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

A novel algorithm, named SeqMapII, of technology mapping with retiming for optimal clock period for K-LUT based FPGAs was recently proposed by Pan and Liu [13]. The time complexity of their algorithm, however, is $O(K^3n^4 \log(Kn^2) \log n)$ for sequential circuits with n gates, which is too high for medium and large size designs in practice. In this paper, we present three strategies to improve the performance of the approach in [13]: 1) efficient label update with single K-cut computation based on the monotone property of labels that we showed for sequential circuits, 2) a novel approach for the K-cut computation on partial flow networks, which are much smaller in practice, 3) SCC (strongly connected component) partition 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.

1. Introduction

In this paper, we study the technology mapping with retiming problem for *sequential circuits* in K-input lookup tables (K-LUTs) based FPGA designs. Most of the previous LUT mapping algorithms (see the survey paper [4] by Cong and Ding for details) apply only to combinational circuits. For sequential circuits, they assume that the positions of flipflops are fixed so that the whole circuit can be partitioned into several combinational subcircuits. The limitation of this assumption is that one cannot consider the effect of mapping on retiming. However, retiming has been shown to be a very effective technique to move flipflops to reduce the clock period without changing the input-output behavior [10, 11]. Therefore, we study the problem of *finding the mapping solution with the minimum clock period under retiming for sequential circuits.*

Several FPGA synthesis and mapping algorithms have been proposed for sequential circuits [12, 15, 17]. However, those algorithms are heuristics in nature. A significant advancement was made recently by Pan and Liu [13]. They proposed an algorithm named SeqMapII to find a mapping solution with the minimum clock period under retiming. They introduced the idea of expanded circuits 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 solutions 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)[1].

In this paper, we present three strategies to improve the performance of the label computation in SeqMapII [13], which is the most time-consuming step. First, we proved that the monotone property of labels hold for sequential circuits and develop a more efficient label update approach 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 size of our flow networks are always

[1] 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 [14].

less than n, which is a big improvement over $O(Kn^2)$. Finally, SCC (strongly connected component) partition is used to eliminate many redundant label computation and further speed up the algorithm. In practice, our algorithm, named TurboMap, 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.

2. Problem Formulation and Definitions

Give 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. The clock period of a sequential circuit is the maximum delay of combinational paths. The clock period under retiming is the *minimum clock period* which can be achieved with retiming. 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 mapping solution in which the output signals of the LUTs are a subset of those in the original circuit is referred as *simple mapping solution* [15]. Pan and Liu [15] showed 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:

Problem 2 *Given a target clock period c, determine the existence of a simple mapping solution whose* retimed clock period *is c or less.*

Obviously, if Problem 2 can be solved, Problem 1 can be solved with binary or linear search. As in [3], the results in [13, 15] and this paper apply only to K-bounded circuits[2].

We use $G(V, E, W)$ to denote the retiming graph of a sequential circuit [11], 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. 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[3]. It was shown in [13] 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* [13].

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 $\overline{X}$. A cut is called K-cut, if $|V(X, \overline{X})| \leq K$.

3. TurboMap Algorithm

3.1. Review of the SeqMapII Algorithm

The SeqMapII algorithm consists of three steps: 1) label computation, 2) mapping generation, 3) retiming to get the final solution [13].

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 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 [13] presented a procedure IMPROVE(v) to determine the new lower-bound $l_{new}(v)$. They introduced the concept of the expanded circuit for each node. The expanded circuit of node v is used to represent all possible K-LUTs for implementing v 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 v in

[2] When a circuit is not K-bounded, we can use the gate decomposition algorithm, such as balanced tree decomposition [1], dmig [2] or DOGMA [6], to decompose the gates with more than K fanins.

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

$\mathcal{E}_v^i$. 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 in $\mathcal{E}_v^i$ that is not a replication of a PI.

Pan and Liu [13] 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 nodes in the original circuit [13].

In the expanded circuit of node v, based on the current lower-bound $l(u)$ 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 [13], the new lower-bound is computed by:

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

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 $O(K^3n^2log(Kn^2))$. The labels of all nodes can be determined in $O(K^3n^4log(Kn^2))$ time [13].

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.

We improve the SeqMapII algorithm in three ways. 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 reduce the number of iterations. Our algorithm, named TurboMap, will be presented in the following three subsections.

3.2. Label Update with Single K-Cut Computation

In SeqMapII, to compute $l_{new}(v)$ for a node v, it is necessary to perform binary search among $O(Kn^2)$ possible values 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:

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

Obviously, $l'_{new}(v)$ can be computed with only one K-cut computation. Recall this result is similar to Lemma 2 in [3] which applys to combinational circuits only.

The correctness of our approach is based on the fact that $l_{new}(v) \leq l'_{new}(v) \leq l^{opt}(v)$, or in other words, $l'_{new}(v)$ is a tighter lower-bound. This is 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 $e(u, v) \in G_v$, $l^{opt}(v) \geq l^{opt}(u) - c \cdot w(e)$. We have that [8]:

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.*

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 [8]:

Theorem 2 *For a sequential circuit which has a mapping solution with the retimed clock period of no more than c, we have that* $l_{new}(v) \leq l'_{new}(v) \leq l^{opt}(v)$ *at any iteration.*

With the computation of the tighter lower-bound $l'_{new}(v)$ in our algorithm, 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 $l_{new}(v)$ in SeqMapII [13] takes $O(K^3n^4 \log(Kn^2))$ time. To compute $l'_{new}(v)$ in TurboMap takes only $O(K^3n^4)$ time and $l'_{new}(v)$ will converge to $l^{opt}(v)$ with no more iterations than $l_{new}(v)$ need.

