FPGA Synthesis with Retiming and Pipelining for Clock Period Minimization of Sequential Circuits

Jason Cong and Chang Wu
Department of Computer Science
University of California, Los Angeles, CA 90095

Abstract

In this paper, we present a new algorithm, named TurboSYN, for FPGA synthesis with retiming and pipelining to minimize the clock period for sequential circuits. For a target clock period, since pipelining can eliminate all critical I/O paths, but not critical loops, we concentrate on FPGA synthesis to eliminate the critical loops. We combine the combinational functional decomposition technique with retiming to perform the sequential functional decomposition, and incorporate it in the label computation of TurboMap [11] to eliminate all critical loops. The results show a significant improvement over the state-of-the-art FPGA mapping and resynthesis algorithms ($1.7 \sim 2$ times reduction on the clock period). Moreover, we develop a novel approach for positive loop detection which leads to over $10 \sim 50$ times speedup of the algorithm. As a result, TurboSYN can optimize sequential circuits of over 10^4 gates and 10^3 flipflops in reasonable time.

1 Introduction

The FPGA synthesis and technology mapping is to produce a functionally equivalent circuit for a given circuit using specified programmable logic blocks (PLBs). In this paper, we consider lookup table (LUT) based FPGAs. A lookup table with K inputs and one output, denoted K-LUT, can implement any combinational function of up to K inputs. It is the most widely used PLB in current FPGAs [1, 18, 27]. There have been many research results on K-LUT mapping and synthesis for combinational circuits. A comprehensive survey can be found in [7]. In particular, FlowMap [6] produces depth-optimal mapping solutions without resynthesis while FlowSYN [5] and BoolMap-D [15] can produce mapping solutions with even smaller depth using resynthesis techniques by exploiting Boolean optimization. For sequential circuits, however, these approaches assume that the positions of flipflops (FFs) are fixed so that the entire circuit can be partitioned into a number of combinational subcircuits, and each combinational subcircuit is mapped independently. However, delay-optimal mapping solutions to all combinational subcircuits may not lead to a cycle-optimal mapping solution for the sequential circuit, since they do not consider retiming which allows FFs to be moved without changing the circuit behavior [16],

Several heuristics have been proposed for technology mapping of sequential circuits with consideration of retiming [20, 24, 25]. Recently, a significant advancement was made by Pan and Liu [19]. They proposed a novel algorithm, named SeqMapII, to find mapping solutions with the minimum clock periods under retiming for sequential circuits. Although their algorithm runs in polynomial time, the time complexity is too high to be used directly in practical designs. The recent work of TurboMap by Cong and Wu [11] leads to a significant speedup of the SeqMapII algorithm. It reduces the runtime of SeqMapII by 2×10^4 times on average and can optimize circuits of 10^4 gates efficiently with optimal clock periods. Although the results of these algorithms are better than those of FlowMap [6] based approach, they are not as good as those of FlowSYN [5] based approach in general, because FlowSYN exploits Boolean optimization. It is natural to expect better results by considering resynthesis, retiming and pipelining at the same time for sequential circuits.

For general logic synthesis with retiming, Malik et. al. [17] proposed an approach of peripheral retiming with resynthesis and, Dey et. al. [3] proposed a logic resynthesis approach based on global path delays for retiming. However, peripheral retiming applies to only pipelined circuits which are a special class of sequential circuits. For the approach in [3], it is difficult to choose global path delays without exact information of whether it can be realized with resynthesis. Note that both approaches depend somewhat on the circuit structure or the initial positions of the FFs.

In this paper, we present a new algorithm for FPGA synthesis with retiming and pipelining to minimize the clock periods for sequential circuits. For a path in an LUT circuit, the path's *delay-to-register (DR) ratio* is the ratio of the number of LUTs on it over the number of FFs on it. An I/O path is a path from a primary input (PI) to a primary output (PO). A loop is a path starting and ending at the same LUT. The clock period of an LUT circuit under retiming is bounded by both the maximum loops' DR ratio and the maximum I/O paths' DR ratio. Pipeline is a kind of FF insertion to reduce the clock period without changing the circuit input-output behavior. It can be done by inserting the same number of FFs on the fanout edges of every PI and retiming for the minimum clock period. If both retiming and pipelining are allowed, the clock period, however, is only limited by the maximum DR ratio of all loops [22], since the maximum DR ratio of I/O paths can be reduced by pipelining. This implies that it is more important to generate a mapping solution for loop's DR ratio minimization rather than I/O path's DR ratio minimization. Furthermore, the synthesis algorithm can have much more freedom to reduce the maximum loops' DR ratio without restriction on the maximum I/O paths' DR ratio. Therefore, we concentrate on minimizing the maximum DR ratio of loops instead of the clock period. Since most of the existing commercial FPGAs are register-rich, retiming and pipelining are particularly attractive in FPGA designs.

