Optimal Clock Period Clustering for Sequential Circuits with Retiming*

Arvind K. Karandikar
FM5 171
Intel Corporation
Folsom, CA 95630
akarandi@pcocd2.intel.com

Peichen Pan
Dept. of ECE
Clarkson University
Potsdam, NY 13699
panp@sun.soe.clarkson.edu

C. L. Liu
Dept. of CS
University of Illinois at U.-C.
Urbana, IL 61801
liucl@cs.uiuc.edu

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 clock period. Current algorithms address combinational circuits only, and treat a sequential circuit as a special case, by removing the flip-flops (FFs) and clustering the remaining combinational logic. This approach segments a circuit and assumes the positions of the FFs are fixed. The positions of FFs are in fact dynamic, because of retiming. As a result, current algorithms can only consider a small portion of the available solution space. In this paper, we present a clustering algorithm that does not remove the FFs. It also considers the effect of retiming. The algorithm can produce clustering solutions with optimal clock periods under the unit delay model. For the general delay model, it can produce clustering solutions with clock periods provably close to minimum.

1 Introduction

Circuit partitioning/clustering is an important aspect of VLSI design [1, 2]. It consists of dividing a circuit into parts, each of which can be implemented as a separate component (e.g., a chip) that satisfies several 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 [3]-[13]. The classical objective of partitioning is to minimize the cut-size, i.e., the number of nets spanning two or more parts.

We consider circuit clustering subject to a bound on the area of each component, with possible replication of nodes, i.e., a gate may be assigned to more than one component. 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.

Previous work has mainly focused on combinational circuits for which effective heuristics and optimal algorithms have been proposed [14, 15, 16]. However, most circuits in practice are sequential. Clustering algorithms for combinational circuits can be applied to sequential circuits, by clustering the combinational logic between FFs as was done in the past [14]. In other words, the FFs in a sequential circuit are simply removed to obtain a combinational network. Then the combinational network is partitioned/clustered. Finally, the FFs are placed back, as indicated in Fig. 1.

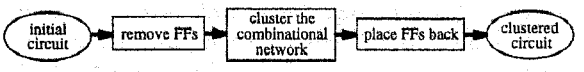


Figure 1: Conventional clustering approach.

For sequential circuits, there is a functionality-preserving transformation known as retiming [17]. Retiming allows a designer to move FFs around within a circuit. The conventional approach greatly restricts the solution space that can be explored, since it does not consider different FF configurations that can be derived using retiming, It also segments a circuit into independent pieces when removing the FFs.

Ignoring the effect of FFs 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. 2(a). Assuming the delay of each gate is 1, the inter-cluster delay is 2, and each cluster can accommodate 3 gates, after clustering the combinational portion optimally, we obtain the solution shown in Fig. 2(b). This solution has a clock period of 8. Retiming can only reduce the clock period to 7, due to the 7 units of delay on the combinational path from i to o.

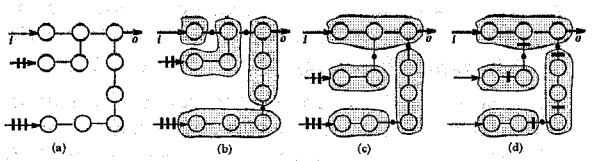


Figure 2: Clustering and retiming.

On the other hand, consider the clustering solution shown in Fig. 2(c), which is NOT an optimal clustering

*The work was partially supported by a Design Automation Scholarship Award. Research of C.L. Liu was partially supported by the NSF under grant MIP-9612184.

of the combinational logic. This solution, however has a clock period of 3, when retiming is applied to it, as shown in Fig. 2(d).

We propose a sequential clustering approach which does not break a circuit by removing FFs. The approach also takes into account the effect of retiming. We then present an efficient clustering algorithm based on the new approach. The algorithm produces a clustering solution with the minimum clock period under the unit delay model. For the general delay model, it can produce a clustering solution with a clock period provably close to minimum.

We point out that there have been efforts to combine retiming with the classical bi-partitioning problem [18, 19]. The resulting problem is obviously NP-complete. Our problem is different in nature. In fact, under the assumption that delay values are constants, the algorithm proposed in this paper has polynomial time complexity.

