An Improved Algorithm for Performance Optimal Technology Mapping with Retiming in LUT-Based FPGA Design

Jason Cong and Chang Wu*
Department of Computer Science
University of California, Los Angeles, CA 90024

Abstract

The FPGA technology mapping and synthesis problem for combinational circuits has been well studied. But for sequential circuits, most of the previous synthesis and mapping algorithms assume that the positions of flipflops are fixed and synthesize each combinational block independently. Retiming is a technique to reduce the clock period by repositioning flipflops [LeSa91]. With retiming, the previous mapping algorithms may not lead to an optimal solution of a sequential circuit by mapping each combinational block independently. Pan and Liu [PaLi96b] recently proposed a novel algorithm, named SeqMapII, of technology mapping with retiming for the optimal clock period. The time complexity of their algorithm, however, is $O(K^3n^4 \log(Kn^2) \log n)$ for sequential circuits with n gates and targeting K-LUT based FPGAs, which is too high for medium and large size designs in practice (more than 20 hours of CPU time to get the optimal solution for a design of 134 gates on a SPARC10 workstation). In this paper, we present three strategies to improve the performance of the SeqMapII algorithm. First, we show that the monotone property of labels hold for sequential circuits and develop an efficient label update with single K-cut computation. Second, we propose a new K-cut computation method on partial flow networks, which are very small in practice. Finally, SCC (strongly connected component) partition is performed to eliminate many redundant label computation to further speedup the algorithm. In practice, our algorithm works in $O(K^2n^3 \log n)$ time and $O(Kn)$ space according to our experimental results. It is 2×10^4 times faster than SeqMapII-opt for computing optimal solutions and 2 times faster than SeqMapII-heu which uses very small expanded circuits as a heuristic.

*The work described in this paper was conducted while the author was at UCLA on leave from Beijing Integrated Circuit Design Center.

1. Introduction

FPGA is widely used in ASIC designs and prototyping. The technology mapping and synthesis problem for FPGAs is to produce an equivalent circuit for a given circuit using only the specific programmable logic blocks (PLBs). More specifically, without synthesis, the PLBs in the mapping solution form a cover of gates in the original circuit possibly with overlap. Different FPGAs have different PLBs. In this paper, we consider a generic type of PLB: K-input lookup tables (K-LUTs), which has been widely used in current FPGA technology [Xi94, At95, Al95]. Most of the previous LUT mapping algorithms optimize either area [FrRC90, FrRV91a, MuSB91a] or delay [FrRV91b, MuSB91b, CoDi94]. The algorithms in [CoHw95, CoDi93a] consider both delay and area. The algorithms in [ScKC94, ToSO94] also consider the routability. But most of these approaches apply only to combinational circuits[1]. For sequential circuits, they assume that the positions of flipflops are fixed so that the entire circuit can be partitioned into several combinational subcircuits and mapped independently. The limitation of this assumption is that one cannot consider the effect of mapping on retiming. Retiming [LeSa91, LeRS83] is a technique of moving flipflops without changing the circuit behavior. For single phase clock, edge-triggered flipflops, Leiserson and Saxe [LeSa91, LeRS83] solved the retiming problems to minimize the clock period or the number of flipflops. Shenoy and Rudell [ShRu94] enhanced the performance of their retiming algorithm.

Given a sequential circuit, if we map each combinational subcircuits independently, the optimal solutions for all combinational subcircuits may not lead to an optimal mapping solution for the entire sequential circuit. *Finding the mapping solution with the minimum clock period under retiming* is an interesting and important problem.

Several FPGA synthesis and mapping algorithms have been proposed specifically for sequential circuits. The approach in [MuBS93] does not consider retiming. Its objective is to consider proper packing of LUTs with flipflops to minimize the number of CLBs for Xilinx FPGAs. The methods in [WeRo93, PaLi96a] are heuristics which consider loopless sequential circuits. A significant advancement was made recently by Pan and Liu [PaLi96b]. They proposed a novel algorithm, named SeqMapII, to find a mapping solution with the minimum clock period under retiming. Similar to the FlowMap algorithm [CoDi94], their algorithm works in two phases: the labeling phase and the mapping generation phase (a postprocess-

[1] See the survey paper [CoDi96] for more details.

ing step of retiming is needed to get the final solution). They introduced the idea of the expanded circuit to represent all possible K-LUTs under retiming and node-replication. Iterative computation is used to compute labels of all nodes. The time and space complexities are $O(K^3n^4log(Kn^2)logn)$ and $O(K^2n^2)$, respectively, for a circuit with n gates. Although SeqMapII runs in polynomial time, it has two shortcomings: 1) too many candidate values ($O(Kn^2)$) need considering for each label update, 2) the expanded circuits are too large ($O(Kn^2)$ nodes) to compute the optimal solutions. Experimental results show that the runtime of SeqMapII for computing the optimal solution is too long in practice (more than 20 hours of CPU time to get the optimal solution for a design of 134 gates on a SPARC10)[2].

In this paper, we present three strategies to improve the performance of the label computation in SeqMapII [PaLi96b], which is the most time-consuming step. First, we proved that the monotone property of labels hold for sequential circuits and develop an efficient label update with single K-cut computation 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. The experimental results show that the average numbers of nodes of the partial flow networks are always less than n, which is a big improvement over $O(Kn^2)$. Finally, SCC (strongly connected component) partition is performed to eliminate many redundant label computation and 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. It is 2×10^4 times faster than SeqMapII-opt for computing optimal solutions and, 2 times faster than SeqMapII-heu which uses very small expanded circuits as a heuristic.

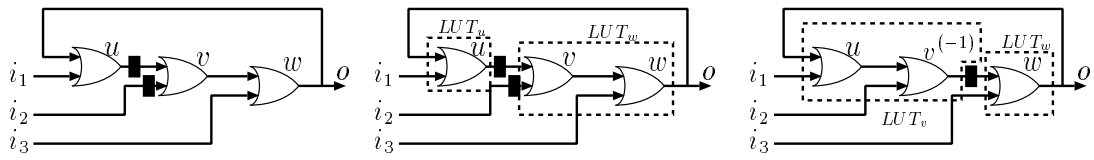
The rest of the paper is organized as the following. Section 2 presents the problem formulation and definitions. Section 3 gives a review of the approach by Pan and Liu [PaLi96b]. Then, our work on the efficient label computation is presented in Section 4.1 and the K-cut computation on partial flow networks is presented in Section 4.2. Followed by the SCC partitioning presented in Section 4.3, we summarize our algorithm, named TurboMap, in Section 4.4. The experimental results are presented in Section 5, followed by conclusions and future work in Section 6.

