An Efficient Algorithm for Performance-Optimal FPGA Technology Mapping with Retiming

Jason Cong and Chang Wu

Abstract— **It is known that most field programmable gate array (FPGA) mapping algorithms consider only combinational circuits. Pan and Liu [22] recently proposed a novel algorithm, named SeqMapII, of technology mapping with retiming for clock period minimization. Their algorithm, however, requires $O(K^3 n^5 \log(Kn^2) \log n)$ run time and $O(K^2 n^2)$ space for sequential circuits with n gates. In practice, these requirements are too high for targeting K-lookup-table-based FPGA's implementing medium or large designs. In this paper, we present three strategies to improve the performance of the SeqMapII algorithm significantly. Our algorithm works in $O(K^2 n_l n \mid P_v \mid \log n)$ run time and $O(K \mid P_v \mid)$ space, where n_l is the number of labeling iterations and $\mid P_v \mid$ is the size of the partial flow network. In practice, both n_l and $\mid P_v \mid$ are less than n. Area minimization is also considered in our algorithm based on efficient low-cost K-cut computation.**

Index Terms—**Expanded circuit, field programmable gate array (FPGA), lookup table, retiming, technology mapping.**

I. Introduction

The technology mapping and synthesis problem for field programmable gate array (FPGA's) is to produce an equivalent circuit for a given circuit using only specific programmable logic blocks (PLB's). More specifically, without synthesis, the PLB's in a mapping solution form a cover of gates in the original circuit possibly with overlap. There are a variety of different PLB architectures. In this paper, we consider a generic type of PLB: the K-input lookup table (K-LUT), which has been widely used in current FPGA technology [1], [18], [30]. Most of the previous LUT mapping algorithms optimize either area (e.g., [13], [14], and [20]) or delay (e.g., [5], [15], and [21]). The algorithms in [4] and [7] consider both delay and area. The algorithms in [25] and [27] consider the routability. A comprehensive survey of FPGA mapping algorithms is given in [6]; however, most of these approaches apply only to combinational circuits. For sequential circuits, these approaches assume that the positions of flip-flops (FF's) are fixed so that the entire circuit can be partitioned into combinational subcircuits, each of which is mapped separately. A major limitation of these approaches is that they do not consider mapping and retiming simultaneously. In fact, the optimal mapping solutions for all combinational subcircuits may not lead to an optimal mapping solution for the entire sequential circuit due to the effect of *retiming*. Retiming is a technique of moving FF's within the circuit without changing the circuit behavior. For single-phase clock and edge-triggered FF's, Leiserson and Saxe [16], [17] solved the retiming problem of minimizing the clock period or the number of FF's.

Several FPGA synthesis and mapping algorithms have been proposed specifically for sequential circuits. The approach in [19] does not consider retiming, but rather, its objective is to consider proper packing of LUT's with FF's to minimize the number of configurable logic blocks for Xilinx FPGA's [30]. The methods in [23] and [29] are heuristics that consider loopless sequential circuits. Touati *et al.* [28] proposed an approach of retiming specifically for Xilinx FPGA's after mapping, placement, and routing. A significant advancement was made recently by Pan and Liu [22]. They proposed a novel algorithm, named SeqMapII, to find a mapping solution with the minimum clock period under retiming. Similar to the FlowMap algorithm [5], their algorithm works in two phases: the labeling phase and the mapping generation phase. They introduced the idea of expanded circuits to represent all possible K-LUT's under retiming and node-replication. An iterative method is used to compute labels for all nodes. The time and space complexities for SeqMapII are $O(K^3 n^5 \log(Kn^2) \log n)$ and $O(K^2 n^2)$, respectively, for a circuit with n gates [22].[1] Although the SeqMapII algorithm runs in polynomial time, it has two shortcomings: 1) too many candidate values ($O(Kn^2)$) need to be considered for each label update and 2) the expanded circuits are too large ($O(Kn^2)$ nodes) for computing the optimal solutions. Experimental results show that the run time of SeqMapII for computing the optimal solutions is too long in practice (e.g., more than 12 h of CPU time for a design of 134 gates on a SPARC5 workstation).

In this paper, we present three strategies to improve the performance of the label computation significantly, which is the most time-consuming step in SeqMapII [22]. First, we prove that the monotone property of labels holds for sequential circuits, then develop an efficient label update to speed up the algorithm by a factor of $\log(Kn^2)$. Second, we propose a new approach of K-cut computation on partial flow networks, which are much smaller than the expanded circuits used in SeqMapII, while guaranteeing the optimality of the results.

Manuscript received March 11, 1997. This work was supported in part by the National Science Foundation under Young Investigator Award MIP9357582 and by grants from Xilinx and Lucent Technologies under the California MICRO program. This paper was recommended by Associate Editor A. Saldanha.

The authors are with the Computer Science Department, University of California, Los Angeles, CA 90095 USA.

Publisher Item Identifier S 0278-0070(98)06759-1.

[1]The authors of [22] later reduced the time complexity of SeqMapII to $O(K^3 n^5 \log n)$ [24] using the monotone property to be presented in Section IV of this paper (first presented in [10]).

Our experimental results show that the average numbers of nodes in the partial flow networks are far less than n, which is a big improvement over the $O(Kn^2)$ number of nodes in the expanded circuits used in SeqMapII [22]. Last, strongly connected-component (SCC) partitioning and heuristic label ordering are used to eliminate much redundant label computation to further speed up the algorithm. In practice, our algorithm works in $O(K^2n^3 \log n)$ time and $O(Kn)$ space according to our experimental results. The area reduction is also considered in our algorithm by choosing a low-cost K-cut for every node. As a result, our algorithm is 2.8 $\times$ 10^3 times faster than SeqMapII-opt for computing optimal solutions, and even eight times faster than SeqMapII-heu, which uses very small expanded circuits as a heuristic. Furthermore, our algorithm reduces LUT count by 28%, and FF count by 27% and achieves clock periods 23% shorter as compared with SeqMapII-heu [22].

The remainder of this paper is organized as the following. Section II presents the problem formulation and definitions. Section III gives a review of the approach by Pan and Liu [22]. Our improved algorithm is presented in Section IV. The experimental results are presented in Section V, followed by conclusions and future work in Section VI.[2]

II. Problem Formulation and Definitions

Given a *sequential* circuit, the technology mapping problem for K-LUT-based FPGA's is to construct an equivalent circuit consisting of K-LUT's and FF's. For performance optimization, we study the following problem.

Problem 1: For a sequential circuit, find an equivalent LUT circuit with the minimum clock period under retiming.

As in [5] and [22], the *unit delay model* is used in this paper, which assumes that the delay of each LUT is one and the delay of each net is zero or a constant.

A mapping solution in which the output signals of all LUT's are from the original circuit is called a *simple mapping solution* [22]. As shown in Fig. 1(b), the outputs of LUT$_u$ and LUT$_w$ are the outputs of u and w in the original circuit shown in Fig. 1(a). But the output of LUT$_v$ in Fig. 1(c) is one clock cycle ahead of the output of v in the original circuit. Pan and Liu [22] showed that there exists a simple mapping solution whose clock period under retiming is equal to the minimum clock period among all mapping with retiming solutions. This means that there is no need to move FF's before mapping in order to get an optimal solution to Problem 1. Furthermore, they proposed to solve the decision version of the problem.

Problem 2: Given a target clock period ϕ, determine the existence of a simple mapping solution whose clock period under retiming is no more than ϕ.

As in [5] and [22], the results in this paper apply to only K-bounded networks, i.e., each gate in the network has at most K fan-ins.[3] In the remainder of this paper, all circuits are assumed to be K-bounded.

[2] An extended abstract of this paper appears in [10].

[3] When a circuit is not K-bounded, we can use gate decomposition algorithms in [2], [3], and [8] to decompose gates with more than K fan-ins.

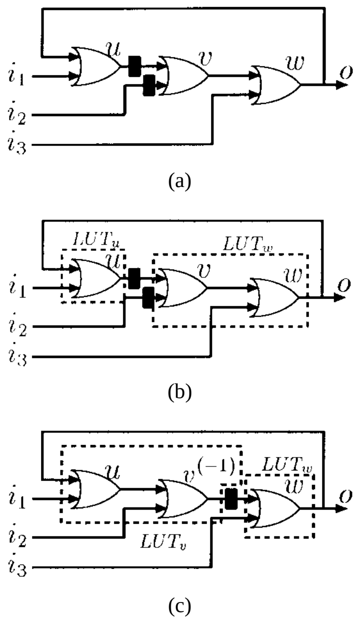

Fig. 1. Simple and nonsimple mapping solutions.