The rest of this paper is organized as follows. Section 2 contains the problem description. In Section 3, we introduce the strong clustering problem. We will solve this problem and then turn its solution into a solution to the clustering problem. An optimal algorithm to the strong clustering problem is described in Section 4. We present some experimental results in Section 5, and conclude this paper in Section 6.

2 Problem statement

We represent a circuit as a directed graph. A node in the graph represents either a primary input (PI), primary output (PO) or a gate, and an edge $u \xrightarrow{e} v$ represents an interconnection from node u to node v. An edge e has a weight, $w(e)$, which denotes the number of FFs on the interconnection. A node v has an area and a delay[1] associated with it. The *clock period* ϕ of a circuit is the maximum delay on the combinational paths (paths without FFs) in the circuit.

Retiming a node by a value i means removing i FFs from each fan-out edge, and in the same time adding i FFs to each fan-in edge of the node. In general, all nodes except PIs and POs can be retimed collectively (referred to as a retiming of the circuit). A retiming r can be represented by a mapping from the nodes to integers, where $r(v)$ denotes the retiming value for node v. Note that if v is a PI or PO, $r(v) = 0$. After applying retiming r, the weight of edge $u \xrightarrow{e} v$ becomes $w(e) + r(v) - r(u)$.

Given a sequential circuit, a clustered circuit (of the given circuit) is an equivalent circuit formed by *legal clusters*. A (legal) cluster is a subcircuit of the given one and has 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 initial circuit and the same is true for edges. In other words, logic may be replicated in forming the clustered circuit. Retiming is also allowed in our formulation to consider 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 a given parameter. *The clustering problem addressed in this paper is to find a clustered circuit with minimum clock period.* Without loss of generality, we assume the PIs and POs are not clustered and no inter-cluster delay is incurred on edges involving them.

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* [14].

In practice, the 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: *Given a sequential circuit, and a target clock period ϕ, find an equivalent clustered circuit with a clock period of ϕ, if such a circuit exists.* Of course, if we can 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.

To consider the effect of retiming during clustering, we generalize the important concept of *l-values* introduced in [20] and use l-value as an indirect way to consider retiming. For this we introduce a second weight $w_1(e)$ for each edge $u \xrightarrow{e} v$. $w_1(e)$ is defined to be $-\phi \cdot w(e) + d(v)$, where $d(v)$ is the propagation delay of v. *The l-value of a node is simply the maximum weight of the paths from the PIs to the node, using the w_1 weight*[2].

Finally, we list a few more definitions: We use N to denote the circuit to be clustered. For a node v in N, we use $\delta(u)$ to denote the delay incurred on edges starting at u and entering another cluster, so it is zero if u is a PI, or D if otherwise. $\Delta(u, v)$ for two nodes u and v denotes the weight of the longest path from u to v in N, using the w_1 weight.

3 The strong clustering problem

Dealing directly with clock periods during clustering proves to be difficult since the clustered circuit is not formed yet. In this section, we introduce the 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 and then turn its solution into a solution to the clustering problem.

Problem 1 (*Strong Clustering Problem*) *Given a non-negative integer ϕ, find an clustered circuit for N such that no gate is retimed in the clustered circuit and the l-values of the POs are less than or equal to ϕ.*

There is a close tie between the strong clustering problem and the the clustering problem, as stated in the following result. (All results are stated without proofs due to space limitation.)

Theorem 1 *If there is no solution to the strong clustering problem, then there is no clustered circuit with a clock period of ϕ. If, on the other hand, S is a solution to the strong clustering problem, S can be retimed to a clock period less than ϕ plus the maximum of D and the gate delays in S.*

[1] In this paper, we assume delay values are integers.

[2] If we divide the l-values by ϕ, they form a special continuous retiming proposed in [21].

As a result of Theorem 1, instead of directly solving the original clustering problem, we find a solution to the strong clustering problem, then retime that solution to obtain a solution to the clustering problem. We next present an optimal algorithm for the strong clustering problem.