Our algorithm, names TurboSYN, is the first algorithm which combines the combinational synthesis technique for FPGAs with technology mapping, retiming, and pipelining in one phase for performance optimization. It applies to general synchronous sequential circuits and does not depend on the initial positions of the FFs. The results show that TurboSYN presents substantial improvement over the state-of-the-art FPGA synthesis algorithms. We believe this new approach is applicable to the general logic synthesis with retiming as well by applying different resynthesis techniques.

The rest of the paper is organized as follows. Section 2 is the problem formulation and overview of the TurboSYN algorithm. Section 3 is the label computation for MDR ratio minimization, which is the most important part of the algorithm. In Section 4, we propose a novel speedup technique followed by a summary of the algorithm. Sections 5 and 6 are the experimental results on a set of MCNC and ISCAS benchmarks and conclusions.

2 Problem Formulation and Overview of the TurboSYN Algorithm

In this paper, sequential circuits are represented by retiming graphs. The retiming graph $G(V, E, W)$ of a sequential circuit is a directed graph, where V is the set of nodes, E is the set of edges and W is the set of weights of edges [16]. Each node in V represents a gate or a PI/PO in the original circuit. Each edge $e(u, v)$ in E represents a directed connection from node u to node v. The *weight* $w(e)$ of an edge e is the number of FFs on the connection represented by the edge. The *path weight* $w(p)$ of a path p is the sum of the weights of all edges on the path. A *loop* is a path that starts from and ends at the same node. In an LUT network and under the unit-delay mode, the *delay* $d(u)$ is 1 for every internal node u, and 0 for every PI or PO. The *path delay* $d(p)$ of a path p is the sum of the delays of all nodes on the path. The *delay-to-register ratio* of a loop l, denoted $DR(l)$, is $\lceil \frac{d(l)}{w(l)} \rceil$. The *maximum delay-to-register ratio* of all loops is called the *MDR ratio* of the circuit. For a target clock period ϕ, a loop l is *critical*, if $DR(l) > \phi$. Pipelining is one way to insert FFs followed by retiming to reduce the clock period. With both retiming and pipelining, the clock period of a circuit is bounded only by the MDR ratio of the circuit, based on the theories in [16, 22]. Therefore, to find a mapping solution with the minimum clock period under retiming and pipelining, we propose to solve the following problem:

Problem 1 *For a sequential circuit, find a functionally equivalent LUT circuit with the minimum MDR ratio.*

The decision formulation is:

Problem 2 *For a sequential circuit and target clock period ϕ, decide if there exists a functionally equivalent LUT circuit with the MDR ratio of no more than ϕ.*

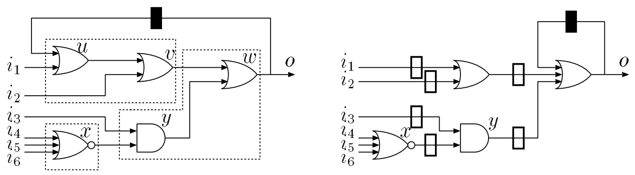
As in [6, 11, 19], this paper assumes that the initial circuits are K-bounded. (When a circuit is not K-bounded, we can use gate decomposition algorithms, such as balanced tree decomposition [2], DMIG [4] or DOGMA [9], to decompose the gates with more than K fanins.) To solve Problem 1, our algorithm, named TurboSYN, works in three steps:

(1) label computation (to be explained later) with sequential functional decomposition to search for a mapping solution with the minimum MDR ratio,

(2) mapping generation and area reduction,

(3) pipelining and retiming to get the final solution.

In this paper, we concentrate on the first step of label computation for solving Problem 2 which is the most critical step among the three steps, since the label computation determines the minimum possible MDR ratio after Steps 2 and 3. We combine the label computation of TurboMap [11] with OBDD based functional decomposition, which has been shown to be very effective for the FPGA mapping problem [5, 14], to search for a mapping solution with the minimum MDR ratio. To further speed up the algorithm, we propose an efficient positive loop detection algorithm which can speed up the label computation by over $10 \sim 50$ times. For Steps 2 and 3, we propose a label relaxation method and low-cost K-cut computation for area minimization and use the methods similar to those in [6, 11, 16, 19].



(a) circuit with one positive loop (b) synthesis to eliminate positive loop

▮ is a flipflop in the original circuit, ▯ is a flipflop inserted by pipelining

Figure 1: Resynthesis to eliminate critical loops for 3-LUT and the target MDR ratio 1.