[2]The authors of SeqMapII observed that its runtime can be reduced by several enhancements of the program. However, they feel that the resulting complexity is still too high for practical designs. Therefore, they turned to develop efficient heuristics based on the idea in SeqMapII [PaLi96c].

2. Problem Formulation and Definitions

Given a *sequential* circuit, the technology mapping problem for K-LUT based FPGAs is to construct an equivalent circuit consisting of K-LUTs and flipflops. In the *unit delay model*, the delay of each LUT is one, the delays of nets are zero. 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.*



(a) original circuit **(b) simple mapping solution** **(c) non-simple mapping solution**

Figure 1. Simple and non-simple mapping solutions.

A mapping solution in which the output signals of all the LUTs are from the original circuit is called a *simple mapping solution* [PaLi96b]. As shown in Figure 1(b), the outputs of LUT_u and LUT_w are the outputs of u and w in the original circuit as shown in Figure 1(a). But the output of LUT_v in Figure 1(c) is one clock cycle ahead of the output of v in the original circuit. Pan and Liu [PaLi96b] proposed that *there is a simple mapping solution whose retimed clock period is equal to the minimum clock period among all mapping solutions.* Furthermore, they transformed the optimization problem into a deterministic problem [PaLi96b]:

Problem 2 *Given a target clock period c, determine if there exists a simple mapping solution whose retimed clock period is c or less.*

Obviously, if Problem 2 can be solved, Problem 1 can be solved by binary search on clock periods as well. As in [CoDi94], the results in [PaLi96a, PaLi96b] and this paper apply to only K-bounded networks, i.e., each gate in the network has at most K fanins[3]. In the remaining of this paper, all the circuits are assumed to be K-bounded.

[3]When a circuit is not K-bounded, we can use the gate decomposition algorithm, such as balanced tree decomposition [BrRS87], dmig [ChCD92] or DOGMA [CoHw96], to decompose the gates with more than K fanins.

We use $G(V, E, W)$ to denote the retiming graph of a sequential circuit, where V is the set of nodes which represents gates in the circuit, E is the set of edges which represents the connection between gates, and W is the set of edge weights [LeSa91]. For an edge e, its weight, denoted $w(e)$, is the number of flipflops on the connection represented by e. The path weight, denoted $w(p)$ for a path p, is the sum of weights of all edges on the path. For a node v, G_v is a subgraph of G consisting all nodes which have paths to v.

For a simple mapping solution M and a given clock period c, the *edge length*, denoted $length(e)$, of an edge e is defined to be $-c \cdot w(e) + 1$. 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 is the maximum length of all paths from the PIs to v in M[4]. It was shown in [PaLi96b] that for a mapping solution M and a given c, *the retimed clock period is c or less iff* $l_M(v) \leq c + 1$ *for every PO* v. For a node v in the original circuit, the label of v, denoted $l^{opt}(v)$, is defined to be the *minimum* of the l-values of the K-LUTs rooted at v among *all* mapping solutions. It was shown that *there is a mapping solution whose retimed clock period is c or less, iff* $l^{opt}(v) \leq c + 1$ *for every PO* v [PaLi96b].

In a directed graph G with one sink and one source, a cut $(X, \overline{X})$ is a partition of the nodes in 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 to one or several nodes in $\overline{X}$. If $|V(X, \overline{X})| \leq K$, a cut $(X, \overline{X})$ is called a K-feasible cut, or K-cut in short.

3. Review of the SeqMapII Algorithm

In this section, we present a review of the SeqMapII algorithm of technology mapping for the optimal clock period under retiming proposed by Pan and Liu [PaLi96b].

To solve Problem 2, the SeqMapII algorithm works in two phases: 1) label computation, 2) mapping generation [PaLi96b].

Their labeling process computes a lower-bound on the value of each node label and repeatedly improve the lower-bounds until there is no further improvement for all the lower-bounds.

The initial lower-bounds for all the PIs are zero and for all the others nodes are $-\infty$. For the current lower-bound $l(v)$ of each node v, Pan and Liu [PaLi96b] presented a procedure IMPROVE(v) to determine a new lower-bound $l_{new}(v)$. They introduced the concept of the expanded circuit for each node v to represent all possible K-LUTs for implementing v

[4] We define $l_M(v) = +\infty$ if there is a path from PIs to v with a loop of positive length in M.

with consideration of retiming and node replication. An expanded circuit $\mathcal{E}_v^i$ at node v with control number i is a DAG rooted at v formed by replication of nodes in G_v and it has the property that all paths from any given node in $\mathcal{E}_v^i$ to the root have the same number of flipflops. A replication of node u is denoted as u^w if it passes w flipflops before reaching the root v in $\mathcal{E}_v^i$. 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 source (without inputs) in $\mathcal{E}_v^i$ that is not a replication of a PI. For example, Figures 2(b,c,d,e) are four expanded circuits $\mathcal{E}_c^i$ of node c in Figure 2(a) with $i = 0, 1, 2, 3$, respectively.

