378	21	2779	21	3168	24	21	3164	12	21	3112	6	60.62

show that our algorithm can automatically detect and utilize this freedom, to produce clustered circuits with decreasing clock periods.

When the area bound is 10% of the number of gates, the clock period of the clustered circuit obtained by our algorithm is very close to the optimal clock period of the original circuit, which can also be viewed as a clustered circuit with M equal to the number of gates. This shows the effectiveness of our algorithm in avoiding cutting sequential critical paths as 10% is still a small fraction of the size of the circuit.

VI. CONCLUSIONS

Circuit clustering is an important step in the design of VLSI circuits. The solution formed at this step greatly influences the quality of the final design. We have developed a clustering algorithm for sequential circuits that incorporates retiming. The clock period of the solution obtained by our algorithm is optimal for the unit delay model, and provably close to optimal for the general delay model.

We are currently working to improve the algorithm by further reducing the running time and the replicated logic. We are also looking into the clustering problem with other constraints such as pin count.

Another direction for further research concerns classical circuit partitioning. If the effect on the clock period is not taken into account during partitioning, a critical path may be cut a large number of times. We are investigating ways to balance clock period and cut-size.

APPENDIX
PROOF OF THEOREM 1

Let G denote the circuit. We use $w(p), w_1(p)$, and $d(p)$ to denote the w weight, w_1 weight, and the total delay of a path p in G, respectively. Let l_v denote the l-value of node v in G.

(*if*) Suppose there is a PO v such that $l_v > \phi$. Let p be a path from a PI u to v such that $w_1(p) = l_v$. Then, $w_1(p) = -\phi \cdot w(p) + d(p) > \phi$, or $d(p) > (w(p)+1)\phi$. To have a clock period of ϕ on p, there must be $\lceil d(p)/\phi \rceil - 1 \geq w(p)+1$ FF's on p, by pigeonhole principle. This is obviously impossible since retiming cannot change the number of FF's on any path from a PI to a PO.

(*only if*) Let r be the following retiming:

$$r(v) = \begin{cases} 0, & v \text{ is a PI or a PO} \\ \left\lceil \dfrac{l_v}{\phi} \right\rceil - 1, & \text{otherwise} \end{cases}$$

and G_r denote the circuit obtained by retiming G according to r. We claim G_r has a clock period of ϕ.

First, we verify that r is a legal retiming. That is, the number of FF's on any edge in the retimed circuit is nonnegative. Consider an edge $u \overset{e}{\rightarrow} v$. We have, $l_v \geq l_u + w_1(e) = l_u - \phi w(e) + d(v)$. There are four cases depending on whether u and v are gates, PO's, or PI's. All the case can be proved similarly. Here we show the case that u is a gate and v is a PO. In this case, we have $\phi \geq l_v \geq l_u - \phi w(e) + d(v)$, so $\phi \geq l_u - \phi w(e)$. Dividing both sides by ϕ and taking the ceiling, we have $1 = r(v)(=0) + 1 \geq \lceil l_u/\phi \rceil - w(e) = r(u) + 1 - w(e)$. After rewriting, we have the number of FF's on e after the retiming is $w(e) + r(v) - r(u) \geq 0$.

To show the retimed has a clock period less than $\phi + K$, we need to show that for any path p such that $d(p) \geq \phi + K$, there is at least one FF on it. Let the first and last nodes of p be u and v, respectively. Then,

$$l_v \geq l_u + w_1(p) = l_u - \phi w(p) + d(p) - d(u).$$

Dividing both sides by ϕ and taking the ceiling, we have,

$$\begin{aligned} r(v) + 1 \geq\ & r(u) + 1 - w(p) \\ & + \left\lceil \frac{d(p) - d(u)}{\phi} \right\rceil \geq r(u) + 1 - (w(p) - 1). \end{aligned}$$

Thus, the number of FF's on p in $G_r, w_r(p) = w(p) + r(v) - r(u) \geq 1$. □

REFERENCES

[1] N. Sherwani, *Algorithms for VLSI Physical Design Automation.* Norwell, MA: Kluwer, 1995.

[2] C. J. Alpert and A. B. Kahng, "Recent directions in netlist partitioning: A survey," *Integr., VLSI J.*, vol. 19 pp. 1–81, 1995.

[3] F. M. Johannes, "Partitioning of VLSI circuits and systems," in *ACM/IEEE Design Automation Conf. (DAC)*, 1996, pp. 83–87.

[4] C. J. Alpert and S.-Z. Yao, "Spectral partitioning: The more eigenvectors the better," in *ACM/IEEE Design Automation Conf. (DAC)*, 1995, pp. 195–200.

[5] J. Cong, Z. Li, and R. Bagrodia, "Acyclic multi-way partitioning of boolean networks," in *ACM/IEEE Design Automation Conf. (DAC)*, 1994, pp. 670–675.

[6] P. K. Chan, M. D. F. Schlag, and J. Y. Zien, "Spectral based multi-way FPGA partitioning," in *Int. Symp. Field-Programmable Gate Arrays*, 1995, pp. 133–139.

[7] D. Cheng, C.-C. Lin, and M. Marek-Sadowska, "Circuit partitioning with logic perturbation," in *Int. Conf. on Computer-Aided Design (ICCAD)*, 1995, pp. 650–655.

[8] S. Dutt and W. Deng, "A probability-based approach to VLSI circuit partitioning," in *ACM/IEEE Design Automation Conf. (DAC)*, 1996, pp. 100–105.