To illustrate our approach, let us consider the example shown in Figure 1(a) for a target MDR ratio of 1. The minimum delay of path $u \rightarrow v \rightarrow w$ is at least 2 even with resynthesis since the total number of inputs to nodes u, v and w is 4. Thus, the DR(l) of loop $l(u \rightarrow v \rightarrow w \rightarrow u)$ is at least 2, which is larger than the target MDR ratio of 1. Now we replace the edge (w, u) with (w, w) through resynthesis as shown in Figure 1(b). As a result, the DR(l) is reduced to 1 and, so is the clock period after two level of pipeline insertion. Note that it is hard, if not impossible, for the existing FPGA mapping and resynthesis algorithms to get a mapping solution with a clock period of 1 for this example.

3 Label Computation with Resynthesis for MDR Ratio Minimization

The deterministic Problem 2 is to decide whether there exists a mapping solution with the MDR ratio of no more than the target ϕ. With binary search one can compute the minimum MDR ratio of all possible mapping solutions to solve Problem 1. In this section, we present a novel iterative label computation method to meet a given MDR ratio ϕ. First, we review the label computation of TurboMap [11] and explain how it can be expanded for solving Problem 2. Then, we introduce a sequential functional decomposition technique and combine it with the label computation of TurboMap [11] to form the label computation of our TurboSYN algorithm.

3.1 Definitions

For a mapping solution with a target clock period ϕ, the *length* of an edge $e(u,v)$, denoted $length(e)$, is defined to be $-\phi \cdot w(e) + d(v)$, where $w(e)$ is the edge weight and $d(v)$ is the delay of node v. The *path length* of a path p, denoted $length(p)$, is the sum of lengths of all edges on the path. Obviously, the *delay-to-register ratio* DR(l) of a loop l is larger than ϕ, if and only if $length(l) > 0$. A loop with positive length is called a *positive loop*.

Given a mapping solution M of a circuit, the *l-value* of each node v, denoted $l_M(v)$, is the maximum length of all the paths from PIs to v in M. If there is one positive loop in M, $l_M(v) = +\infty$ for every node v on the loop. Clearly, we have the following result:

Theorem 1 *For a mapping solution M, the MDR ratio is ϕ or less if and only if the l-value $l_M(v)$ is finite for every node v.*

The *label* of each node v, denoted $l^{opt}(v)$, is the minimum $l_M(v)$ of v among all possible mapping solutions M. Based on Theorem 1, Problem 2 can be solved by computing all node labels and checking whether all of them are finite.

3.2 Label Computation and Review of TurboMap

Pan and Liu [19, 21] proposed an iterative labeling process in SeqMapII to compute the node labels in $O(K^3n^5)$ time and $O(K^2n^2)$ space for a circuit with n gates[1]. A recent improvement was proposed by Cong and Wu [11], which leads to significant reduction in runtime (over 10^4 times) and space requirement (over 800 times). Our label computation algorithm is based on that of [11].

One important concept is the expanded circuit for a node v, denoted $\mathcal{E}_v$, proposed by Pan and Liu [19] to represent all possible LUTs rooted at v under retiming and node replication. The expanded circuit $\mathcal{E}_v$ is a directed acyclic graph rooted at v. It has the property that all paths from any given node $u^w \in \mathcal{E}_v$ to the root v^0 pass the same number (w) of FFs, where u^w is a replication of node u in the original circuit. Node u^w is a leaf if it has no fanins in $\mathcal{E}_v$. A *cut* $(X,\overline{X})$ is a partition of $\mathcal{E}_v$, such that all leaves are in X and the root is in $\overline{X}$. The *node cut-set* $V(X,\overline{X})$ is the set of nodes in X which have connections to nodes in $\overline{X}$. A cut is called a *K-cut* if $| V(X,\overline{X}) | \leq K$.