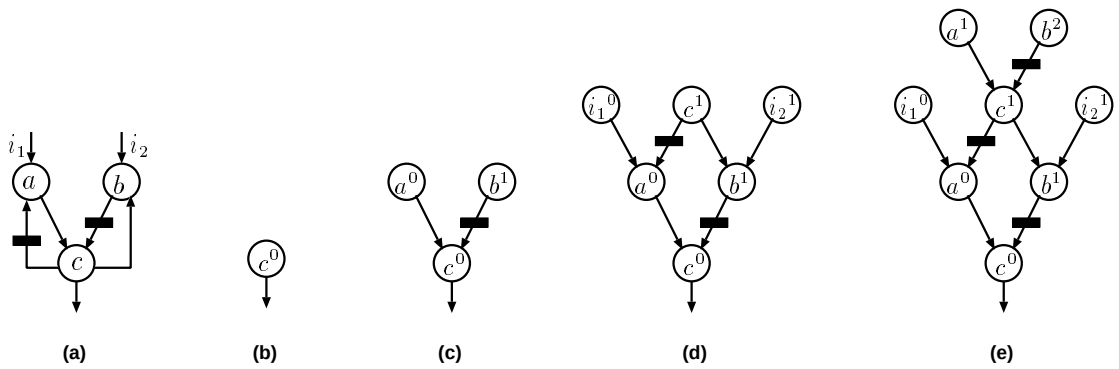


Figure 2. Expanded Circuits

Pan and Liu [PaLi96b] showed that *to examine all K-LUTs for a node v, it sufficed to examine all the K-LUTs 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, i.e., each edge has at most one flipflop. 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 [PaLi96b].

In the expanded circuit of node v, based on the current lower-bounds $l(u)$ of node labels, let the *height* of a K-cut $(X, \overline{X})$ be $h(X, \overline{X}) = \max\{l(u) - c \cdot w + 1 \mid \forall\ u^w \in V(X, \overline{X})\}$. In SeqMapII [PaLi96b], the new lower-bound is computed as follows,

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

This value is determined by binary search among $O(Kn^2)$ possible values in $\{l(u) - c \cdot w + 1 \mid u^w \in \mathcal{E}_v^{Kn}\}$ and performing a K-cut computation for each value. The computation time for every $l_{new}(v)$ is $O(K^3n^2log(Kn^2))$ based on the max-flow min-cut theory. The labels of all nodes can be determined in $O(K^3n^4log(Kn^2))$ time [PaLi96b].

This approach is the first polynomial algorithm to find a mapping solution with the optimal clock period under retiming. But we have observed two shortcomings in this approach. First, the expanded circuit $\mathcal{E}_v^{Kn}$ is very large ($O(Kn^2)$ nodes and $O(K^2n^2)$ edges). Second, there are too many values ($O(Kn^2)$) to be considered when computing the new lower-bound of each node label.

4. 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 show the monotone property of the node labels and develop a new procedure for computing a *tighter* new lower-bound with single K-cut computation. Second, we propose a new approach to compute a K-cut on a partial flow network. The partial flow network is built incrementally during the K-cut computation. The number of nodes in it is always far less than n as shown in our experiments. Third, we use SCC (strongly connected component) partition to eliminate many redundant label computation to further speedup the algorithm.

4.1. Label Update with Single K-Cut Computation

In SeqMapII [PaLi96b], to compute $l_{new}(v)$ for a node, it is necessary to perform binary search among $O(Kn^2)$ possible values in $\{l(u)-c\cdot w+1 \mid u^w \in \mathcal{E}_v^{\infty}\}$, which needs $O(\log(Kn^2))$ K-cut computations. In our approach, we compute a *tighter* lower-bound $l'_{new}(v)$ with single K-cut computation.

Let $\mathcal{L}(v) = max\{l(u)-c\cdot w(e) \mid \forall e(u,v) \in G_v\}$ for a node v. We update the lower-bound 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 only one K-cut computation. Recall this result is similar to Lemma 2 in [CoDi94] which applys to combinational circuits only.

The correctness of our approach is based on the fact that $l_{new}(v) \le l'_{new}(v) \le l^{opt}(v)$, or in other words, $l'_{new}(v)$ is a tighter lower-bound. This can be proved based on the monotone property of node labels. In a sequential circuit which has a mapping solution with the retimed clock period of no more than c, we call the set of its node labels $l^{opt}(v)$ is *monotone* if for any edge $e(u,v) \in G_v$, $l^{opt}(v) \ge l^{opt}(u)-c\cdot w(e)$. We have that:

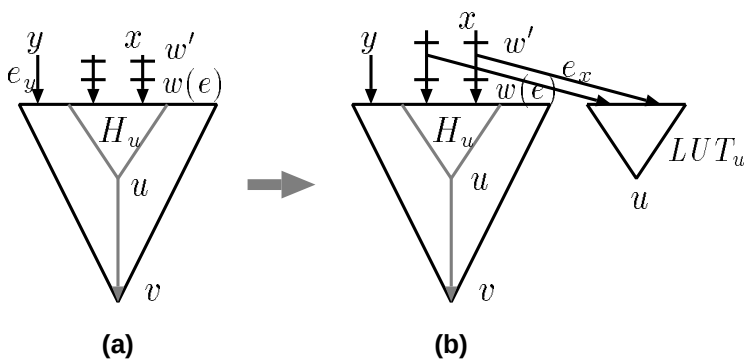


Figure 3. Proof of monotone property of node labels.

Theorem 1 *(Monotone Property) In a sequential circuit which has a mapping solution with the retimed clock period of no more than c, the labels are monotone. That is,* $l^{opt}(v) \geq l^{opt}(u) - c \cdot w(e)$ *for any edge* $e(u, v)$ *in the original circuit.*

Proof: For each edge $e(u, v)$ in the original circuit, there exists a simple mapping solution M such that the l-value $l_M(v)$ of v equals to $l^{opt}(v)$. Let LUT_v denote the K-LUT rooted at v in M. There are two cases:

Case 1: u is a fanin to LUT_v. According to the definition of l-values, the l-values of u and v in M satisfy $l_M(u) - c \cdot w(e) + 1 \leq l_M(v) = l^{opt}(v)$. Since $l_M(u) \geq l^{opt}(u)$, we have that $l^{opt}(v) \geq l_M(u) - c \cdot w(e) + 1 > l^{opt}(u) - c \cdot w(e)$.