We use $G(V, E, W)$ or G to denote the *retiming graph* [17] of a sequential circuit, where V is the set of nodes representing the gates in the circuit, E is the set of edges representing the connection between the gates, and W is the set of edge weights. Edge $e(u, v)$ denotes the connection from gate u to gate v and $w(e)$ denotes the number of FF's on the connection. The *path weight,* denoted $w(p)$ of a path p, is the sum of weights of all edges on the path. G_v is a subgraph of G consisting of node v and all nodes that have paths to v.

For a simple mapping solution M and a given clock period ϕ, the *edge length,* denoted $length(e)$ of an edge e, is defined to be $-\phi \cdot w(e) + 1$. The *path length,* denoted $length(p)$ of a path p, is $\sum_{e \in p} length(e)$. The l-value $l_M(v)$ of a node v in a mapping solution M is the maximum length of all paths from primary inputs (PI's) to v in M. We define $l_M(v) = +\infty$ if there is a path from one of the PI's to v going through a (feedback) loop of positive length. It was shown that for a mapping solution M and a given ϕ, the retimed clock period is no more than ϕ if and only if $l_M(v) \leq \phi + 1$ *for every PO* v [22]. The *label* of node v, denoted $l^{opt}(v)$, is defined to be the *minimum* l-values of the K-LUT's rooted at v among *all* mapping solutions. Clearly, there is a mapping solution with retimed clock period of no more than ϕ if and only if $l^{opt}(v) \leq \phi + 1$ for every primary output (PO) v [22].

Now let us introduce the definition of K-cuts. In a directed graph with one sink and one source, a *cut* $(X, \overline{X})$ is a partition of the graph such that the sink is in $\overline{X}$ and the source is in X. The *node cut-set* $V(X, \overline{X})$ is the set of nodes in X that are connected directly to nodes in $\overline{X}$. If $\mid V(X, \overline{X}) \mid \leq K$, a cut $(X, \overline{X})$ is called a *K-feasible cut,* or *K-cut* in short. A cut is a *min-cut* if $\mid V(X, \overline{X}) \mid$ is minimum. To determine the existence of a K-cut, one can compute the max-flow from the source to the sink and decide whether it is larger than K (see [5] for details). This process is called *K-cut computation* in this paper.

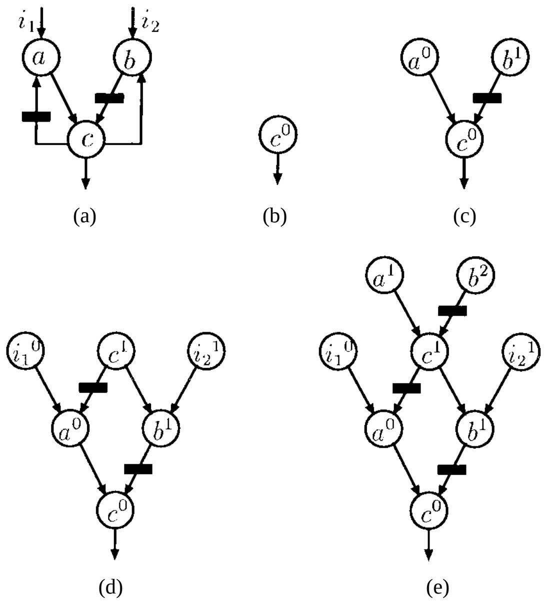

Fig. 2. Expanded circuits.

III. Review of the SeqMapII Algorithm

The SeqMapII algorithm works in two phases: label computation and mapping generation. The label computation starts with a lower bound on the value of each node label and repeatedly improves the lower bounds until they all converge to the node labels. The initial lower bounds are zero for the PI's and $-\infty$ for the other nodes. Based on the current lower bound $l(v)$ of each node v, Pan and Liu [22] presented a procedure to determine a new lower bound $l_{new}(v)$. They introduced the concept of expanded circuits for each node v to represent all possible K-LUT's rooted at v with consideration of retiming and node replication. An expanded circuit $\mathcal{E}_v^i$ of node v with *control number* i is a directed acyclic graph (DAG) rooted at v formed by replication of nodes in G_v, such that $\mathcal{E}_v^i$ has the property that all paths from a node to the root have the same number of FF's. If a replication of node u passes w FF's before reaching the root v in $\mathcal{E}_v^i$, we denote it u^w and call its *weight* w. The *control number* i of $\mathcal{E}_v^i$ is the shortest distance (in terms of the number of edges) between the root and each leaf in $\mathcal{E}_v^i$ that is not a replication of a PI in G_v. For example, Fig. 2(b)–(e) is four expanded circuits $\mathcal{E}_c^i$ of node c shown in Fig. 2(a) with control number i from zero to three, respectively.

Pan and Liu [22] showed that to examine all K-LUT's for a node v, it sufficed to examine all the K-LUT's that can be derived from the K-cuts in $\mathcal{E}_v^{Kn}$. With the assumption that the weight of each edge is at most one, it was shown that the numbers of nodes and edges in $\mathcal{E}_v^{Kn}$ are bounded by $O(Kn^2)$ and $O(K^2n^2)$, respectively, where n is the number of gates in the original circuit [22]. In an expanded circuit of node v, the *height* $h(X,\overline{X})$ of a K-cut $(X,\overline{X})$ is defined as

$$h(X,\overline{X}) = \max_{\forall u^w \in V(X,\overline{X})} \{l(u) - \phi \cdot w + 1\}$$

based on the current lower bounds $l(u)$ of node labels. The new lower bound is computed as

$$l_{new}(v) = \min_{\forall K\text{-cut}(X,\overline{X}) \in \mathcal{E}_v^{Kn}} h(X,\overline{X}).$$

This value is determined by binary search among $O(Kn^2)$ candidates in the set of $\{l(u) - \phi \cdot w + 1 \mid u^w \in \mathcal{E}_v^{Kn}\}$ and performing a K-cut computation for each candidate value. The computation time for every $l_{new}(v)$ is $O(K^3n^2\log(Kn^2))$ based on network flow computation. The labels of all nodes can be determined in $O(K^3n^5\log(Kn^2))$ time because there is a total of n^3 label updates [22].[4]

SeqMapII [22] is the first polynomial algorithm to find a mapping solution with the optimal clock period under retiming. However, two major shortcomings make this approach inefficient in practice. First, the expanded circuit $\mathcal{E}_v^{Kn}$ is too large [$O(Kn^2)$ nodes and $O(K^2n^2)$ edges], which requires prohibitively large memory and run time for circuits with more than 1000 gates. Second, too many values ($O(Kn^2)$) have to be considered when computing the new lower bound of each node label.

IV. TurboMap Algorithm

In this section, we present three strategies to improve the label computation of the SeqMapII algorithm, which is the most time-consuming step. First, we prove the monotone property of the node labels and develop a new procedure for computing a *tighter* lower bound with a *single* K-cut computation. Second, we propose a new approach to compute K-cuts on much smaller partial flow networks, which are built *incrementally* during the K-cut computation. Third, SCC partitioning and depth-first-search (DFS) ordering are used to eliminate much redundant label computation and reduce the number of labeling iteration to further speed up the algorithm.

A. Label Update with Single K-Cut Computation

In SeqMapII [22], to compute $l_{new}(v)$ for a node, it is necessary to perform binary search among all $O(Kn^2)$ possible values in $\{l(u) - \phi \cdot w + 1 \mid u^w \in \mathcal{E}_v^{Kn}\}$, which requires $O(\log(Kn^2))$ K-cut computations. In our approach, we compute a *tighter* lower bound $l'_{new}(v)$ with *single* K-cut computation to speed up the algorithm by a factor of $O(\log(Kn^2))$. Let

$$\mathcal{L}(v) = \max_{\forall e(u,v) \in G_v} \{l(u) - \phi \cdot w(e)\}.$$

We update the lower bound on the value of the label of v as follows:

$$l'_{new}(v) = \begin{cases} \mathcal{L}(v), & \text{if } \exists K\text{−cut with } h(X,\overline{X}) \le \mathcal{L}(v) \\ \mathcal{L}(v) + 1, & \text{otherwise.} \end{cases}$$

Obviously, $l'_{new}(v)$ can be computed with single K-cut computation. Recall that this result is similar to Lemma 2 in [5], which applies to combinational circuits only.

[4]The total number of label updates was shown to be $O(n^2)$ in [22]. In [24], however, it changed to $O(n^3)$ due to the difficulty of proving that $O(n^2)$ guarantees finding an optimal solution.