[9] C. Fiduccia and R. Matheyses, "A linear-time heuristic for improving network partitions," in *ACM/IEEE Design Automation Conf. (DAC)*, 1982, pp. 175–181.

[10] L. W. Hagen, D. J.-H. Huang, and A. B. Kahng, "On implementation choices for iterative improvement partitioning algorithms," in *European Design Automation Conf.*, 1995, pp. 144–149.

[11] S. Hauck and G. Borriello, "An evaluation of bipartitioning techniques," in *Chapel Hill Conf. Advanced Research VLSI*, 1995, pp. 383–402.

[12] L. J. Hwang and A. E. Gamal, "Min-cut replication in partitioned networks," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 96–106, 1995.
[13] C. Kring and A. Newton, "A cell-replicating approach to min-cut-based circuit partitioning," in *Int. Conf. Computer-Aided Design (ICCAD)*, 1991, pp. 2–5.
[14] K. Roy-Neogi and C. Sechen, "Partitioning with performance optimization," in *Int. Symp. Field-Programmable Gate Arrays*, 1995, pp. 146–152.
[15] M. Shih and E. S. Kuh, "Circuit partitioning under capacity and I/O constraints," in *Custom Integrated Circuits Conf.*, 1994, pp. 659–662.
[16] H. Vaishnav and M. Pedram, "Delay optimal partitioning targeting low power VLSI circuits," in *Int. Conf. Computer-Aided Design (ICCAD)*, 1995, pp. 638–643.
[17] H. Yang and D. Wong, "New algorithms for min-cut replication in partitioned circuits," in *Int. Conf. Computer-Aided Design (ICCAD)*, 1995, pp. 216–223.
[18] C.-W. Yeh, C.-K. Cheng, and T.-T. Y. Lin, "Circuit clustering using a stochastic flow injection method," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 154–162, 1995.
[19] R. Murgai, R. Brayton, and A. Sangiovanni-Vincentelli, "On clustering for minimum delay/area," in *Int. Conf. Computer-Aided Design (ICCAD)*, 1991, pp. 6–9.
[20] E. Lawler, K. Levitt, and J. Turner, "Module clustering to minimize delay in digital networks," *IEEE Trans. Comput.*, vol. 18, pp. 47–57, 1969.
[21] R. Rajaraman and D. Wong, "Optimal clustering for delay minimization," in *ACM/IEEE Design Automation Conf. (DAC)*, 1993, pp. 309–314.
[22] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, pp. 5–35, 1991.
[23] N. V. Shenoy and R. Rudell, "Efficient implementation of retiming," in *Int. Conf. Computer-Aided Design (ICCAD)*, 1994, pp. 226–233.
[24] R. B. Deokar and S. S. Sapatnekar, "A fresh look at retiming via clock skew optimization," in *ACM/IEEE Design Automation Conf. (DAC)*, 1995, pp. 310–315.
[25] P. Pan, "Continuous retiming: algorithms and applications," in *Int. Conf. on Computer Design (ICCD)*, 1997, pp. 116–121.
[26] L.-T. Liu, M. Shih, N. Chou, C.-K. Cheng, and W. Ku, "Performance-driven partitioning using retiming and replication," in *Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 296–299, 1993.
[27] L.-T. Liu, M.-T. Kuo, C.-K. Cheng, and T. Hu, "Performance-driven partitioning using a replication graph approach," in *ACM/IEEE Design Automation Conf. (DAC)*, 1995, pp. 206–210.
[28] P. Pan and C. L. Liu, "Optimal clock period FPGA technology mapping for sequential circuits," in *ACM/IEEE Design Automation Conf. (DAC)*, pp. 720–725, 1996.
[29] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* New York: McGraw-Hill, 1990.

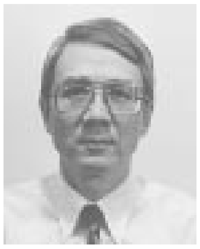
Peichen Pan received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign in 1995.

He is currently an Assistant Professor of electrical and computer engineering at Clarkson University, Potsdam, NY. From 1987 to 1990, he was a Research Associate with the Institute of Computer Science and Technology, Peking University, Beijing, China. His research interests include physical design, logic synthesis, and design and analysis of algorithms.

Dr. Pan received David J. Kuck Outstanding Doctoral Thesis Award from the University of Illinois at Urbana-Champaign in 1996.

Arvind K. Karandikar received the B.E. degree in electronics and telecommunication from the University of Poona, India, in 1994 and the M.S. degree in electrical engineering from Clarkson University, Potsdam, NY, in 1996.

He is currently with the Validation Technology Group, Intel Corporation, Folsom, CA. His research interests include test generation and genetic programming.



C. L. Liu (M'64–SM'82–F'86) received the B.Sc. degree from Cheng Kung University, Taiwan, R.O.C., in 1956, the M.S. and E.E. degrees in 1960, and the Sc.D. degree in 1962, all in electrical engineering, from the Massachusetts Institute of Technology, Cambridge.

He is currently a Professor of computer science and the President of National Tsing Hua University, Hsinchu, Taiwan. From 1972 to 1997, he was a Professor of computer science at the University of Illinois, Urbana-Champaign. His areas of research interest include design and analysis of algorithms, computer-aided design of integrated circuits, and combinatorial mathematics. He is Editor-in-Chief of the *ACM Transactions on Design Automation of Electronic Systems.*

Prof. Liu is a Fellow of the Association for Computing Machinery.