Case 2: u is covered inside LUT_v. For edge $e(u, v)$ in the original circuit, the $w(e)$ flipflops must be pushed back to LUT_v's fanins in M as shown in Figure 3(a). (Note that they cannot be pushed down to the output of LUT_v as M is a simple mapping solution.) Let H_u be the sub-DAG rooted at u inside LUT_v. Clearly, H_u can be covered by a K-LUT. By replicating H_u explicitly outside LUT_v to form LUT_u, we can get another simple mapping solution M' as shown in Figure 3(b). Note that for any input node x to H_u, the edge (x, LUT_v) has a weight $(w(e) + w')$, because $w(e)$ flipflops were pushed back to this edge. Refer to Figure 3(b), the weight of edge $e_x(x, LUT_u)$ is w' to make M' a simple mapping solution, because the $w(e)$ flipflops are on the output of u in the original circuit. So the l-value $l_{M'}(u)$ in M' is

$l_{M'}(u) = max\{l_{M'}(x) - c \cdot w' + 1 \mid x \text{ is an input to } LUT_u \text{ in } M'\} = max\{l_M(x) - c \cdot w' + 1\}$.

Since every input of LUT_u is an input of LUT_v,

$$\begin{aligned} l^{opt}(v) &= l_M(v) = max\{l_M(y) - c \cdot w(e_y) + 1 \mid y \ is \ an \ input \ to \ LUT_v\} \\ &\geq \max\{l_M(x) - c \cdot (w' + w(e)) + 1 \mid x \ is \ an \ input \ to \ H_u \ of \ LUT_v\} \\ &= \max\{l_M(x) - c \cdot w' + 1 \mid x \ is \ an \ input \ to \ H_u \ of \ LUT_v\} - c \cdot w(e) \\ &= l_{M'}(u) - c \cdot w(e) \\ &\geq l^{opt}(u) - c \cdot w(e). \end{aligned}$$

This concludes that $l^{opt}(v) \geq l^{opt}(u) - c \cdot w(e)$ 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). Based on Theorem 1, we can prove that:

Theorem 2 *For a sequential circuit which has a mapping solution with the retimed clock period of no more than c, the property $l_{new}(v) \leq l'_{new}(v) \leq l^{opt}(v)$ holds at any 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, we have that $l(v) = -\infty < l^{opt}(v)$. Now suppose that $l(v) \leq l^{opt}(v)$ holds for every node v at the current iteration, we prove that $l'_{new}(v) \leq l^{opt}(v)$ for every node v at any computation order of nodes.

Based on the condition that $l(v) \leq l^{opt}(v)$, we have that:

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

Case 1: If there is a K-cut $\mathcal{E}_v$ 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 $\mathcal{E}_v$ with $h(X, \overline{X}) \leq \mathcal{L}(v)$, then $l'_{new}(v) = \mathcal{L}(v) + 1$. To prove $l^{opt}(v) \geq l'_{new}(v)$, let us prove that it is impossible for $l^{opt}(v) < l'_{new}(v)$. Since we proved that $\mathcal{L}(v) \leq l^{opt}(v)$, if $l^{opt}(v) < l'_{new}(v) = \mathcal{L}(v) + 1$, it must be that $l^{opt}(v) = \mathcal{L}(v)$.

Suppose M is the 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 $\overline{X} = LUT_v$. So we have that $l^{opt}(v) = l_M(v) = \max\{l_M(u) - c \cdot w + 1 \mid \forall \ u^w \in V(X, \overline{X})\}$. From the definition of $l^{opt}(u)$ and the assumption that $l(u) \leq l^{opt}(u)$ for every node u, it must be that

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

It means that the height $h(X, \overline{X}) = \max\{l(u) - c \cdot w + 1\} \leq \mathcal{L}(v)$. The result is contradict with the assumption that there is no K-cut in $\mathcal{E}_v$ with height of no more than $\mathcal{L}(v)$. So we have that $l^{opt}(v) \geq \mathcal{L}(v) + 1$ when $l'_{new}(v) = \mathcal{L}(v) + 1$.

Now we can conclude that $l_{new}(v) \leq l'_{new}(v) \leq l^{opt}(v)$ for every node v at any iteration. ■

With the computation of the tighter lower-bound $l'_{new}(v)$ in our algorithm and based on Theorem 2 in [PaLi96b], we have the following result:

Corollary 1 *For a sequential circuit with n nodes and a target clock period c, the labels of all nodes can be computed in $O(K^3n^4)$ time.*

Since $\mathcal{E}_v^{Kn}$ has $O(Kn^2)$ nodes and $O(K^2n^2)$ edges, to compute each $l_{new}(v)$ in SeqMapII takes $O(K^3n^2 \log(Kn^2))$ time [PaLi96b]. But, to compute each $l'_{new}(v)$ in our algorithm takes only $O(K^3n^2)$ time and $l'_{new}(v)$ will converge to $l^{opt}(v)$ with no more iterations than $l_{new}(v)$ will. So the label computation time of our algorithm is $O(K^3n^4)$.

4.2. K-Cut Computation on Partial Flow Networks

In this subsection, we present a new approach to compute $l'_{new}(v)$ for every node v on a partial flow network, which is much smaller than $\mathcal{E}_v^{Kn}$ but still guarantees the optimal solutions. To check whether $l'_{new}(v) \leq \mathcal{L}(v)$ with the approach in SeqMapII [PaLi96b], we need to build the $\mathcal{E}_v^{Kn}$ and construct the corresponding flow network and then, perform the K-cut computation. In TurboMap, to check whether $l'_{new}(v) \leq \mathcal{L}(v)$, we construct the flow network incrementally without constructing the entire $\mathcal{E}_v^{Kn}$. And more important, we construct the flow network *just* large enough to detect whether a K-cut exists. Recall that all the previous max-flow computation based FPGA mapping algorithms [CoDi94, CoHw95, PaLi96b, PaLi96a] build the entire expanded circuit or the flow network before the max-flow computation to test the existence of a K-cut.

The basic idea of our algorithm is that although the flow network for $\mathcal{E}_v^{Kn}$ is very large, the union of the first $K+1$ *shortest* augmenting paths are usually very small as compared with $\mathcal{E}_v^{Kn}$ (refer to the light shaded area vs. the entire flow network shown in Figure 5). If we start from v^0 to search for the shortest augmenting paths to the sink, only a small portion of $\mathcal{E}_v^{Kn}$ will be searched. So if we start from v^0 and grow the flow network during the K-cut computation *incrementally*, the flow network would be very small as compared with $\mathcal{E}_v^{Kn}$. We call it the *partial flow network*.