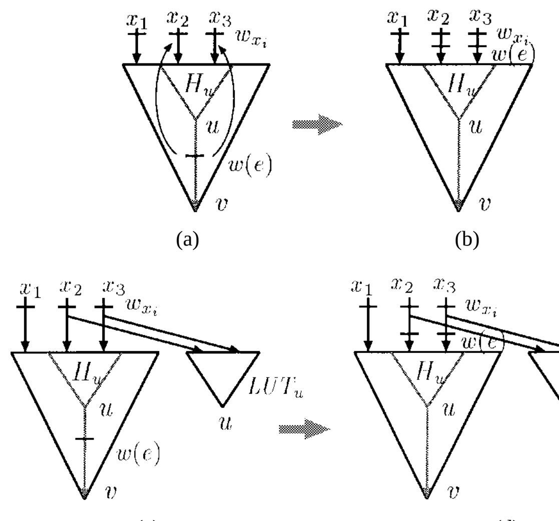

Fig. 3. Proof of monotone property of node labels.

The correctness of our approach is based on the fact that $l_{new}(v) \leq l'_{new}(v) \leq l^{opt}(v)$, where $l_{new}(v)$ is the lower bound computed in SeqMapII [22]. This can be proved based on the *monotone property* of node labels. In a sequential circuit, which has a mapping solution with a clock period of no more than a given ϕ, we say that the set of its node labels $l^{opt}(v)$ is *monotone* if for any edge $e(u,v)$, $l^{opt}(u) - \phi \cdot w(e) \leq l^{opt}(v)$.

Theorem 1 (Monotone Property): In a sequential circuit that has a mapping solution with the clock period of no more than a given ϕ under retiming, the node labels are monotone. That is, $l^{opt}(u) - \phi \cdot w(e) \leq l^{opt}(v)$ for every edge (u,v) in the retiming graph of the circuit.

Proof: For each edge $e(u,v)$ in the original circuit, there exists a simple mapping solution M such that $l_M(v) = l^{opt}(v)$. Let LUT$_v$ denote the K-LUT rooted at v in M. We consider the following two cases.

Case 1: u is a fan-in to LUT$_v$. According to the definition of l-values, the l-values of u and v in M satisfy $l_M(u) - \phi \cdot w(e) + 1 \leq l_M(v) = l^{opt}(v)$. Since $l^{opt}(u) \leq l_M(u)$ by definition, we have $(l^{opt}(u) - \phi \cdot w(e) < l_M(u) - \phi \cdot w(e) + 1 \leq l^{opt}(v))$.

Case 2: u is covered inside LUT$_v$, as shown in Fig. 3(a). Let $w(e)$ be the number of FF's on edge $e(u,v)$ and w_{x_i} be the number of FF's on edge $e(x_i, \mathrm{LUT}_v)$ for each fan-in x_i of LUT$_v$. In formation of LUT$_v$, we have to push those $w(e)$ FF's on $e(u,v)$ back to LUT$_v$'s fan-in edges as shown in Fig. 3(b).[5] Let H_u be the sub-DAG rooted at u inside LUT$_v$.[6] The edge weight after retiming of each fan-in edge $e(x_i, H_u)$ of H_u is $w^r_{x_i} = w_{x_i} + w(e)$. The edge weight for the rest of the fan-ins of LUT$_v$ remains unchanged, i.e., $w^r_{x_i} = w_{x_i}$. Since H_u can be covered by a K-LUT by replicating H_u explicitly outside LUT$_v$ to form LUT$_u$, we get another simple mapping solution M' as shown in Fig. 3(d). Note that the weight of edge $e(x_i, \mathrm{LUT}_u)$ is w_{x_i} because those $w(e)$ FF's were pushed back only on edges $e(x_i, \mathrm{LUT}_v)$. The l-value $l_{M'}(u)$ in M' is

$$\begin{aligned} l_{M'}(u) &= \max_{\forall\,\text{fanin}\, x_i \,\text{to}\, \mathrm{LUT}_u} \{l_{M'}(x_i) - \phi \cdot w_{x_i} + 1\} \\ &= \max_{\forall\,\text{fanin}\, x_i \,\text{to}\, \mathrm{LUT}_u} \{l_M(x_i) - \phi \cdot w_{x_i} + 1\}. \end{aligned}$$

Since every input of LUT$_u$ is also an input of LUT$_v$

$$\begin{aligned} l^{opt}(v) &= l_M(v) = \max_{\forall\,\text{fanin}\, x_i \,\text{to}\, \mathrm{LUT}_v} \{l_M(x_i) - \phi \cdot w^r_{x_i} + 1\} \\ &\geq \max_{\forall\,\text{fanin}\, x_i \,\text{to}\, H_u} \{l_M(x_i) - \phi \cdot (w_{x_i} + w(e)) + 1\} \\ &= \max_{\forall\,\text{fanin}\, x_i \,\text{to}\, H_u} \{l_M(x_i) - \phi \cdot w_{x_i} + 1\} - \phi \cdot w(e) \\ &= l_{M'}(u) - \phi \cdot w(e) \\ &\geq l^{opt}(u) - \phi \cdot w(e). \end{aligned}$$

This concludes that $l^{opt}(u) - \phi \cdot w(e) \leq l^{opt}(v)$ for any edge $e(u,v)$ in the original circuit. □

Let *one iteration* denote the computation process where $l(v)$ is updated once for every node v (in an arbitrary order). We prove the following.

Theorem 2: For a sequential circuit G that has a mapping solution with a clock period under retiming of no more than a given ϕ, the inequalities $l_{new}(v) \leq l'_{new}(v) \leq l^{opt}(v)$ hold during every iteration.

Proof: It is clear that $l_{new}(v) \leq l'_{new}(v)$ based on the definitions of $l_{new}(v)$ and $l'_{new}(v)$. We prove $l'_{new}(v) \leq l^{opt}(v)$ by mathematical induction.

Initially, $l(v) = -\infty < 1^{opt}(v)$. Now suppose that $l(v) \leq l^{opt}(v)$ holds for every node v at the current iteration. We prove that the newly updated lower bound $l'_{new}(v) \leq l^{opt}(v)$ holds for every node v. Since $l(v) \leq l^{opt}(v)$, we have

$$\begin{aligned} \mathcal{L}(v) &= \max_{\forall\, e(u,v) \in G} \{l(u) - \phi \cdot w(e)\} \\ &\leq \max_{\forall\, e(u,v) \in G} \{l^{opt}(u) - \phi \cdot w(e)\} \\ &\leq l^{opt}(v). \end{aligned}$$

Case 1: If there is a K-cut in $\mathcal{E}_v^{Kn}$ with $h(X, \overline{X}) \leq \mathcal{L}(v)$, then $l'_{new}(v) = \mathcal{L}(v) \leq l^{opt}(v)$.

Case 2: If there is no K-cut in $\mathcal{E}_v^{Kn}$ with $h(X, \overline{X}) \leq \mathcal{L}(v)$, then $l'_{new}(v) = \mathcal{L}(v) + 1$. To prove $l'_{new}(v) \leq l^{opt}(v)$, let us prove by contradiction. Suppose $l^{opt}(v) < l'_{new}(v) = \mathcal{L}(v) + 1$. Since $\mathcal{L}(v) \leq l^{opt}(v)$, it must be that $l^{opt}(v) = \mathcal{L}(v)$.

Suppose M is a simple mapping solution such that $l_M(v) = l^{opt}(v)$ and LUT$_v$ is the LUT rooted at v. Since any LUT rooted at v corresponds to a K-cut in $\mathcal{E}_v^{Kn}$, let $(X, \overline{X})$ be the cut with $\overline{X} = \mathrm{LUT}_v$. So $(l^{opt}(v) = l_M(v) = \max_{\forall u^w \in V(X,\overline{X})} \{l_M(u) - \phi \cdot w + 1\})$. Based on the definition of $l^{opt}(u)$ and the assumption that $l(u) \leq l^{opt}(u)$, it must be the case that

$$\begin{aligned} \mathcal{L}(v) = l^{opt}(v) = l_M(v) &= \max_{\forall u^w \in V(X,\overline{X})} \{l_M(u) - \phi \cdot w + 1\} \\ &\geq \max\{l^{opt}(u) - \phi \cdot w + 1\} \quad \forall u^w \in V(X, \overline{X}) \\ &\geq \max\{l(u) - \phi \cdot w + 1\} \quad \forall u^w \in V(X, \overline{X}). \end{aligned}$$

It means that $h(X, \overline{X}) = \max_{\forall u^w \in V(X,\overline{X})} \{l(u) - \phi \cdot w + 1\} \leq \mathcal{L}(v)$. The result is contradictory to the assumption that there is no K-cut in $\mathcal{E}_v^{Kn}$ with height of no more than $\mathcal{L}(v)$. ■

[5] Note that the FF's cannot be pushed down to the output of LUT$_v$, as M is a simple mapping solution.