4 An optimal algorithm for the strong clustering problem

The algorithm has two phases. In the first phase, for each node in N a label and a corresponding cluster are computed. *The label we want to compute is simply the minimum l-value of the node in all clustered circuits that do not involve retiming.* Consequently, if the is a PO with a label value larger than ϕ, the strong clustering problem does not have a solution and the algorithm stops. If the labels of the POs are all less than or equal to ϕ, the algorithm goes to the second phase to generate a solution to the strong clustering problem, by assembling 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.

4.1 Computing the labels

Due to the presence of loops, there are cyclic dependencies among the labels. The procedure for computing the labels maintains a lower bound on the value of each label and successively tightens the bound.

Initially, the bounds for all nodes are set to -∞ except for the PIs, whose bounds are always zero. In general, a certain number of iterations are needed. 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 POs is found to have a lower bound larger than ϕ. Later, we will present an estimate for B. If the lower bound for a PO is found exceeding ϕ, we know there is no solution to the strong clustering problem. In this case, the labeling procedure terminates with a return value FAILURE. Fig. 3 shows the overall structure of the labeling procedure, where TIGHTENBOUND is the routine that tightens the bound for a node. Each time TIGHTENBOUND is called, it returns a new lower bound and an associated legal cluster.

We now introduce the procedure TIGHTENBOUND. To make sure the lower bounds approach the labels, we minimize the new lower bound for each node. Consider a node v. For each node u in N, we calculate a value $l'(u)$ as follows :

$$l'(u) = l(u) + \Delta(u, v) + \delta(u), \qquad (1)$$

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

If u is an input to a cluster at v, then the new bound for v is at least $l'(u)$. Thus, we set the new lower bound to be the value given by the following formula:

$$\min_{C, \text{ a cluster at } v} \left(\max\{l'(u) \mid u \text{ is an input to } C\}\right). \qquad (2)$$

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

Problem 2 *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.*

```
LABEL(N, φ)
    for each v in the circuit N
        if (v is a PI) then l(v) = 0
        else l(v) = -∞
    for i = 1 to B do // B, number of iterations
        changed = FALSE
        for each v in N
            (l_new, C_new) = TIGHTENBOUND(v)
            if (v is a PO and l_new > φ) then
                return FAILURE // no solution
            if (l_new > l(v)) then
                l(v) = l_new
                changed = TRUE
            C(v) = C_new
        if (changed = FALSE) then
            return SUCCESS
```

Figure 3: The labeling procedure.

To solve Problem 2, we form a cluster $C_{v,L}$ at node 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 the same cluster.

Lemma 1 *Problem 2 has a positive answer iff $C_{v,L}$ is a legal cluster and does not contain a PI.*

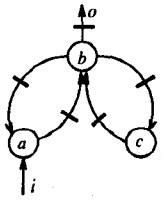
Based on Lemma 1, we can do a binary search on the list of l' values to determine the new lower bound for v. Fig. 4 summarizes the procedure TIGHTENBOUND

```
TIGHTENBOUND(v)
  calculate l'(u) for each node u according to Eq. (1)
  sort all l' values in increasing order L_1, L_2, ..., L_t
  low = 1, high = t
  while (low < high) do
    mid = (low + high)/2
    remove each node u with l'(u) ≤ L_mid
    form cluster C_{v,L}
    if (area of C_{v,L} > M or C_{v,L} contains a PI) then
      low = mid + low
    else
      high = mid
  return (L_low, C_{v,L_low})
```

Figure 4: Tighten the lower bound for one node.

We now use the circuit in Fig. 5 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 the delay of each gate are assumed to be one unit. The table in Fig. 5 shows the all-pairs matrix Δ for the circuit.

Initially, $l(i) = 0$, $l(a) = l(b) = l(c) = l(o) =$ -∞. Suppose we tighten the lower bounds for a, b, c in this order. After



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

Figure 5: A circuit and its all-pairs matrix for ϕ=2.