To compute the node labels, the TurboMap algorithm [11] assigns a set of lower-bounds of node labels and iteratively update them until all of them converge to the node labels. The initial lower-bounds for internal nodes and POs are $-\infty$. The labels of PIs is 0. To update the lower-bound for node v, we examine the lower-bounds $l(u)$ of all fanin nodes u of v and compute $\mathcal{L}(v) = max\{l(u) - \phi \cdot w(e) \mid \forall e(u,v) \in G\}$. The new lower-bound $l_{new}(v)$ for node v is computed as follows [11]:

$$l_{new}(v) = \begin{cases} \mathcal{L}(v) & \text{if } \exists K\text{-cut with } h(X,\overline{X}) \leq \mathcal{L}(v) \\ \mathcal{L}(v)+1 & \text{otherwise,} \end{cases}$$

[1] Originally, the label computation in SeqMapII was stated as $O(K^3n^4\log(Kn^2))$. Later on, however, the authors revised the complexity to $O(K^3n^5)$ due to the difficulty to prove the convergency of label computation in $O(n)$ iterations [21]. Instead, they proved a bound of $O(n^2)$. The $O(\log(Kn^2))$ speedup is due to the result in [11].

where the *height* $h(X,\overline{X})$ of a K-cut $(X,\overline{X})$ on $\mathcal{E}_v$ is defined to be $max\{l(u) - \phi \cdot w + 1 \mid \forall u^w \in V(X,\overline{X})\}$. To decide whether $l_{new}(v) = \mathcal{L}(v)$, the max-flow computation is performed on a partial flow network to check if there exists a K-cut on the expanded circuit $\mathcal{E}_v$ with height $\leq \mathcal{L}(v)$ [11]. If such a K-cut exists, $l_{new}(v)$ is set to be $\mathcal{L}(v)$. Otherwise, $l_{new}(v)$ is set to be $\mathcal{L}(v) + 1$. The procedure of updating the lower-bound of every node label once is called *one iteration* of label computation. The iterative label computation is repeated until there is no more improvement of the lower-bounds (in this case, the current lower-bounds are the node labels), or stopped after performing n^2 iterations (in this case, there is no solution for the target MDR ratio) [19, 11].

It was showed that the above label computation guarantees to find the optimal solutions if no resynthesis is allowed [11]. With resynthesis, however, the node label $l^{opt}(v)$ can be further reduced as a much larger solution space is explored. In the following subsection, we show how to achieve $l_{new}(v) = \mathcal{L}(v)$ with sequential functional decomposition even if a K-cut with the height of $\mathcal{L}(v)$ cannot be found on the expanded circuits.

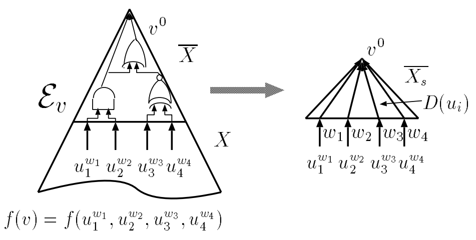

v^0
$\overline{X}$
$\mathcal{E}_v$
X
$u_1^{w_1}\ u_2^{w_2}\ u_3^{w_3} u_4^{w_4}$
$f(v) = f(u_1^{w_1}, u_2^{w_2}, u_3^{w_3}, u_4^{w_4})$
v^0
$\overline{X_s}$
$D(u_i)$
$w_1\ w_2\ w_3\ w_4$
$u_1^{w_1}\ u_2^{w_2}\ u_3^{w_3} u_4^{w_4}$

Figure 2: Sequential logic function for a cut on the expanded circuit. $D(u_i)$ is the maximum delay from $u_i^{w_i}$ to v^0 in the resynthesized cone $\overline{X_s}$.

3.3 Sequential Functional Decomposition

As stated in the previous subsection, if there is no K-cut of height $\leq \mathcal{L}(v)$ for a node v, $l_{new}(v)$ will be set to be $(\mathcal{L}(v) + 1)$ if no resynthesis is considered. In this work, however, we introduce an OBDD based functional decomposition for sequential circuits, and use it to try to maintain $l_{new}(v) = \mathcal{L}(v)$. Based on the property of expanded circuits $\mathcal{E}_v$ that every path to the root v^0 from a given node u^w passes w FFs [19], every cut $(X,\overline{X})$ on $\mathcal{E}_v$ corresponds to a sequential function $f(V(X,\overline{X})) = f(u_1^{w_1}, u_2^{w_2}, .., u_m^{w_m})$, as shown in Figure 2, where $u_i^{w_i}$ are the nodes in the node cut-set $V(X,\overline{X})$. Clearly, we can push all the FFs within the cone $\overline{X}$ to the fanout edges of nodes in the cut-size with retiming to make $\overline{X}$ purely combinational. Therefore, all the combinational logic synthesis technique can be used on such sequential functions $f(V(X,\overline{X}))$ of cones $\overline{X}$ on $\mathcal{E}_v$ directly.