[6] Note that the FF's inside H_u need to be pushed back on edge $e(x_i, H_u)$ as well. For ease of presentation, we assume that w_{x_i} includes both the numbers of FF's originally on edge $e(x_i, H_u)$ and from inside H_u.

Note that the above results depend on neither the order of label update nor the number of iterations performed. Therefore, we can conclude that the inequalities $l_{new}(v) \leq l'_{new}(v) \leq l^{opt}(v)$ hold for every node v during every iteration.

B. K-Cut Computation on Partial Flow Networks

In this subsection, we present a new approach to determine if $l'_{new}(v) = \mathcal{L}(v)$ by max-flow computation on much smaller partial flow networks than that of the expanded circuit $\mathcal{E}_v^{Kn}$ used in [22]. To check whether $l'_{new}(v) \leq \mathcal{L}(v)$ with the approach in SeqMapII [22], one needs to build the expanded circuit $\mathcal{E}_v^{Kn}$ and construct the corresponding flow network, and then decide the existence of a K-cut by max-flow computation. In TurboMap, however, we construct the flow network incrementally without constructing the entire $\mathcal{E}_v^{Kn}$. More important, we construct the flow network *just large enough* to determine whether a K-cut exists. Recall that all the previous flow-computation-based FPGA mapping algorithms [5], [7], and [23] build the entire flow network before the max-flow computation. Thus, they are less efficient than our approach, and can be improved in a similar way.

The basic idea of our algorithm is that, although the flow network for $\mathcal{E}_v^{Kn}$ is very large, the union of the $K+1$ *shortest* augmenting paths (in terms of the number of edges) is usually much smaller. (Note that we only need to determine whether or not the value of a max-flow is no more than K. Searching for $K+1$ shortest augmenting paths is enough.) So if we start from v^0 and grow the flow network during the K-cut computation *incrementally,* the flow network constructed would be much smaller than $\mathcal{E}_v^{Kn}$. We call the flow networks constructed by our approach the *partial flow networks.*

When updating the label lower bound for node v, we construct the partial flow network, denoted $\mathcal{P}_v$, directly from G_v. As shown in Fig. 4(c), the edge direction in $\mathcal{P}_v$ is *reversed* from that in G_v [shown in Fig. 4(a)]. The v^0 is the source of $\mathcal{P}_v$, and all the u^w for PI u will be connected to the sink t of $\mathcal{P}_v$.

A node u^w in an expanded circuit of v is *critical* if $l(u) - \phi \cdot w \geq \mathcal{L}(v)$. The basic idea of partial flow network construction is to perform breadth-first search on $\mathcal{P}_v$ during the construction of $\mathcal{P}_v$. We maintain a first-in, first-out (FIFO) queue Q, which initially includes only node v^0, the source of $\mathcal{P}_v$. Each time, we fetch a node u^w from Q to process and add new nodes to the end of Q until Q is empty or u^w has an edge to the sink t. Suppose u^w is the current node fetched from the queue. If u^w has fan-ins in the partially constructed $\mathcal{P}_v$, we put the fan-ins to the end of Q. If, however, u^w does not have fan-ins and u is not a PI in G_v, we create the fan-in edges for u^w and add new nodes to the flow network as follows. For each fan-in edge $e(a, u)$ of u in G_v, we create two nodes $a^{w+w(e)}$ and $a_1^{w+w(e)}$ if they have not been created and put $a^{w+w(e)}$ first and then $a_1^{w+w(e)}$ to the end of Q. We add a new edge $e(a^{w+w(e)}, a_1^{w+w(e)})$ and assign the flow capacity to be ∞ if $a^{w+w(e)}$ is a critical node, or "1" otherwise. Then we add edge $e(u^w, a^{w+w(e)})$ with flow capacity of ∞. If u is a PI in G_v, we connect u^w to the sink t with flow capacity of ∞ and find one augmenting path. Whenever an augmenting path is found, we augment the path, clear Q, and start from v^0 again to search for another shortest augmenting path until no more augmenting paths exist [in this case, we assign $l'_{new}(v) = \mathcal{L}(v)$], or we find the $(K+1)$th augmenting path [here, a K-cut does not exist and we assign $l'_{new}(v) = \mathcal{L}(v) + 1$].

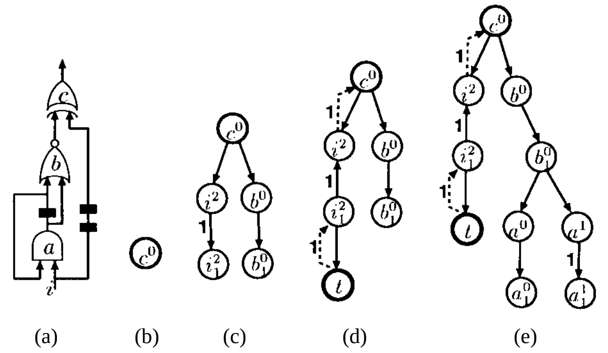
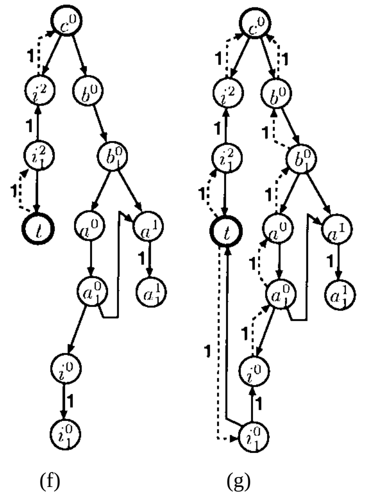

Fig. 4. Incremental construction of the partial flow network with K-cut computation. (a) G_c and (b)–(g) are the flow networks we constructed step by step. The sources of the flow networks are c^0; the sinks are t. The number beside each edge is the flow capacity, which, in default, is ∞.

Let us look at an example of constructing the partial flow network for node c in G_c shown in Fig. 4(a). Node i is a primary input with $l(i) = 0$. Each black bar represents an FF. For $K = 3$ and $\phi = 1$, suppose $l(a) = l(b) = 1$. We now compute $l'_{new}(c)$. Since $\mathcal{L}(c) = 1$, we only need to decide whether $l'_{new}(c) = \mathcal{L}(c)$ or $\mathcal{L}(c) + 1$. The construction of the partial flow network is shown step by step in Fig. 4(b)–(g). At first, we create c^0 as the source of the partial flow network and put it in an FIFO queue Q. Then, for c^0 getting from Q, nodes b^0, i^2 and edges $e(c^0, b^0), e(c^0, i^2)$ with flow capacity of ∞ will be created based on edges $e(b, c), e(i, c)$ in G_c. Nodes b^0, i^2 will be put to the end of Q. In the following, b^0, i^2 will be fetched successively from Q, and nodes b_1^0, i_1^2 and edges $e(b^0, b_1^0), e(i^2, i_1^2)$ will be created. Since b^0 is critical [because $l(b) - \phi \cdot 0 = 1 = \mathcal{L}(c)$], the flow capacity of edge $e(b^0, b_1^0)$ is ∞. On the other hand, i^2 is not critical [because $l(i) - \phi \cdot 2 = -2 < 1 = \mathcal{L}(c)$], so the flow capacity of edge $e(i^2, i_1^2)$ is one. The current flow network is shown in Fig. 4(c). Since i is a PI, a new edge $e(i_1^2, t)$ with flow capacity of ∞ will be created, and one shortest augmenting path is found. After augmenting this path, we clear Q and start from c^0 again

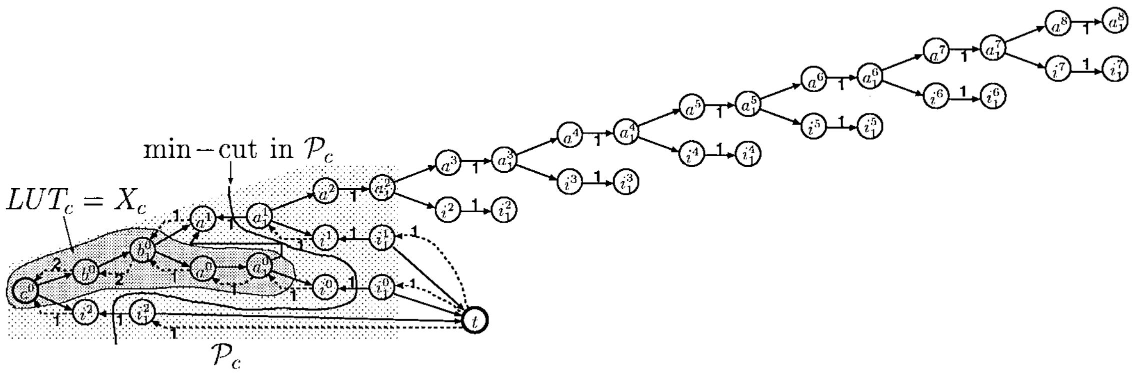

Fig. 5. K-cut computation on partial flow networks. The heavy shaded area is LUT$_c$. The light shaded area is the partial flow network $\mathcal{P}_c$. The entire network corresponds to $\mathcal{E}_c^{Kn=9}$.

to search for another augmenting path. The flow network is shown in Fig. 4(d). When reaching b_1^0, which was created in previous steps, we need to create the fan-out edges of b_1^0 and add two pairs of nodes $\{a^0, a_1^0\}$ and $\{a^1, a_1^1\}$. The new flow network is shown in Fig. 4(e). We then create the fan-out edges of a_1^0 and add two nodes $\{i^0, i_1^0\}$, as shown in Fig. 4(f). Now we find and augment another augmenting path. The new flow network is shown in Fig. 4(g).