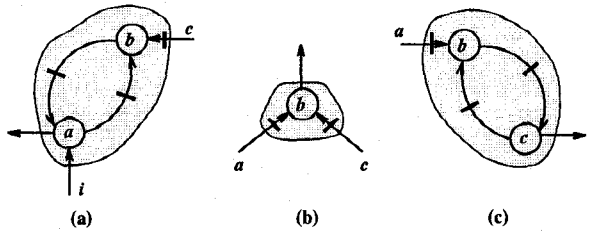


Figure 6: Final clusters.

the second iteration of the for loop in LABEL, the bounds become $l(i) = 0$, $l(a) = l(c) = 1$, $l(b) = 2$, $l(o) = 0$. Now consider the third iteration.

For node a :

$$l'(i) = l(i) + \Delta(i,a) + \delta(i) = 0 + 1 + 0 = 1;$$
$$l'(b) = l(b) + \Delta(b,a) + \delta(b) = 2 + (-1) + 2 = 3;$$
$$l'(c) = l(c) + \Delta(c,a) + \delta(c) = 1 + (-2) + 2 = 1.$$

The new bound for a is still 1, and the corresponding cluster is {a,b}, as shown in Fig. 6(a).

For node b :

$$l'(i) = l(i) + \Delta(i,b) + \delta(i) = 0 + 0 + 0 = 0;$$
$$l'(a) = l(a) + D + \Delta(a,b) + \delta(a) = 1 + (-1) + 2 = 2;$$
$$l'(c) = l(c) + D + \Delta(c,b) + \delta(c) = 1 + (-1) + 2 = 2.$$

The new bound for b is still 2, and the corresponding cluster is {b}, as shown in Fig. 6(b).

For node c :

$$l'(i) = l(i) + \Delta(i,c) + \delta(i) = 0 + (-1) + 0 = -1;$$
$$l'(a) = l(a) + \Delta(a,c) + \delta(a) = 1 + (-2) + 2 = 1;$$
$$l'(b) = l(b) + \Delta(b,c) + \delta(b) = 2 + (-1) + 2 = 3.$$

The new bound for c is still 1, and the corresponding cluster is {b,c}, as shown in Fig. 6(c).

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

4.2 Generating a clustered circuit

After the successful completion of the labeling procedure, we obtain for each node, a label and a corresponding cluster. In this phase of the algorithm, we construct a solution for the strong clustering problem. This is done by

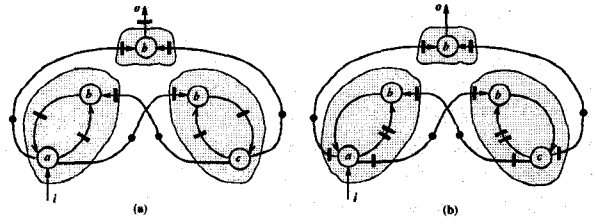


Figure 7: (a) Initial clustered circuit (b) the retimed one.

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 FFs on each connection is kept the same as that on the connection in N. In general, some clusters are redundant in that their outputs have no paths to the POs. After all connections are made, we can remove the redundant clusters by tracing the clustered circuit backwards from the POs, and deleting clusters not encountered. Let the resulting circuit be S. For example, Fig. 7(a) shows the solution for the circuit in Fig. 5 formed using the clusters in Fig. 6.

The following result states the correctness of the algorithm.

Theorem 2 *If the labeling procedure returns FAILURE, there is no solution to the strong clustering problem. If, on the other hand, the procedure returns SUCCESS, S is a solution to the problem.*

To turn S into a solution to the decision version of the original clustering problem, we calculate the l-values of the nodes in S. This can be done by running a single source longest path algorithm on S [22] (according to the definition of l-values). After the l-values are determined, we then retime S according to the following retiming:

$$r(v) = \begin{cases} 0 & \text{if } v \text{ is a PI or PO} \\ \left\lceil \frac{l\text{-value of } v \text{ in } S}{\phi} \right\rceil - 1 & \text{otherwise} \end{cases}$$

Let S_r denote the retimed circuit. We have the following result:

Theorem 3 *If the labeling procedure returns FAILURE, there is no clustered circuit with a clock period of ϕ. On the other hand, if the procedure returns 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 delay of each gate is zero [15], 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, we further have that S_r has a clock period less than or equal to ϕ if the labeling procedure returns SUCCESS.*

For the clustered circuit in Fig. 7(a), we retime all copies of node b by a value of 1 to obtain the clustered circuit in Fig. 7(b), which has a clock period of 2.