Let $D_{\overline{X_s}}(u_i)$ denote the maximum delay among all paths from $u_i^{w_i}$ to the root v^0 in the resynthesized cone $\overline{X_s}$. Then, the new lower-bound of the node label, denoted $l_s(v)$, is $\max\{l(u_i) - \phi \cdot w_i + D_{\overline{X_s}}(u_i) \mid \forall\ u_i^{w_i} \in V(X,\overline{X})\}$. Thus, $l_s(v) \leq \mathcal{L}(v)$ implies that $D_{\overline{X_s}}(u_i) \leq \mathcal{L}(v) - l(u_i) + \phi \cdot w_i$ for every node $u_i^{w_i} \in V(X,\overline{X})$. In order to satisfy those conditions, we resynthesize cone $\overline{X}$ (or equivalently, function $f(V(X,\overline{X}))$) using the OBDD based functional decomposition technique, since it shows to be very effective for FPGA mapping problem [5, 14]. Similar to the FlowSYN

```
LabelUpdateSYN($v, \phi$)
1  $\mathcal{L}(v) \leftarrow \max\{l(u) - \phi \cdot w(e) \mid \forall e(u,v)\}$
2  decide the existence of a $K$-cut with $h(X,\overline{X}) \leq \mathcal{L}(v)$
3  if exists such a $K$-cut in $\mathcal{E}_v$, return $\mathcal{L}(v)$
4  else for $h$ from 0 and increased one by one
5      compute a min-cut in $\mathcal{E}_v$ of height $\leq \mathcal{L}(v) - h$
6      if cut-size $>$ Cmax, return $\mathcal{L}(v) + 1$
7      else decompose the corresponding function $f(v)$
8      if $(l_s(v) \leq \mathcal{L}(v))$, return $\mathcal{L}(v)$
```

Figure 3: Label update with resynthesis. Cmax is a given constant to bound the cut-sizes, which is set to be 15 in TurboSYN.

algorithm [5], we sort the inputs ${u_i}^{w_i}$ of $f(V(X,\overline{X}))$ in increasing order of $(l(u_i) - \phi \cdot w_i)$. Then, we choose the first K inputs as the bound set B and the rest as the free set F to decompose the sequential function $f(V(X,\overline{X}))$ into $f(V(X,\overline{X})) = f(B,F) = g(\vec{\alpha}(B),F)$, where $\vec{\alpha}(B)$ is a set of encoding functions depending on B and, $\mid \vec{\alpha}(B) \mid < K$. We iteratively decompose $g(\vec{\alpha}(B),F)$ until it is a K-bounded function. The decompositions forms a resynthesized cone $\overline{X_s}$. If the constraint $D_{\overline{X_s}}(u_i) \leq \mathcal{L}(v) - l(u_i) + \phi \cdot w_i$ is satisfied for every input $u_i^{w_i}$, $l_{new}(v)$ is set to be $\mathcal{L}(v)$. Otherwise, we perform the above resynthesis on another partition $(X', \overline{X'})$ with larger $\overline{X'}$, or set $l_{new}(v) = \mathcal{L}(v) + 1$ when we reach the stopping condition shown in Figure 3 line 6.

Based on the above discussions, the label update algorithm in TurboSYN works in three steps. First, we compute $\mathcal{L}(v)$ based on the lower-bounds of node labels. Then, we decide the existence of a K-cut in $\mathcal{E}_v$ with the height $\leq \mathcal{L}(v)$ using the flow-based K-cut computation of [11]. If such a K-cut is found, $l_{new}(v)$ is set to be $\mathcal{L}(v)$. Otherwise, we compute a sequence of min-cuts $(X_h, \overline{X_h})$ with the height of $(\mathcal{L}(v) - h)$ for h starting 0 and increasing one by one. For each sequential function $f(V(X_h,\overline{X_h})) = f(u_1^{w_1}, u_2^{w_2}, .., u_m^{w_m})$, we try to decompose it into a set of K-bounded subfunctions such that $l_s(v) \leq \mathcal{L}(v)$. If such a decomposition can be found, $l_{new}(v)$ is set to be $\mathcal{L}(v)$. Otherwise, $l_{new}(v)$ is set to be $\mathcal{L}(v) + 1$ at the end of the resynthesis sequence. The pseudo code is shown in Figure 3. With such resynthesis, we observed that in many cases $l_{new}(v)$ can assume value $\mathcal{L}(v)$ instead of $\mathcal{L}(v) + 1$. As a result, we may be able to keep each $l^{opt}(v)$ to be a finite number for a given MDR ratio, which means all the critical loops can be eliminated.

4 Algorithm Speedup and Summary

Although the label computation stated in the previous section can get mapping solutions with smaller MDR ratios, the runtime, however, can be very long due to the complexity of the functional decomposition and the iterative nature of the label computation. Since the computation time is proportional to the number of label computing iterations, in this section we present a novel technique to reduce the iteration number to speed up the algorithm.