As shown in Fig. 5, after finding three augmenting paths, there exist no more. The value of max-flow is $3(= K)$ and therefore $l'_{new}(c) = \mathcal{L}(c) = 1$. The light shaded area is the partial flow network $\mathcal{P}_c$, which corresponds to the expanded circuit $\mathcal{E}_c^3$ in this example. As shown in Fig. 5, $\mathcal{P}_c$ is much smaller than the entire flow network corresponding to $\mathcal{E}_c^{Kn=9}$. The heavy shaded area X_c shown in Fig. 5, of the min-cut $(X_c, \overline{X_c})$ in $\mathcal{P}_c$, forms a K-LUT rooted at c.

Since only the first $(K+1)$ shortest augmenting paths are searched, the incrementally constructed flow networks are much smaller than the flow networks corresponding to the large $\mathcal{E}_v^{Kn}$. Our experimental results on MCNC and ISCAS benchmarks show that the average numbers of nodes in the partial flow networks are always far less than n. As a result, each label update takes only $O(K^2n)$ time and $O(Kn)$ space in practice.

To bound the partial flow networks to be no larger than the flow network of $\mathcal{E}_v^{Kn}$, we add an additional criterion to the partial flow network construction. Since the shortest path from a leaf to the root in $\mathcal{E}_v^{Kn}$ is bounded by Kn and each node needs to be split into two nodes to construct the corresponding flow network, the shortest path from the root to a leaf in the flow network of $\mathcal{E}_v^{Kn}$ is bounded by $2Kn$. Therefore, we limit the augmenting path length to be no more than $2Kn$ in $\mathcal{P}_v$. In other words, if the shortest augmenting path length from the source v^0 to u^w during the construction of $\mathcal{P}_v$ is $2Kn$, we mark u^w as a leaf and connect it directly to the sink without growing further to u's fan-ins. Let $\mathcal{F}_v^{2Kn}$ be the corresponding flow network of $\mathcal{E}_v^{Kn}$. We prove that $\mathcal{P}_v \subseteq \mathcal{F}_v^{2Kn}$, and there is a K-cut in $\mathcal{F}_v^{2Kn}$ if and only if there is a K-cut in $\mathcal{P}_v$.

Theorem 3: $\mathcal{P}_v \subseteq \mathcal{F}_v^{2Kn}$.

Proof: Let $\delta_f(v^0, u^w)$ denote the shortest path length (in terms of the number of edges) from the root v^0 to u^w in the residue flow network after pushing f flows. Based on Lemma 27.8 in [11], the shortest path distance in the residual graph is monotonically increasing if we always augment the shortest augmenting paths. So $\delta_0(v^0, u^w) \le \delta_f(v^0, u^w) \le 2Kn$ for any leaf $u^w \in \mathcal{P}_v$. This implies that the shortest path distance from the root to every leaf in $\mathcal{P}_v$ is less than $2Kn$. So $\mathcal{P}_v \subseteq \mathcal{F}_v^{2Kn}$. □

Theorem 4: There exists a K-feasible cut in $\mathcal{F}_v^{2Kn}$ if and only if there exists a K-feasible cut in $\mathcal{P}_v$.

Proof: ($\Leftarrow$) It is obvious because $\mathcal{P}_v \subseteq \mathcal{F}_v^{2Kn}$.

($\Rightarrow$) We shall prove that for the min-volume min-cut $(X, \overline{X})$ in $\mathcal{F}_v^{2Kn}$, X and V$(X, \overline{X})$ are included in $\mathcal{P}_v$, where the min-volume min-cut is a min-cut with the minimum $|X|$. (Note that the root v^0 is the source of the flow network and belongs to X.)

One important property of the min-volume min-cut $(X, \overline{X})$ is that every node in X is reachable from the source in the residual graph of a maximum flow based on Lemma 6 in [5].

Since $(X, \overline{X})$ is K-feasible, X has at most $2Kn$ nodes. The reason is that if $|X| > 2Kn$, at least one node u in the original circuit has $2K+1$ duplications in X with $\lceil (2K+1)/2 \rceil = K+1$ different weights. (Note that each node in $\mathcal{E}_v^{Kn}$ becomes a node pair with the same weight in $\mathcal{F}_v^{2Kn}$.) Since every node is reachable from the PI's (nodes isolated from the PI's can be deleted easily with a preprocessing), there must be a path p of $i \rightsquigarrow u$ in the original circuit from a PI i to u. Thus, for each duplication u^w, there is a unique duplication $u^w \rightsquigarrow i^{w+w(p)}$ of p in $\mathcal{F}_v^{2Kn}$. It means that the max-flow in $\mathcal{F}_v^{2Kn}$ is at least $K+1$, and the cut is not K-feasible.

Let F be a maximum flow with $value(F) \le K$. Since any $u^w \in X$ is reachable from the source v^0 and $|X| \le 2Kn$, the shortest path length in the residual graph $\delta_F(v^0, u^w) \le 2Kn - 1$. Therefore, for any flow f with $value(f) \le value(F)$, $\delta_f(v^0, u^w) \le \delta_F(v^0, u^w) \le 2Kn - 1$. Note that $u^w \notin \mathcal{P}_v$ if and only if $\delta_f(v^0, u^w) > 2Kn$. Thus, any u^w in X is also in $\mathcal{P}_v$. Since any $u_1^w \in V(X, \overline{X})$ is connected directly to a node $u^w \in X$, where $\{u^w, u_1^w\}$ is a node pair and will be created together in $\mathcal{P}_v$, thus $u_1^w \in \mathcal{P}_v$. □

C. SCC Partitioning and DFS Ordering

An SCC of a retiming graph $G(V, E, W)$ is a maximal set of vertices $U \subseteq V$ such that for every pair of nodes u and v in U, there are both paths $u \rightsquigarrow v$ and $v \rightsquigarrow u$. Clearly, the labels of two nodes u and v are mutually dependent only if they are in one SCC. For u and v in different SCC's, e.g., $u \rightsquigarrow v$ but $v \not\rightsquigarrow u$, the label of u can be computed before the label of v without iterations between u and v.

TurboMap first computes all the SCC's and sorts them in a topological order in linear time based on the algorithm in [12]. Then it labels the nodes in each SCC together. For different SCC's, the labels are computed separately in the topological order of SCC's from PI's to PO's. Our results show that with the SCC partitioning, the computation time of TurboMap can be reduced by 50% on average.

The computation order is also important to the rate of convergence of labeling computation. In [22], it is proved that the number of labeling iterations is bounded by $O(n^2)$, which is too large in practice. In TurboMap, we compute the node labels in the order based on DFS from the outputs to the inputs of each SCC. If node u is searched after node v, we update the label of u before v in every iteration. Our experimental results show that if there is a feasible solution for a target clock period, all the node labels can be computed in 5–20 iterations in almost all the cases, which is much less than the worst case bound of $O(n)$ under an arbitrary computation order.

D. Area Minimization

After obtaining the minimum clock period $\phi_{\min}$ and all the node labels, we can form a mapping with retiming solution by implementing each X as the LUT rooted at each node v for the min-cut $(X, \overline{X})$ of v computed during label computation. Although this solution has the minimum clock period after retiming, the area may not be good. In this subsection, we propose a heuristic to compute a low-cost K-cut for every node based on $\phi_{\min}$ to reduce the number of LUT's. Since the number of FF's is undetermined before retiming, we reduce only the number of LUT's during mapping, while minimizing the number of FF's during a postprocessing of retiming [17].

There are two approaches to reduce the numbers of LUT's in final mapping solutions. First is to enlarge the volume of each K-cut, as in [5]. Second is to maximize the sharing of fan-ins between LUT's as follows.