4.3 Time complexity

To determine the time complexity of the labeling procedure, we first determine the number of iterations in the labeling procedure. For this, we modify the labeling procedure by setting the initial lower bound for each node to be its l-value in N, since the label is obviously larger than or equal to the l-value. If the label of a node is finite, there are at most $n-1$ edges that cross cluster boundaries on a path with a weight equal to the label. As a result, the label is at most $(n-1)D$ larger than the initial l-value. Each iteration increases at least one of the lower bounds by at least one, for otherwise the variable *changed* is FALSE in the whole iteration and the labeling procedure terminates with SUCCESS. After at most $n(n-1)D$ iterations, all lower bounds must have reached their maximum possible values if they are finite. Thus, we can set B to be $n(n-1)D+1$ in the labeling procedure. If the labeling procedure does not return after $n(n-1)D+1$ iterations, we know at least one of the nodes has an infinite label value, and simply stop the procedure by returning FAILURE.

For a 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 $O(n^2 \log n + nm)$ [22], where n and m denote the number of nodes and the number of edges in N, respectively. In procedure TIGHTENBOUND which takes $O((n+m)\log n)$ time is called at most B times for each node. Hence the time cost of the labeling procedure is $O(n^3 mD \log n)$ in the worst case. Our experiments suggest that the worst-case scenario is unlikely to occur in practice. We also have several methods to improve the running time, which are omitted here due to space limitation.

4.4 Cluster 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 POs. For this, techniques similar to those in [14] can be used. For example, if the label of a node v is equal to the l-value a copy of the node in the clustered circuit, the entire cluster at v can be removed, and replaced by the copy. As an example, for the circuit in Fig. 7(a), the l-values of node b in both the cluster at node a and the cluster at node c are 1, the same as the label of b. As a result, we remove the cluster at node b and replace it with the b in the cluster at a (or b). The resulting clustering solution is shown in Fig. 8.

Several techniques were proposed to reduce replication for combinational circuits in [14]. Similar ideas apply here except that l-values are used in place of combinational path delays.

5 Experimental results

In this section, we describe our experiments, and summarize the results.

The test examples are from the sequential circuits in the ISCAS 89 suite. For each circuit, we ran our algorithm with three different area bounds, M=5, 10 and 15. The results obtained are summarized in Table 2, where column ϕ_{opt} lists the minimum clock period, $\#g$ lists the number of gates in the original and clustered circuits, and $\#c$ lists the number of clusters in each clustered circuit. The final column lists the average running time of our program, for the three area bounds on a SPARC 5.

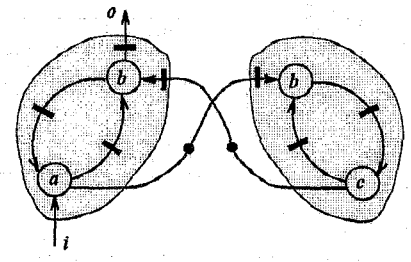


Figure 8: Clustered circuit after cluster reduction.

We set gate area and delay to be one unit and inter-cluster delay to be two units in our experiments. Thus, the clock period of each clustered circuit is at most one unit away from optimal. As the cluster size increases, there is more freedom in assigning gates to clusters. From the table it is clear that our algorithm can automatically detect and utilize this freedom, to produce clustering solutions with decreasing clock periods.

When the area bound $M = 15$, the optimal clock period obtained for the clustered circuit is very close to the optimal clock period of the original circuit. An area bound of 15 is still quite small compared to the size of the circuit. This clearly indicates the effectiveness of our algorithm.

6 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 combines retiming with clustering, and generates a clustering solution without breaking the circuit. The solution obtained has optimal clock period for the unit delay model, and provably close to optimal clock period for the general delay model.