Now, let us define the critical nodes in the expanded circuit or the corresponding flow network. For $\mathcal{L}(v)$ based on the current lower-bound, a node u^w in the expanded circuit or

the corresponding flow network is *critical* if $l(u) - c \cdot w \geq \mathcal{L}(v)$. Otherwise, it is not critical. Let $\mathcal{E}_f$ be the partial flow network. As shown in Figure 4(c), the edge direction is reversed from that in graph G_v (shown in Figure 4(a)). The root v^0 is the source of $\mathcal{E}_f$ and all the u^w for the PIs u in the original circuit will be connected to the sink t.

Initially, we have only the source v^0 in the partial flow network. We go from v^0 along the fanin edges of v in G_v to search for the shortest augmenting paths to the sink based on the breadth first search (BFS). Suppose u^w is the current node during the BFS search. For each fanin edge $e(a, u)$ of u in G_v, we create a pair of nodes $a^{w+w(e)}$ and $a_1^{w+w(e)}$ if they have not been created before, and assign edge $(a^{w+w(e)}, a_1^{w+w(e)})$ with flow capacity ∞ if $a^{w+w(e)}$ is a critical node, or 1 otherwise. Then we connect u^w to $a^{w+w(e)}$ with flow capacity ∞. If a is a PI in the original circuit, we also connect $a_1^{w+w(e)}$ to the sink t with flow capacity ∞. When one shortest augmenting path is found, we augment this path and start from v^0 again to search another shortest augmenting path until there is no more augmenting path (in this case, we assign $l'_{new}(v) = \mathcal{L}(v)$) or we find the $(K+1)$-th augmenting path (in this case, a K-cut does not exists and we assign $l'_{new}(v) = \mathcal{L}(v) + 1$).

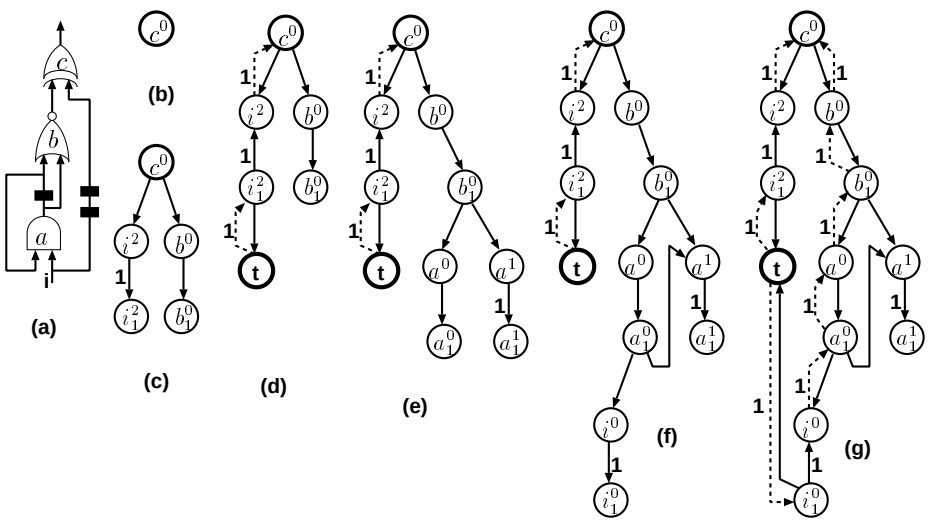


Figure 4. Incremental Construction of the Partial Flow Network with K-Cut Computation. The source is c^0, the sink is t. Default flow capacity of the edge is ∞.

Let us look at the example of constructing the partial flow network for node c in G_c shown in Figure 4(a). The node i is a primary input. Each black bar represents a flipflop. For

$K = 3$ and $c = 1$, suppose the labels $l^{opt}(a) = l^{opt}(b) = 1$ are known and $\mathcal{L}(c) = 1$, we want to decide whether $l'_{new}(c) = \mathcal{L}(c)$ or $\mathcal{L}(c)+1$. The computation is shown step by step in Figures 4(b,c,d,e,f,g). At first, we create c^0 as the source and put it to a FIFO (first-in-first-out) queue Q. Then, getting c^0 from Q, for edges $(b, c), (i, c) \in G_c$, we create nodes b^0, i^2 and edges $(c^0, b^0), (c^0, i^2)$ with flow capacity ∞ in the flow network, and put b^0, i^2 to Q. In the next two steps, we will get b^0, i^2 successively from Q and split them by adding two more nodes b^0_1, i^2_1 and two more edges $(b^0, b^0_1), (i^2, i^2_1)$. Since $l(b) - c \cdot 0 = \mathcal{L}(c) = 1$, the flow capacity of edge (b^0, b^0_1) is ∞. The flow capacity of edge (i^2, i^2_1) is 1 as shown in Figure 4(c). As i is a PI, we add a new edge (i^2_1, t) to the sink t with flow capacity of ∞ and claim finding one shortest augmenting path from the source c^0 to the sink t. After augmenting this path, we clear Q and start from c^0 again to search another path. The residue graph is shown in Figure 4(d). Since b^0_1 was just created in the previous step, we need to build the fanout edges of b^0_1 and add two pairs of nodes a^0, a^0_1 and a^1, a^1_1 when BFS search reachs b^0_1. The new network is shown in Figure 4(e). In the next step, we will build a^0_1 and add two nodes i^0, i^0_1 as shown in Figure 4(f).Since i is a PI, we find another augmenting path. The new residue graph is shown in Figure 4(g).