Similar to CutMap [7], we define the cost of a node to be zero if the node has already been marked as an LUT root, and one otherwise. Initially, all PI's, PO's, and nodes with large fan-out numbers are marked as LUT roots. Our approach begins with an FIFO queue Q containing all PO's. For every node v fetched from the queue, we compute a low-cost K-cut and put all nodes in the cut-set to the end of Q and mark them as LUT roots. If the cut cost of the cut of v computed during labeling is zero, the cut has the minimum cost. Otherwise, we try to compute another K-cut of v with smaller cost with two additional max-flow computations in a heuristic way.

First, we compute a zero-cost min-cut for node v by constructing a new flow network and computing the max-flow. Let the *cut capacity* of a node be the edge capacity of the edge connecting the node pair of the node in the new flow network.[7] We assign the cut capacity of u^w to be zero if u^w is noncritical, i.e., $l^{opt}(u) - \phi \cdot w + 1 \le l^{opt}(v)$, and u has already be marked

[7] Recall that each node in the expanded circuit corresponds to a pair of nodes in the flow network.

TABLE I
Cut-Capacity Assignment for $K = 5$. The Min-Cut Size Is the Minimum Cut Size on $\mathcal{P}_v$

	node cut-capacity	
min-cut size	LUT root	non-LUT root
5	5	6
4	2	3
3	1	3
2	1	5

TABLE II
Initial Circuits for MCNC FSM's and ISCAS Benchmarks

circuit	PI	PO	GATE	FF	Φ	Φ_{FM}
bbara	4	2	28	10	12	4
dk15	3	5	45	2	27	1
dk16	2	3	162	5	36	14
dk17	2	3	42	5	20	2
kirkman	12	6	106	5	13	6
ex1	8	19	140	5	20	8
s1	8	6	107	5	21	7
ssc	7	7	74	4	17	7
keyb	7	2	134	5	24	10
styr	9	10	281	5	45	17
sand	11	9	327	17	44	16
planet1	7	19	348	6	46	19
scf	27	54	516	7	35	14
s9234.1	28	39	1299	135	19	5
s5378	35	49	1503	164	11	4
s15850.1	76	148	3801	515	32	8
s38417	28	106	9817	1464	24	8

as an LUT root. Otherwise, we assign the cut capacity to be ∞. If the min-cut in the new flow network is K-feasible, it corresponds to a zero-cost K-cut on $\mathcal{P}_v$. Otherwise, there does not exist any zero-cost K-cut on $\mathcal{P}_v$. In this case, if the min-cut we computed happens to have cost one, it corresponds to a min-cost K-cut on $\mathcal{P}_v$. If, however, the cost of the min-cut is larger than one, we try to find a lower cost K-cut with one additional K-cut computation by assigning different cut capacities to LUT roots and non-LUT roots based on the min-cut size on $\mathcal{P}_v$, as shown in Table I. The cut capacity is assigned in such a way that a min-cut in the new network corresponds to a min-cost cut on $\mathcal{P}_v$ if the min-cut is K-feasible. For example, suppose on $\mathcal{P}_v$ one min-cut size is three with cost of two, and there exists another cut with a cut size of four and cost of one. Based on the assignment table, the cut capacity of the min-cut is $1 + 2 \cdot 3 = 7$. The cut capacity of the other cut is $3 + 1 \cdot 3 = 6$, however, so it is the min-cut on the new flow network and can be found through max-flow computation. Note that this approach is not guaranteed to find the min-cost K-cut on $\mathcal{P}_v$ because the min-cut on the new flow network may not be K-feasible. For the previous example, if there is also a cut with cut size of $6 > K = 5$ and cost of zero, the cut capacity is six. It is also a min-cut on the new flow network but not K-feasible, and cannot be implemented in one LUT. In the case that the min-cut found on the new flow network is not K-feasible, we keep the min-cut on $\mathcal{P}_v$ for node v.

After getting the low-cost K-cut $(X, \overline{X})$ of every node v, we then try to pack single-output fan-ins of the cut into X (or LUT_v). Our experimental results show that this approach is very efficient (with only two additional max-flow compu-

TABLE III
PERFORMANCE COMPARISON BETWEEN TURBOMAP AND SEQMAPII ON A SPARC5. ENTRY "***" INDICATES THAT SEQMAPII-OPT DID NOT PRODUCE A RESULT AFTER RUNNING 24 H. "GEO-MEAN" LISTS THE GEOMETRIC MEAN OF THE RESULTS. "SUB-MEAN" LISTS THE GEOMETRIC MEAN OF THE RESULTS OF THE FIRST EIGHT EXAMPLES

circuit	TurboMap				SeqMapII-heu				SeqMapII-opt			
	LUT	FF	Φ	CPU	LUT	FF	Φ	CPU	LUT	FF	Φ	CPU
bbara	12	7	3	0.3	19	9	4	7.0	12	9	3	204.6
dk15	7	2	1	0.4	23	3	6	13.0	7	2	1	169.0
dk17	6	3	1	0.4	14	4	4	15.0	6	3	1	303.3
kirkman	53	23	5	1.2	66	22	5	11.7	65	23	5	4341.9
s1	62	11	7	1.7	62	12	7	17.9	62	12	7	14359.5
sse	45	10	6	0.9	50	16	6	14.3	50	16	6	3499.7
ex1	91	21	8	2.8	89	24	8	18.4	89	24	8	31067.6
keyb	80	18	9	2.7	89	14	9	34.6	89	14	9	45867.3
dk16	96	14	14	11.3	113	15	14	140.3	***	***	***	***
styr	166	8	16	10.4	190	16	16	73.7	***	***	***	***
sand	177	31	15	11.5	211	27	15	179.8	***	***	***	***
planet1	205	26	18	19.7	236	42	18	222.9	***	***	***	***
scf	319	25	13	48.9	343	101	13	203.4	***	***	***	***
s9234.1	435	210	4	120.1	499	282	4	274.6	***	***	***	***
s5378	437	285	4	107.1	457	280	4	167.5	***	***	***	***
s15850.1	1148	744	8	512.1	1525	844	8	811.4	***	***	***	***
s38417	3319	2378	6	1994.9	4002	2706	6	7921.1	***	***	***	***
geo-mean	112	30	6	8.9	144	38	8	74.4				
ratio	1	1	1	1	1.28	1.27	1.23	8.38	—	—	—	—
sub-mean	29	9	4	0.97	42	11	6	15	31	10	4	2748.94
ratio	1	1	1	1	1.44	1.13	1.54	15.54	1.05	1.05	1	2841.64

tations) and effective (reducing LUT count by 19% and FF count by 7%).

E. Summary of the TurboMap Algorithm

In the preceding subsections, we have presented three strategies to speed up the label computation of the SeqMapII algorithm [22] and one heuristic method to reduce the area. For a target clock period, our algorithm, named TurboMap, performs SCC partitioning at first. Then, in topological order from the PI's to the PO's, TurboMap computes the node labels for each SCC separately. For each SCC, a number of efficient label update iterations are performed.

To find the minimum clock period, TurboMap performs a binary search using the upper bound of the clock period computed by FlowMap [5] on each combinational subcircuit independently. After getting the minimum clock period and the low-cost K-cut for every node, TurboMap generates the mapping solution and then performs LUT packing [5] and retiming to achieve the minimum clock period [17], [22].

Theorem 5: For a K-bounded sequential circuit with n nodes, the TurboMap algorithm produces a K-LUT mapping solution with the minimum clock period under retiming in $O(K^2 n_l n \mid P_v \mid \log n)$ time, where n_l is an upper bound on the labeling iteration number and $\mid P_v \mid$ is an upper bound on the numbers of nodes in the partial flow networks.