To get the minimum MDR ratio, there will be two cases for label computation:

1) the case that there exists a mapping solution for a target MDR ratio and,

2) the case that there does not exist any mapping solution for a target MDR ratio.

Our experimental results show that, though the iteration number is far less than n in the first case, it goes to n^2 in the second case due to the lack of a better stopping criterion other than the very conservative upper-bound of n^2 in [21]. Clearly the reduction of the iteration number for the second case is critical to the speedup of the algorithm.

To reduce the iteration number for the second case, we propose a novel positive loop detection technique, called *PLD*. Our PLD technique is based on the idea of predecessor graphs which are similar to those used in the shortest path theory [13]. The set of predecessors $\pi[v]$ of node v is defined to be $\pi[v] = \{u \mid \forall e(u,v) \in G, l(u) - \phi \cdot w(e) + 1 \geq l(v)\}$ if $l(v) > \infty$, or otherwise, $\pi[v] = \emptyset$, for the current lower-bounds $l(u)$ of labels. The predecessor graph is defined to be $G_\pi(V_\pi, E_\pi)$, where $V_\pi = \{v \mid v \in V \ and \ \pi[v] \neq \emptyset\}$ and $E_\pi = \{e(u,v) \mid e(u,v) \in E \ and \ u \in \pi[v]\}$. A node v is *isolated* from node u in G_π if there does not exist any path from u to v in G_π.

Based on the topological order of SCCs (strongly connected components) for the label computation proposed in [11], we have the following important theorem:

Theorem 2 *For a sequential circuit $G(V,E,W)$ and one SCC with n nodes, there exists a positive loop in this SCC in all possible mapping solutions, if and only if after $6n$ iterations of label computation, all nodes of the SCC is isolated from the PIs in the predecessor graph G_π.*

Due to the page limitation, the proof is left out in [12]. According to this result, our PLD procedure for each SCC works as follows after every labeling iteration: 1) constructing G_π, 2) checking whether this SCC is totally isolated from the PIs in G_π. This procedure guarantees to determine exactly the existence of positive loops (if there is any) within $6n$ iterations for a SCC of n nodes. It is a significant improvement over the upper-bound of n^2 iterations proposed in [21]. We incorporated this approach into both the TurboMap [11] and our TurboSYN algorithms to compute the minimum MDR ratio and obtained 10$\sim$50 times speedup in runtime.

Since the functional decomposition is very time consuming, we use a heuristic to analyze the resynthesizability of nodes such that if resynthesis fails to maintain $l_{new}(v)$ to be $\mathcal{L}(v)$ for nodes v a consecutive times, we mark those nodes as non-resynthesizable and prohibit further resynthesis for them. As a result, our algorithm can compute mapping solutions with the minimal MDR ratio for circuits with over 10^4 gates and 10^3 FFs in reasonable time.

A number of LUT reduction techniques are considered in TurboSYN [12]. First, we try to reduce the number of nodes which need resynthesis by label relaxation, i.e., not using the resynthesized results of some nodes and increasing their labels if no positive loops will occur. Second, efficient low-cost K-cut computation (similar to the min-cost K-cut computation in [8], but much more efficient) is performed to maximize the sharings of inputs of different LUTs, thus, to reduce the number of LUTs. At last, mpack [4] and flowpack [6] is performed to further reduce the number of LUTs. The flipflop minimization is left for retiming [16].

Our algorithm, named TurboSYN, can be summarized as follows. First, it runs TurboMap [11] to get an upper-bound *UB* of the minimum MDR ratio. Then, TurboSYN uses binary search over all values in the range of 1 to *UB* to get the minimum MDR ratio. For each target MDR ratio ϕ, the iterative label computation with resynthesis and PLD procedure are performed to test if there exists a mapping solution with the MDR ratio $\leq \phi$. After the minimum MDR ratio ϕ_{min} is computed, we perform the aforementioned area