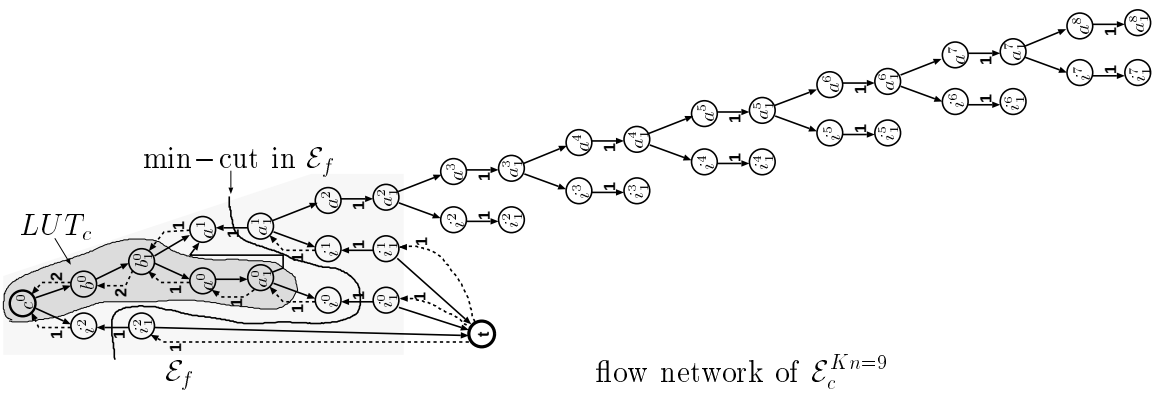


Figure 5. K-Cut Computation on the Partial Flow Network.

As shown in Figure 5, after finding 3 augmenting paths, there is no more in the residue graph. So 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 $\mathcal{E}_f$ we generated incrementally, which corresponds to the expanded circuit $\mathcal{E}^3_c$ in this example. As shown in Figure 5, $\mathcal{E}_f$ is much smaller than the flow network of $\mathcal{E}_c^{Kn=9}$. For the min-cut $(X_f, \overline{X_f})$ in $\mathcal{E}_f$, where the source c^0 is in X_f and the

sink t is in $\overline{X_f}$, the corresponding DAG X, shown as the heavy shaded area, of X_f in $\mathcal{E}_c^{Kn}$ forms a K-LUT rooted at c as shown in Figure 5.

Since we only search the first $(K+1)$ shortest augmenting paths, the incrementally constructed flow network will be much smaller than the flow network of $\mathcal{E}_v^{Kn}$. 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 for all the examples we tested. So practically, each label update takes only $O(K^2n)$ time and $O(Kn)$ space in TurboMap and, the overall label computation takes $O(K^2n^3)$ time and $O(Kn)$ space.

4.3. SCC Partition

A strongly connected component (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, we have both paths $u \rightsquigarrow v$ and $v \rightsquigarrow u$ [CoLR90]. Since the node labels are the minimum l-values which is the longest path from the PIs to the nodes, the labels of two nodes u and v are mutual dependent if they are in one SCC. But if they are in different SCCs, for example, suppose $u \rightsquigarrow v$ but $v \not\rightsquigarrow u$, the label of v will dependent on the label of u, but the label of u will not dependent on the label of v. So the label of u can be computed before v. The algorithm introduced in [CoLR90] can find all the SCCs and sort them in a topologic order from the PIs to the POs. In TurboMap, the nodes in one SCC will be labeled together with the iterative label computation stated in the preceding two subsections. For different SCCs, the labels will be computed separately in the topologic order of SCCs from the PIs to the POs. For one SCC, since the node labels of the preceding SCCs have been computed beforehand, the lower-bounds of this SCC will converge to the labels faster than all the node labels of the preceding SCCs are unknown. Our results show that with SCC partitioning, the computation time of TurboMap can be reduced by 50% in average.

4.4. Summary of TurboMap Algorithm

In the preceding subsections, we have presented three strategies to speedup the label computation of SeqMapII algorithm [PaLi96b]. For each given clock period, our algorithm, named TurboMap, perform SCC partition at first. Then, for the SCCs in the topologic order from the PIs to the POs, TurboMap computes the node labels for each SCC component separately according to the topological order. For each SCC, a number of label update iterations are performed. In each iteration, the efficient label update for each node v with

single K-cut computation on the partial flow network is used to compute the new lower-bound $l'_{new}(v)$.

To find the minimum clock period, TurboMap performs binary search using the upper-bound of the clock period computed by FlowMap [CoDi94] on each combinational subcircuit independently. After getting the minimum clock period, TurboMap will generate the mapping solution and then, perform the postprocessing of LUT-packing [CoDi94] and retiming to minimize the number of flipflops [LeSa91].

Based on the above discussions, we have that:

Theorem 3 *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 $O(K^2n^3 \log n)$ time and $O(Kn)$ space in practice.*

Proof: Based on the result in [PaLi96b], the lower-bounds of node labels based on $l_{new}(v)$ will converge to $l^{opt}(v)$ in n iterations. Since $l'_{new}(v)$ computed in TurboMap is a tighter lower-bound than $l_{new}(v)$, $l'_{new}(v)$ will converge to $l^{opt}(v)$ in n iterations as well.

Since the partial flow network has only $O(n)$ nodes and $O(Kn)$ edges in practice, as shown in our experiments, the K-cut computation on the partial flow network takes $O(K^2n)$ time and $O(Kn)$ space. So the label computation for a given target clock period take $O(K^2n^3)$ time and $O(Kn)$ space. Clearly, the minimum clock period under retiming is less than n. With binary search, the total runtime of label computation is $O(K^2n^3 \log n)$. Since the mapping generation takes only $O(Kn)$ time, the total runtime of TurboMap is $O(K^2n^3 \log n)$ with $O(Kn)$ space in practice. ■

Note that the SeqMapII algorithm takes $O(K^3n^4 \log(K^2n) \log n)$ time and $O(K^2n^2)$ space to compute the optimal mapping solution with the minimum clock period under retiming [PaLi96b], TurboMap is $O(Kn \log(K^2n))$ times faster with $O(Kn)$ times less space.

5. Experimental Results

The TurboMap algorithm has been implemented in C language on Sun SPARC workstations and incorporated into SIS package [SeSL92]. For the ease of verification, we choose 13 MCNC FSMs which have the property that there exists an input sequence for the FSM starting from the initial state and getting back to this initial state again. This property guarantees that there is an equivalent initial state for any retiming [ToBr93, LeSa81]. Therefore, we can compute the new initial state for the mapping solution and compare the I/O behavior to verify our results. To generate the gate level circuits for technology mapping, the

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
sse	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

Table 1. Initial circuits for MCNC FSMs and ISCAS Benchmarks.

following commands are performed within SIS: *state_minimize stamina, state_assign jedi, source sis_lib/script, full_simplify, read_library lib2.genlib, read_library -a lib2_latch.genlib, map -s, sweep*. The initial gate level circuits for technology mapping consist *inverter, 2/3-input nand/nor gates* and *edge triggered flipflops*. To test the capability of TurboMap for large circuits, we choose 4 large ISCAS'89 benchmarks. The gate decomposition algorithm *dmig* [ChCD92] are performed to decompose the circuits into 2-bounded circuits. The resulting circuits are shown in Table 1. 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 flipflops, respectively, in the circuits. Column Φ lists the clock periods of the circuits after retiming, but before technology mapping.