Proof: Each K-cut computation on the partial flow network takes $O(K^2 \mid \mathcal{P}_v \mid)$ time and $O(K \mid \mathcal{P}_v \mid)$ space. Each label update iteration needs n K-cut computation. The label computation for a given target clock period takes $O(K^2 n_l n \mid \mathcal{P}_v \mid)$ run time with $O(K \mid \mathcal{P}_v \mid)$ space requirement. Clearly, the minimum clock period under retiming is less than n. With binary search, the total run time of label computation is $O(K^2 n_l n \mid \mathcal{P}_v \mid \log n)$. There are at most $2n$ low-cost K-cut computations, each of which takes $O(K^2 \mid \mathcal{P}_v \mid)$ run time and in total $O(K^2 n \mid \mathcal{P}_v \mid)$ run time. So the total run time of

TABLE IV
COMPARISON OF THE NUMBER OF NODES OF THE EXPANDED CIRCUITS USED BY TURBOMAP AND SEQMAPII. ENTRY "***" MEANS NO RESULTS AFTER RUNNING THE ALGORITHM FOR 24 H. "GEO-MEAN" LISTS THE GEOMETRIC MEAN OF THE RESULTS. "SUB-MEAN" LISTS THE GEOMETRIC MEAN OF THE RESULTS EXCLUDING THE LAST FOUR EXAMPLES

circuit	NODE	FF	TurboMap		SeqMapII-opt		SeqMapII-heu	
			$\mathcal{P}_{max}$	$\mathcal{P}_{avg}$	$\mathcal{E}^{Kn}_{max}$	$\mathcal{E}^{Kn}_{avg}$	$\mathcal{E}^{6}_{max}$	$\mathcal{E}^{6}_{avg}$
bbara	34	10	68	37	2226	1734	82	44
dk15	53	2	59	28	2524	2490	61	36
dk16	167	5	380	190	49456	47784	196	117
dk17	47	5	94	62	2948	2840	78	49
kirkman	124	5	114	47	9550	8501	77	48
ex1	167	5	165	71	32478	31902	149	92
s1	121	5	135	65	15100	13629	124	64
ssc	88	4	89	42	9457	9160	108	67
keyb	143	5	161	63	37435	32028	164	91
styr	300	5	252	112	179268	167516	227	123
sand	347	17	220	100	167164	158678	229	116
planet1	374	6	364	175	198261	196171	245	145
scf	597	7	808	349	447506	431549	284	155
s9234.1	1360	135	669	169	***	***	101	57
s5378	1587	164	439	90	***	***	106	40
s15850.1	4013	515	1482	202	***	***	110	46
s38417	9897	1464	1176	133	***	***	119	58
geo-mean			246	91		—	130.9	71.1
ratio			1	1		—	0.53	0.78
sub-mean			168	79	26761.5	24920.6	138.6	79.4
ratio			1	1	159.4	314.8	0.8	1.0

TurboMap is $O(K^2 n_l n \mid \mathcal{P}_v \mid \log n)$ with $O(K \mid \mathcal{P}_v \mid)$ space requirement. □

In practice, $n_l \ll n$ and $\mid \mathcal{P}_v \mid \leq n$. TurboMap can be finished in $O(k^2 n^3 \log n)$ time. Since the original SeqMapII algorithm takes $O(K^3 n^5 \log(Kn^2) \log n)$ time and $O(K^2 n^2)$ space to compute optimal mapping solutions with the minimum clock period under retiming [22], TurboMap is about $O(Kn^2 \log(Kn^2))$ times faster with $O(Kn)$ times less memory. In fact, our monotone property (first presented in [10]) was also adopted by the authors of SeqMapII to improve its performance in a later publication [24].

TABLE V
COMPARISON OF TURBOMAP WITH FLOWMAP AND CUTMAP PLUS RETIMING. "GEO-MEAN" LISTS THE GEOMETRIC MEAN OF THE RESULTS

circuit	TurboMap				FlowMap+Retiming				CutMap+Retiming			
	LUT	FF	Φ	CPU	LUT	FF	Φ	CPU	LUT	FF	Φ	CPU
bbara	12	7	3	0.3	13	9	4	0.4	13	9	4	0.5
dk15	7	2	1	0.4	7	2	1	0.5	7	2	1	0.5
dk16	96	14	14	11.3	101	5	14	1.0	91	5	14	12.2
dk17	6	3	1	0.4	10	5	2	0.4	9	3	2	0.5
kirkman	53	23	5	1.2	48	5	6	0.7	45	5	6	1.3
ex1	91	21	8	2.8	83	5	8	0.9	81	5	8	2.1
s1	62	11	7	1.7	58	5	7	0.6	56	5	7	1.2
sse	45	14	6	0.9	42	4	7	0.6	40	4	7	0.9
keyb	80	18	9	2.7	75	5	10	0.8	72	5	10	5.2
styr	166	8	16	10.4	163	5	17	2.0	153	5	17	13.3
sand	177	31	15	11.5	176	17	16	2.5	164	17	16	36.3
planet1	205	26	18	19.7	213	6	19	2.6	189	6	19	18.1
scf	319	25	13	48.9	325	7	14	3.4	306	7	14	23.1
s9234.1	435	210	4	120.1	468	139	5	10.9	433	148	5	60.9
s5378	437	285	4	107.1	473	301	4	8.4	454	295	4	17.1
s15850.1	1148	744	8	512.1	1245	512	8	45.5	1070	512	8	160.9
s38417	3319	2378	6	1994.9	3594	1464	8	74.9	2880	1464	8	257.9
geo-mean	112	30	6	54.8	116	15.2	7	2.1	108	14.8	7	6.9
ratio	1	1	1	1	1.04	0.51	1.14	0.04	0.96	0.49	1.14	0.13

V. EXPERIMENTAL RESULTS

The TurboMap algorithm has been implemented in the C programming language on Sun SPARC workstations and incorporated into the SIS package [26] and the RASP FPGA synthesis package [9]. The test set includes 13 MCNC finite state machines (FSM's) and four large ISCAS benchmarks as used in [10]. The initial gate-level circuits for technology mapping are shown in Table II. Columns PI and PO list the numbers of primary inputs and primary outputs, respectively, of the circuits. Columns GATE and FF list the numbers of gates and FF's, respectively, in the circuits. Column Φ lists the clock periods of the circuits after retiming but before technology mapping.

The experiments were run on a SUN SPARC5 workstation with 96 MB memory. K is set to be five. FlowMap followed by retiming is performed to get an upper bound (shown in column Φ_{FM} in Table II) of the clock period for each example. LUT packing [5] and retiming are performed as postprocessings to get the final mapping solutions. Table III shows the comparison of TurboMap with SeqMapII [22]. SeqMapII has a parameter selecting i for $\mathcal{E}_v^i$. We choose $i = Kn$ (SeqMapII-opt, which guarantees the optimal solution) and $i = 6$ (SeqMapII-heu, which was used in the experiments by Pan and Liu [22] as a heuristic method), respectively, for each example. Columns LUT and FF list the numbers of LUT's and FF's, respectively, in the final solutions. The Φ columns list the minimum clock periods of the final solutions. The CPU columns list the CPU time in seconds. Note that both TurboMap and SeqMapII-opt can obtain mapping solutions with the minimum clock periods under retiming, but TurboMap is 2.8 $\times$ 10^3 times faster. Moreover, TurboMap is more than 8 times faster than SeqMapII-heu, which may generate suboptimal solutions.[8] Compared with SeqMapII-heu, TurboMap produces mapping solutions with 23% smaller clock periods, 28% fewer LUT's, and 27% fewer FF's.

[8]Note that SeqMapII forks a child process to perform the max-flow computation. The CPU time listed under SeqMapII includes this portion as well, which may differ much with the CPU time listed in [22].

To show the effect of our K-cut computation on partial flow networks, we compare the numbers of nodes of the partial flow networks with those of $\mathcal{E}_v^{Kn}$ and $\mathcal{E}_v^6$ used in SeqMapII [22]. The results are shown in Table IV. The column NODE lists the number of nodes in the retiming graph for each example, which is the sum of the numbers of the PI's, the PO's, and the gates. The column FF lists the numbers of FF's in the original circuits. The columns with subscripts $\max$ and avg list the maximum and average numbers of nodes, respectively, of the partial flow networks or the expanded circuits over all nodes in the original circuits and generated by each algorithm. For the last four examples, $\mathcal{E}_v^{Kn}$ cannot be generated due to either time or space limitations. The results show that the average sizes of the partial flow networks are only slightly larger than $\mathcal{E}_v^6$ and 314 times smaller than $\mathcal{E}_v^{Kn}$. This result, together with our efficient label update and SCC partitioning with DFS ordering, provides an explanation of why TurboMap is significantly faster than SeqMapII-opt.

Table V shows the comparison of TurboMap with the conventional design flow of using FlowMap [5] or CutMap [7] for mapping each combinational circuit independently followed by optimal retiming, whose results are listed in columns "FlowMap+Retiming" and "CutMap+Retiming," respectively. The results show that TurboMap can reduce the clock periods by 14% on average compared with both methods, with 4% fewer LUT's as compared with "FlowMap+retiming" but 4% more LUT's as compared with "CutMap+retiming." TurboMap also uses 49% more FF's to reduce the clock period. Note that the number of FF's will not affect area significantly because it is usually much less than the number of LUT's in a mapping solution. The final PLB count of FPGA's will be determined by the number of LUT's.