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

Another direction for further research concerns classical circuit partitioning. If the effect on the clock period of the circuit 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 cutsize.

References

[1] N. Sherwani, *Algorithms for VLSI Physical Design Automation*. Kluwer Academic Publishers, 1995.

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

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

test circuit			M=5			M=10			M=15			average
name	ϕ_{opt}	#g	ϕ_{opt}	#g	#c	ϕ_{opt}	#g	#c	ϕ_{opt}	#g	#c	time m's"
s208.1	10	104	13	107	22	11	109	12	11	107	8	00'01"
s349	14	161	20	161	33	18	162	17	16	163	12	00'18"
s444	7	181	10	182	39	9	181	19	8	190	14	00'06"
s635	66	286	92	289	59	79	286	29	74	286	20	00'20"
s713	74	393	104	407	83	90	396	41	84	393	27	01'34"
s820	10	289	15	289	59	13	289	30	13	289	20	00'31"
s938	16	446	22	452	93	18	459	47	18	454	31	00'21"
s1196	24	529	34	536	109	29	547	56	28	531	37	00'17"
s1238	22	508	30	523	107	27	545	55	25	530	36	00'15"
s1488	16	653	22	677	138	19	692	72	18	728	51	01'17"
s1494	16	647	22	682	142	19	670	70	18	705	50	01'25"
s1512	23	780	32	798	161	28	810	82	26	891	64	09'14"
s3330	14	1789	20	1805	363	17	2050	211	16	2040	141	04'25"
s5378	21	2779	31	2838	573	27	3130	319	25	2932	200	31'31"

Table 1: Experimental results.

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

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

[6] L.-T. Liu, M.-T. Kuo, C.-K. Cheng, and T. Hu, "A replication cut for two-way partitioning," *IEEE Trans. on Computer-Aided Design*, vol. 14, pp. 623–630, 1995.

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

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

[9] L. J. Hwang and A. E. Gamal, "Min-cut replication in partitioned networks," *IEEE Trans. on Computer-Aided Design*, vol. 14, pp. 96–106, 1995.

[10] C. Kring and A. Newton, "A cell-replicating approach to min-cut-based circuit partitioning," in *Intl. Conf. on Computer-Aided Design (ICCAD)*, pp. 2–5, 1991.

[11] K. Roy-Neogi and C. Sechen, "Partitioning with performance optimization," in *International Symposium on Field-Programmable Gate Arrays*, pp. 146–152, 1995.

[12] M. Shih and E. S. Kuh, "Circuit partitioning under capacity and I/O constraints," in *Custom Integrated Circuits Conference*, pp. 659–662, 1994.

[13] H. Yang and D. Wong, "New algorithms for min-cut replication in partitioned circuits," in *Intl. Conf. on Computer-Aided Design (ICCAD)*, pp. 216–223, 1995.

[14] R. Murgai, R. Brayton, and A. Sangiovanni-Vincentelli, "On clustering for minimum delay/area," in *Intl. Conf. on Computer-Aided Design (ICCAD)*, pp. 6–9, 1991.

[15] E. Lawler, K. Levitt, and J. Turner, "Module clustering to minimize delay in digital networks," *IEEE Trans. on Computers*, vol. 18, pp. 47–57, 1969.

[16] R. Rajaraman and D. Wong, "Optimal clustering for delay minimization," in *ACM/IEEE Design Automation Conf. (DAC)*, pp. 309–314, 1993.

[17] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, pp. 5–35, 1991.

[18] L.-T. Liu, M. Shih, N. Chou, C.-K. Cheng, and W. Ku, "Performance-driven partitioning using retiming and replication," in *Intl. Conf. on Computer-Aided Design (ICCAD)*, pp. 296–299, 1993.

[19] 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)*, pp. 206–210, 1995.

[20] 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.

[21] P. Pan, "Continuous retiming: algorithms and applications," in *Intl. Conf. on Computer Design (ICCD)*, 1997. (to appear).

[22] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. New York: McGraw-Hill Book Company, 1990.