Our tests are performed on a SPARC10 workstation with 64MB memory. K is set to 5. FlowMap followed by retiming is performed to get a upper-bound (shown in Column Φ_{FM} in Table 1) of the clock period for each example. LUT-packing [CoDi94] and retiming [LeSa91] are performed as postprocessings to get the final solutions. Table 2 shows the comparison of TurboMap with SeqMapII [PaLi96b]. SeqMapII has a parameter to set 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 [PaLi96b] as a heuristic method),

circuit	TurboMap				SeqMapII-opt				SeqMapII-heu			
	LUT	FF	Φ	CPU	LUT	FF	Φ	CPU	LUT	FF	Φ	CPU
bbara	13	9	3	0.5	12	9	3	180.8	19	9	4	2.8
dk15	7	2	1	0.3	7	2	1	155.1	23	3	6	4.6
dk17	8	5	1	0.4	6	3	1	293.0	14	4	4	7.3
kirkman	57	17	5	0.7	65	23	5	5373.3	66	22	5	4.8
ex1	93	16	8	1.6	89	24	8	55653.0	89	24	8	9.1
s1	60	7	7	0.9	62	12	7	22435.7	62	12	7	12.6
sse	52	12	6	0.4	50	16	6	4508.4	50	16	6	6.6
keyb	80	13	9	1.5	89	14	9	78534.4	89	14	9	17.4
dk16	104	18	14	11.2				***	113	15	14	71.5
styr	173	10	16	8.4				***	190	16	16	36.6
sand	179	36	15	13.8				***	211	27	15	96.4
planet1	222	26	18	20.3				***	236	42	18	118.9
scf	342	26	13	32.5				***	343	101	13	101.9
s9234	525	238	4	102.0				***	499	282	4	226.6
s5378	490	292	4	92.3				***	456	280	4	145.1
s15850.1	1541	811	8	511.5				***	1525	844	8	732.4
s38417	3988	2552	6	4056.8				***	4002	2706	6	9599.7
total	7934	4090	138	4855.2	—	—	—	—	7988	4417	147	11595.1
ratio	1	1	1	1	—	—	—	—	1.01	1.08	1.07	2.39
subtotal	370	81	32	4.9	380	103	32	111480.7	412	104	41	108
ratio	1	1	1	1	1.03	1.27	1	22844.4	1.11	1.28	1.28	22.13

Table 2. Performance comparison between TurboMap with SeqMapII on a SPARC10. Entry "*" indicates that SeqMapII-opt did not produce a result after running 24 hours. Subtotal means the sum of the results of the first 8 examples.**

respectively for each example. Columns LUT and FF list the numbers of LUTs and flipflops, respectively, in the final solutions. Columns Φ list the minimum clock periods of the final solutions. Columns CPU list the cpu time on a SPARC10, which includes the time for label computation and mapping generation, but excludes the postprocessings. Note that both TurboMap and SeqMapII-opt can get mapping solutions with the optimal clock periods under retiming, but TurboMap is 2×10^4 times faster. Moreover, TurboMap is more than 2 times faster than SeqMapII-heu which may generate the suboptimal solutions[5]. Compared with SeqMapII-heu, TurboMap produces mapping solutions with 7% smaller of the clock periods, uses 8% fewer FFs and about the same LUTs.

To show the effect of our K-cut computation on partial flow networks, we compare the

[5]Since SeqMapII forks a child process to perform the max-flow computation, the CPU times listed under SeqMapII include this portion as well.

circuit	NODE	FF	TurboMap		SeqMapII-opt		SeqMapII-heu	
			$\mathcal{E}_{max}$	$\mathcal{E}_{avg}$	$\mathcal{E}^{Kn}_{max}$	$\mathcal{E}^{Kn}_{avg}$	$\mathcal{E}^{6}_{max}$	$\mathcal{E}^{6}_{avg}$
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
sse	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
total	19419	2359	6673	1935	—	—	2460	1348
ratio			1	1	—	—	0.4	0.7
subtotal			2907	1341	1153373	1103982	2024	1147
ratio			1	1	396.8	823.3	0.7	0.9

Table 3. Comparison of the sizes of expanded circuits used by TurboMap and SeqMapII. Entry "*" means no results after running the algorithm for 24 hours. Subtotal means the sum of the results excludes the last 4 examples.**

sizes (in terms of numbers of nodes) of the partial flow networks with $\mathcal{E}_v^{Kn}$ and $\mathcal{E}_v^6$ which are used in SeqMapII [PaLi96b]. The results are shown Table 3. The column NODE lists the number of nodes in the retiming graph for each example, which is the sum of the numbers of the PIs, the POs and the gates. The column FF lists the numbers of flipflops 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, we cannot generate $\mathcal{E}_v^{Kn}$ 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 823 times smaller than $\mathcal{E}_v^{Kn}$. This result, together with our efficient label update and SCC partition, gives an explanation why TurboMap is significantly faster than SeqMapII-opt and even 2 times faster

than SeqMapII-heu.

6. Conclusions and Future Works

We presented an improved algorithm, named Turbomap, for technology mapping with retiming for optimal clock periods. We showed 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, K-cut computation on the partial flow network of very small size in practice as well as SCC partition. The experimental results showed that TurboMap is about 2×10^4 times faster than SeqMapII for the optimal solutions. In our future work, we want to develop an efficient algorithm to compute the initial state of the mapping solution and deal with area minimization problem. Also, we are in the process of integrating the TurboMap algorithm into RASP [CoPD96].