			TurboSYN				FlowSYN-s				TurboMap			
CIRCUIT	GATE	FF	LUT	FF	Φ	CPU	LUT	FF	Φ	CPU	LUT	FF	Φ	CPU
bbara	28	10	13	23	1	0.9	19	10	4	0.1	13	7	3	0.1
dk16	162	5	49	6	3	156.7	44	5	3	0.4	103	14	14	3.4
dk17	42	5	6	3	1	0.1	10	5	2	0.1	6	3	1	0.1
kirkman	106	5	59	45	2	4.6	51	36	2	0.1	56	36	3	0.3
ex1	140	5	133	41	4	27.2	110	5	5	0.4	93	44	7	0.8
s1	107	5	199	95	4	39.1	112	5	7	2.0	63	9	7	0.5
sse	74	4	88	36	3	8.0	65	4	5	0.2	50	13	6	0.2
keyb	134	5	198	72	4	28.2	80	5	7	0.9	79	14	9	0.9
styr	281	5	386	132	5	153.4	198	5	10	3.8	171	8	16	4.1
sand	327	17	441	87	8	236.0	261	17	15	12.3	178	33	15	4.8
planet1	348	6	224	78	6	166.4	312	6	14	1.8	222	26	18	9.5
scf	516	7	368	31	5	172.7	373	11	8	5.1	344	27	13	21.2
s9234.1	1299	135	677	404	3	286.2	550	141	5	10.6	500	217	4	54.4
s5378	1503	164	645	683	2	220.7	527	291	4	4.8	505	424	3	32.3
s15850.1	3801	515	2487	1546	4	3728.9	1526	557	7	55.1	1393	952	6	2679.4
s38417	9817	1464	6833	2870	5	2886.7	5350	1464	7	3905.0	3648	2420	6	613.4
geo-mean			206	84	3.3	56.6	168	20	5.6	2.2	143	39	6.4	4.2
ratio			1	1	1	1	0.81	0.24	1.72	0.04	0.69	0.46	1.96	0.07

Table 1: Comparison of TurboSYN with FlowSYN-s and TurboMap for 5-LUT. "geo-mean" lists the geometric mean of the results by each approach. The runtime were recorded on a SUN ULTRA 1 with 192MB memory.

```
TurboSYN($G(V, E, W)$)
1   upper bound $UB$ of the MDR ratio←TurboMap [11]
2   binary search of $\phi$ from 1 to $UB$
3     $l(PI) \leftarrow 0$ and $l(u) \leftarrow -\infty$
4     **while** (no positive loops by PLD checking)
5       converge←1
6       **for** each node $v \in V$
7         $l_{new}(v)$ = LabelUpdateSYN($v, \phi$)
8         **if** ($l_{new}(v) > l(v)$), converge← 0, $l(v) \leftarrow l_{new}(v)$
9       **if** (converge), **break** with SUCCESS
10  mapping generation for $\phi_{min}$ and LUT reduction
11  $l(G) \leftarrow \max\{l(PO) \mid for\ \phi_{min}\}$
12  pipeline of $\lceil \frac{l(G)-\phi_{min}}{\phi_{min}} \rceil$ levels if $l(G) > \phi_{min}$
13  retiming for $\phi_{min}$ and FFs minimization
```

Figure 4: Pseudo code of the TurboSYN algorithm.

reduction process and generate the mapping solution M. Finally, pipelining is performed by adding $\lceil \frac{l(G)-\phi_{min}}{\phi_{min}} \rceil$ FFs on every fanout edge of each PI if $l(G) = \max\{l(PO)\} > \phi_{min}$ and followed by retiming. The pseudo code of the TurboSYN algorithm is shown in Figure 4.

5 Experimental Results

The TurboSYN algorithm has been implemented in C language on Sun SPARC workstations and incorporated into the SIS package [23] and the RASP System [10]. The test set consists of 12 MCNC FSM benchmarks and 4 ISCAS'89 benchmarks. SIS sequential synthesis commands and dmig [4] are performed to generate the initial circuits which are shown in the first three columns in Table 1. Columns GATE and FF list the number of gates and FFs in each circuit, respectively. Our experiment was performed on a SUN ULTRA 1 workstation with 192MB memory. K is set to be 5.

In Table 1, we compared TurboSYN with TurboMap [11] and FlowSYN-s. FlowSYN-s is based on the FlowSYN algorithm [5]. It first partitions the sequential circuits into a set of combinational subcircuits by cutting at all FFs, then maps every subcircuits independently with the FlowSYN algorithm, and finally, merges the mapped LUT circuits with the original FFs. The PLD technique has been used in both TurboSYN and TurboMap. Pipelining and retiming have been performed as postprocessing steps for all the three approaches. In Table 1, Columns LUT and FF list the numbers of LUTs and FFs after retiming and pipelining, respectively, in the final mapping solutions. Columns Φ list the minimum clock periods (or MDR ratios) under retiming and pipelining. Columns CPU list the CPU time for all the three algorithms. Note that we did not list the postprocessing time of mpack [4], flowpack [6], pipelining and retiming for the three approaches, because it is usually small compared to the label computation time, and the postprocessings are not our contribution. The results show that TurboSYN can reduce the clock period by 1.72 and 1.96 times as compared with FlowSYN-s and TurboMap, respectively. But TurboSYN uses 19% more LUTs and 76% more FFs as compared with FlowSYN-s, and 31% more LUTs and 54% more FFs as compared with TurboMap. In most of the cases, the CPU time of TurboSYN is less than 5 minutes, except for the last two biggest examples. For s38417 and s15850.1, TurboSYN uses 45 minutes to 1 hour of CPU time due to the big sizes of the circuits.