VI. CONCLUSIONS AND FUTURE WORK

We presented an improved algorithm, named TurboMap, for technology mapping with retiming for optimal clock periods. We proved the monotone property of node labels. Three strategies are used to enhance the performance of SeqMapII, i.e., efficient label update with single K-cut computation, much smaller partial flow networks, and SCC partitioning and DFS ordering. Area reduction is also considered. The experimental results show that TurboMap is about 2.8 $\times$ 10^3 times faster than SeqMapII in computing optimal solutions. TurboMap is even 8 times faster than SeqMapII-heu heuristic algorithm. As a result, we conclude that optimal mapping for minimum clock period under retiming can be carried out efficiently for large circuits in practical use. Furthermore, there is no area overhead compared to conventional approaches to sequential circuits (FlowMap [5] or CutMap [7] followed by retiming). In our future work, we want to develop an efficient algorithm to compute the initial state of the mapping solution and combine resynthesis and pipelining techniques to further reduce the clock periods. We also want to investigate the open issues in this work of whether $\mid \mathcal{P}_v \mid = O(n)$ and whether the label computation can converge in $O(n)$ time in the worst case.

Acknowledgment

The authors are very grateful to Prof. P. Pan and Prof. C. L. Liu for providing the SeqMapII program for the comparative study. The software donation from Synopsys is also gratefully acknowledged.

References

[1] Altera, "Flex 8000 and Flex 10000 Programmable Logic Device family data sheets," 1995.
[2] R. K. Brayton, R. Rudell, A. L. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A multiple-level logic optimization system," *IEEE Trans. Computer-Aided Design*, vol. 6, no. 6, pp. 1062–1081, 1987.
[3] K. C. Chen, J. Cong, Y. Ding, A. B. Kahng, and P. Trajmar, "DAG-map: Graph-based FPGA technology mapping for delay optimization," *IEEE Design Test Comput. Mag.*, pp. 7–20, 1992.
[4] J. Cong and Y. Ding, "On area/depth trade-off in LUT-based FPGA technology mapping," in *Proc. 30th ACM/IEEE Design Automation Conf.*, 1993, pp. 213–218.
[5] ——, "FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE Trans. Computer-Aided Design*, vol. 13, no. 1, pp. 1–12, 1994.
[6] ——, "Combinational logic synthesis for SRAM based field programmable gate arrays," *ACM Trans. Design Automation Electron. Syst.*, vol. 1, no. 2, pp. 145–204, 1996.
[7] J. Cong and Y.-Y. Hwang, "Simultaneous depth and area minimization in LUT-based FPGA mapping," in *Proc. ACM 3rd Int. Symp. Field Programmable Gate Arrays*, 1995, pp. 68–74.
[8] ——, "Structural gate decomposition for depth-optimal technology mapping in LUT-based FPGA design," in *Proc. 33rd ACM/IEEE Design Automation Conf.*, 1996, pp. 726–729.
[9] J. Cong, J. Peck, and Y. Ding, "RASP: A general logic synthesis system for SRAM-based FPGA's," in *Proc. ACM 4th Int. Symp. FPGA*, 1996, pp. 137–143.
[10] J. Cong and C. Wu, "An improved algorithm for performance optimal technology mapping with retiming in LUT-based FPGA design," in *Proc. IEEE Int. Conf. Computer Design*, 1996, pp. 572–578.
[11] T. H. Cormen, C. H. Leiserson, and R. L. Rivest, *Introduction to Algorithms.* The MIT Press, 1990, ch. 27, p. 597.
[12] ——, *Introduction to Algorithms.* Cambridge, MA: MIT Press, 1990, ch. 23, pp. 488–493.
[13] R. J. Francis, J. Rose, and K. Chung, "Chortle: A technology mapping program for lookup table-based field programmable gate arrays," in *Proc. 27th ACM/IEEE Design Automation Conf.*, 1990, pp. 613–619.
[14] R. J. Francis, J. Rose, and Z. Vranesic, "Chortle-crf: Fast technology mapping for lookup table-based FPGA's," in *Proc. 28th ACM/IEEE Design Automation Conf.*, 1991, pp. 613–619.
[15] ——, "Technology mapping of lookup table-based FPGA's," in *Proc. IEEE Int. Conf. CAD*, 1991, pp. 568–571.
[16] C. E. Leiserson, F. M. Rose, and J. B. Saxe, "Optimizing synchronous circuitry by retiming (preliminary version)," in *Proc. 3rd Caltech Conf. Very Large Scale Integration*, R. Bryant, Ed. Rockville, MD: Caltech, Computer Science Press, 1983, pp. 87–116.
[17] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, 1991, vol. 6, pp. 5–35.
[18] AT&T Microelectronics, *AT&T Field-Programmable Gate Arrays Data Book*, 1995.
[19] R. Murgai, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Sequential synthesis for table lookup programmable gate arrays," in *Proc. 30th ACM/IEEE Design Automation Conf.*, 1993, pp. 224–229.
[20] R. Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Improved logic synthesis algorithms for table lookup architectures," in *IEEE International Conference on CAD*, 1991, pp. 564–567.
[21] ——, "Performance directed synthesis for table lookup programmable gate arrays," in *Proc. IEEE International Conf. CAD*, 1991, pp. 572–575.
[22] P. Pan and C. L. Liu, "Optimal clock period FPGA technology mapping for sequential circuits," in *Proc. 33rd ACM/IEEE Design Automation Conf.*, 1996, pp. 720–725.
[23] ——, "Technology mapping of sequential circuits for LUT-based FPGA's for performance," in *Proc. ACM/SIGDA Int. Symp. FPGA's*, 1996, pp. 58–64.
[24] ——, "Optimal clock period FPGA technology mapping for sequential circuits," *ACM Trans. Design Automation Electron. Syst.*, vol. 3, no. 3, 1998. [Online]. Available WWW: http://www.acm.org/todaes/V3N3/L166/paper.ps.gz.
[25] M. Schlag, J. Kong, and P. K. Chan, "Routability-driven technology mapping for lookup table-based FPGA's," *IEEE Trans. Computer-Aided Design*, vol. 13, no. 1, pp. 13–26, 1994.
[26] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Electronics Research Laboratory, University of California, Berkeley, Memo. UCB/ERL M92/41, 1992.
[27] N. Togawa, M. Sato, and T. Ohtsuki, "Maple: A simultaneous technology mapping, placement, and global routing algorithm for field-programmable gate arrays," in *Proc. 31st ACM/IEEE Design Automation Conf.*, 1994, pp. 156–163.
[28] H. Touati, N. Shenoy, and A. Sangiovanni-Vincentelli, "Retiming for table-lookup field-programmable gate arrays," in *Proc. FPGA'92*, pp. 89–94.
[29] U. Weinmann and W. Rosenstiel, "Technology mapping for sequential circuits based on retiming techniques," in *Proc. Eur. Design Automation Conf.*, 1993, pp. 318–323.
[30] Xilinx, *The Programmable Logic Data Book.* San Jose, CA: Xilinx, 1997.

Jason Cong received the B.S. degree in computer science from Peking University, P.R.C., in 1985 and the M.S. and Ph.D. degrees in computer science from The University of Illinois at Urbana-Champaign in 1987 and 1990, respectively.

Currently, he is a Professor and Codirector of the VLSI CAD Laboratory in the Computer Science Department of the University of California, Los Angeles. His research interests include layout synthesis and logic synthesis for high-performance low-power very-large-scale-integration (VLSI) circuits, design and optimization of high-speed VLSI interconnects, field programmable gate array (FPGA) synthesis, and reconfigurable computing. He has published more than 100 research papers and led more than 20 research projects supported by the Defense Advanced Research Project Agency, National Science Foundation (NSF), and a number of industrial sponsors in these areas. He was the General Chair of the Fourth ACM/SIGDA Physical Design Workshop, the Program Chair and General Chair of the 1997 and 1998 International Symposium on FPGA's, respectively, and on program committees of many VLSI CAD conferences, including DAC, ICCAD, and ISCAS. He is an Associate Editor of *ACM Transactions on Design Automation of Electronic Systems*.

Dr. Cong received the Best Graduate Award from Peking University in 1985 and the Ross J. Martin Award for Excellence in Research from The University of Illinois at Urbana-Champaign in 1989. He received the NSF Research Initiation Award and NSF Young Investigator Award in 1991 and 1993, respectively. He received the Northrop Outstanding Junior Faculty Research Award from UCLA in 1993 and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS Best Paper Award in 1995. He received the ACM Recognition of Service Award in 1997.

Chang Wu received the B.S. degree from the Computer Science Department of Shanghai Jiao Tong University, Shanghai, P.R.C., in 1986 and the M.S. degree from the Institute of Pattern Recognition and Artificial Intelligence of Shanghai Jiao Tong University in 1989. He is now a Ph.D. student in the Computer Science Department of the University of California, Los Angeles.

From 1989 to 1995, he was with the Beijing Integrated Circuit Design Center as a Senior Software Engineer developing the Panda VLSI CAD System. He was a Visiting Scholar at the University of California, Los Angeles, from 1995 to 1996. His major interests are logic synthesis and technology mapping with retiming for high-performance very-large-scale-integration design.