7. Acknowledgment

The authors thank very much Professors Peichan Pan and C. L. Liu for providing the SeqMapII program for the comparative study. The authors also appreciate the helpful discussion with Mr. Yeanyow Hwang and Mr. John Peck. This work is partially supported by Beijing Intergrated Circuit Design Center, National Science Foundation Young Investigator Award MIP9357582 and grants from Xilinx and AT&T Microelectronics under the California MICRO program.

References

[Al95] Altera. *Flex 8000 and Flex 10000 Programmable Logic Device Family Data Sheets*. 1995.

[At95] AT&T Microelectronics. *AT&T Field-Programmable Gate Arrays Data Book*. 1995.

[BrRS87] R. K. Brayton, R. Rudell, A. L. Sangiovanni-Vincentelli, and A. R. Wang. Mis: A multiple-level logic optimization system. *IEEE Tans. on Computer-Aided Desing*, 6(6):1062–1081, 1987.

[ChCD92] K. C. Chen, J. Cong, Y. Ding, A. B. Kahng, and P. Trajmar. DAG-Map: Graph-based FPGA Technology Mapping for Delay Optimization. In *IEEE Design and Test of Computers*, 1992.

[CoDi93a] J. Cong and Y. Ding. On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping. In *30th ACM/IEEE Design Automation Conference*, pages 213–218, 1993.

[CoDi94] J. Cong and Y. Ding. FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. *IEEE Trans. on Computer-Aided Design of Integrated Circuits And Systems*, 13(1):1–12, 1994.

[CoDi96] J. Cong and Y. Ding. Combinational Logic Synthesis for SRAM Based Field Programmable Gate Arrays. *ACM Transactions on Design Automation of Electronic Systems*, 1(2), 1996. http://www.acm.org/todaes/V1N2/TOC.html.

[CoHw95] J. Cong and Y.-Y. Hwang. Simultaneous Depth and Area Minimization in LUT-Based FPGA Mapping. In *ACM 3rd Int'l Symp. on Field Programmable Gate Arrays*, pages 68–74, 1995.

[CoHw96] J. Cong and Y.-Y. Hwang. Structural Gate Decomposition for Depth-Optimal Technology Mapping in LUT-based FPGA Design. In *33rd ACM/IEEE Design Automation Conference*, pages 726–729, 1996.

[CoLR90] T. H. Cormen, C. H. Leiserson, and R. L. Rivest. *Introduction To Algorithms*. The MIT Press, 1990.

[CoPD96] J. Cong, J. Peck, and Y. Ding. RASP: A General Logic Synthesis System for SRAM-based FPGAs. In *Proc. ACM 4th Int'l Symp. on FPGA*, pages 137–143, 1996.

[FrRC90] R. J. Francis, J. Rose, and K. Chung. Chortle: A Technology Mapping Program For Lookup Table-Based Field Programmable Gate Arrays. In *27th ACM/IEEE Design Automation Conference*, pages 613–619, 1990.

[FrRV91a] R. J. Francis, J. Rose, and Z. Vranesic. Chortle-crf: Fast Technology Mapping For Lookup Table-Based FPGAs. In *28th ACM/IEEE Design Automation Conference*, pages 613–619, 1991.

[FrRV91b] R. J. Francis, J. Rose, and Z. Vranesic. Technology Mapping Of Lookup Table-Based FPGAs. In *IEEE International Conference on CAD*, pages 568–571, 1991.

[LeRS83] C. E. Leiserson, F. M. Rose, and J. B. Saxe. Optimizing Synchronous Circuitry by Retiming (Preliminary Version). In R. Bryant, editor, *Proceedings of the 3rd Caltech Conference on Very Large Scale Integration*, pages 87–116, Rockville, Maryland, 1983. Caltech, Computer Science Press.

[LeSa81] C. E. Leiserson and J. B. Saxe. Optimizing Synchronous Systems. In *22nd IEEE Annual Symposium on Foundations of Computer Science*, pages 23–36, 1981.

[LeSa91] C. E. Leiserson and J. B. Saxe. Retiming Synchronous Circuitry. *Algorithmica*, 6:5–35, 1991.

[MuBS93] R. Murgai, R. K. Brayton, and A. Sangiovanni-Vincentelli. Sequential Synthesis For Table Lookup Programmable Gate Arrays. In *30th ACM/IEEE Design Automation Conference*, pages 224–229, 1993.

[MuSB91a] 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*, pages 564–567, 1991.

[MuSB91b] R. Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli. Performance Directed Synthesis For Table Lookup Programmable Gate Arrays. In *IEEE International Conference on CAD*, pages 572–575, 1991.

[PaLi96a] P. Pan and C. L. Liu. Technology Mapping of Sequential Circuits for LUT-based FPGAs for Performance. In *ACM/SIGDA International Symposium on FPGAs*, pages 58–64, 1996.

[PaLi96b] P. Pan and C. L. Liu. Optimal Clock Period FPGA Technology Mapping for Sequential Circuits. In *33th ACM/IEEE Design Automation Conference*, pages 720–725, 1996.

[PaLi96c] P. Pan and C. L. Liu. private communication. 1996.

[ScKC94] M. Schlag, J. Kong, and P. K. Chan. Routability-Driven Technology Mapping for Lookup Table-Based FPGA's. *IEEE Trans. on Computer-Aided Design of Integrated Circuits And Systems*, 13(1):13–26, 1994.

[SeSL92] 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, Memorandum No. UCB/ERL M92/41, 1992.

[ShRu94] N. Shenoy and R. Rudell. Efficient Implementation of Retiming. In *31th ACM/IEEE Design Automation Conference*, pages 226–233, 1994.

[ToBr93] H. Touati and R. K. Brayton. Computing the Initial States of Retimed Circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits And Systems*, 12(1):157–162, 1993.

[ToSO94] N. Togawa, M. Sato, and T. Ohtsuki. Maple: A Simultaneous Technology Mapping, Placement, and Global Routing Algorithm for Field-Programmable Gate Arrays. In *31th ACM/IEEE Design Automation Conference*, pages 156–163, 1994.

[WeRo93] U. Weinmann and W. Rosenstiel. Technology Mapping For Sequential Circuits Based On Retiming Techniques. In *Proceedings of European Design Automation Conference*, pages 318–323, 1993.

[Xi94] Xilinx. *The Programmable Logic Data Book*. 1994.