6 Conclusions and Future Work

We present a new algorithm TurboSYN for FPGA synthesis with retiming and pipelining to minimize the clock period. Instead of minimizing the clock period directly, we propose to minimize the MDR ratio and use pipeline to eliminate all critical I/O paths. We propose a novel positive loop detection technique to enhance the label computation of both the TurboMap [11] and TurboSYN algorithms. The results show significant improvement over existing FPGA mapping and resynthesis algorithms (1.7 to 2 times reduction on the clock periods).

TurboSYN loses on area as compared to TurboMap and FlowSYN-s due to shortcomings of the single-output functional decomposition. Since no sharings of inputs between different LUTs are considered, the more functional decompositions we perform to reduce the MDR ratio, the more LUTs we may generate. Though the single-output functional decomposition is powerful enough for MDR ratio minimization, the multi-output functional decomposition [26] will be useful for area minimization. However, multi-output functional decomposition is more difficult and takes much longer time. We are going to incorporate new logic synthesis methods into our TurboSYN algorithm for area minimization.

7 Acknowledgements

The authors would like to thank Professors Peichan Pan and C. L. Liu for providing preprints of their work and SeqMapII program for the comparative study, Dr. Eugene Ding for providing the FlowSYN program. This work is partially supported by National Science Foundation Young Investigator Award MIP9357582 and grants from Xilinx, Lucent Technologies and Xerox PARC under the California MICRO program and the NYI matching program.

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 Tans. on Computer-Aided Desing,* 6(6):1062–1081, 1987.

[3] S. T. Chakradhar, S. Dey, M. Potkonjak, and S. G. Rothweiler. Sequential Circuit Delay Optimization Using Global Path Delays. In *30th ACM/IEEE Design Automation Conference*, pages 408–489, 1993.

[4] 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*, pages 7–20, 1992.

[5] J. Cong and Y. Ding. Beyond the Combinatorial Limit in Depth Minimization for LUT-Based FPGA Designs. In *IEEE International Conference on CAD,* pages 110–114, 1993.

[6] 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.

[7] 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):145–204, 1996.

[8] 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.

[9] 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.

[10] 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.

[11] J. Cong and C. Wu. An Improved Algorithm for Performance Optimal Technology Mapping with Retiming in LUT-Based FPGA Design. In *IEEE International Conference on Computer Design*, pages 572–578, 1996.

[12] J. Cong and C. Wu. *FPGA Synthesis with Retiming and Pipelining for Clock Period Minimization of Sequential Circuits.* UCLA-CSD 970011, Technique Report, March 1997.

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

[14] Y. Lai, K. R. Pan, and M. Pedram. FPGA Synthesis using Function Decomposition. In *International Conference on Computer Design*, pages 30–35, 1994.

[15] C. Legl, B. Wurth, and K Eckl. A Boolean Approach to Performance-Directed Technology Mapping for LUT-Based FPGA Designs. In *Proc. ACM/IEEE Design Automation Conference.*, pages 730–733, 1996.

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

[17] S. Malik, K. J. Singh, R. K. Brayton, and A. Sangiovanni-Vincentelli. Performance Optimization of Pipelined Logic Circuits Using Peripheral Retiming and Resynthesis. *IEEE Trans. on Computer-Aided Design of Integrated Circuits And Systems*, 12(5):568–578, 1993.

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

[19] 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.

[20] 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.

[21] P. Pan and C. L. Liu. Optimal Clock Period FPGA Technology Mapping for Sequential Circuits with Retiming. *ACM Transactions on Design Automation of Electronic Systems*, to appear.

[22] M. C. Papaefthymiou. Understanding Retiming Through Maximum Average-Delay Cycles. *Mathematical Systems Theory*, 27:65–84, 1994.

[23] 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.

[24] H. Touati, N. Shenoy, and A. Sangiovanni-Vincentelli. Retiming for Table-Lookup Field-Programmable Gate Arrays. In *FPGA'92*, pages 89–94, 1992.

[25] 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.

[26] B. Wurth, K. Eckl, and K. Antreich. Functional Multiple-Output Decomposition: Theory and an Implicit Algorithm. In *Proc. ACM/IEEE Design Automation Conference.*, pages 54–59, 1995.

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