3.3. K-Cut Computation in Partial Flow Networks

In this subsection, we present a new approach to compute $l'_{new}(v)$ 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 [13], 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 [3, 5, 13, 15] 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 much smaller as compared to $\mathcal{E}_v^{Kn}$. 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 and the corresponding flow network. For $\mathcal{L}(v)$ based on the current lower-bound, a node u^w in the expanded circuit 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 1(c), the edge direction will be reversed from that in graph G_v (shown in Figure 1(a)). The root v^0 is the source of $\mathcal{E}_f$ and all the u^w for 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 BFS (breadth first search) search. Suppose u^w is the current node. For each fanin edge $e(a,u)$ 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 augmenting path can be found (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 $l'_{new}(v) = \mathcal{L}(v) + 1$).

Let us look at the example of constructing the partial flow network for node c in G_c shown in Figure 1(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 1(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_1^0, i_1^2 and two more edges $(b^0, b_1^0), (i^2, i_1^2)$. Since $l(b) - c \cdot 0 = \mathcal{L}(c) = 1$, the flow capacity of edge (b^0, b_1^0) is ∞. The flow capacity of edge (i^2, i_1^2) is 1 as shown in Figure 1(c). As i is a PI, we add a new edge (i_1^2, 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 1(d). Since b_1^0 was just created in the previous step, we need to build the fanout edges of b_1^0 and add two pairs of nodes a^0, a_1^0 and a^1, a_1^1 when BFS search reaches b_1^0. The new network is shown in Figure 1(e). In the next step, we will build a_1^0 and add two nodes i^0, i_1^0 as shown in Figure 1(f). Since i is a PI, we find another augmenting path. The new residue graph is shown in Figure 1(g).

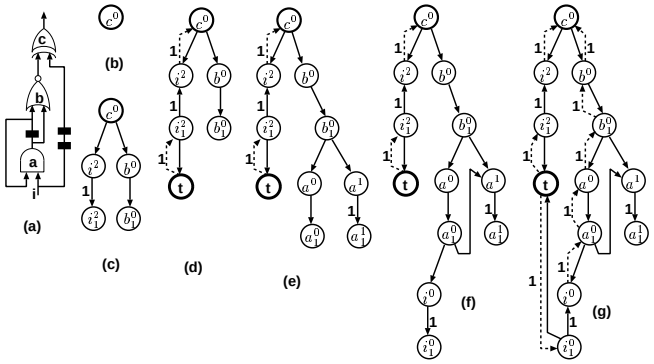

Figure 1. Incremental Construction of the Partial Flow Network with Max-flow Computation. Default flow capacity is ∞.

After finding 3 augmenting paths, there is no more in the residue graph. So the value of max-flow is $3 (= K)$ and therefore, $l'_{new}(c) = \mathcal{L}(c) = 1$.

Since we only search the first $(K+1)$ shortest aug-

menting paths, which are usually very short as compared with Kn, the incrementally constructed flow network will be much smaller than $\mathcal{E}_v^{Kn}$. Our experimental results on MCNC and ISCAS benchmarks show that the average sizes of the flow networks are always far less than n for all 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.

3.4. 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 $v \rightsquigarrow u$ [9]. 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 [9] 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.

3.5. Summary of the TurboMap Algorithm

In preceding subsections, we have presented three strategies to speedup the label computation of SeqMapII algorithm [13]. 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-bounds $l'_{new}(v)$.

To find the minimum clock period, TurboMap performs binary search using the upper-bound of the clock period computed by FlowMap [3] 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 [3] and retiming to minimize the number of flipflops [11].

4. Experimental Results

Our TurboMap algorithm has been implemented in C language on Sun SPAR stations and incorporated into SIS package [16]. The test set includes 13 MCNC FSM benchmarks and 4 ISCAS'89 benchmarks. SIS sequential synthesis commands and dmig [2] are performed to generate the initial circuits. The results are shown in Table 2. Column NODE lists the numbers of nodes (gates+PIs+POs) in the retiming graphs of the initial circuits. Column FF lists the number of flipflops.

Our tests are performed on a SPARC10 workstation with 64MB memory. K is set to 5. FlowMap [3] is performed to get a upper-bound (shown in Column Φ_{FM} in Table 1) of the clock period (Φ) for each example. LUT-packing [3] and retiming [11] are performed as postprocessings to get the final solutions. Table 1 shows the comparison of TurboMap with SeqMapII [13]. SeqMapII has a parameter to set i for $\mathcal{E}^i$. 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 [13] as a heuristic method), respectively for each example. 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[4]. 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 sizes (in terms of numbers of nodes) of our flow networks with $\mathcal{E}^{Kn}$ and $\mathcal{E}^6$ which are used in SeqMapII [13]. The columns with subscripts max and avg list the maximum and average sizes, respectively, of the expanded circuits over all nodes in each example and generated by each algorithm. For the last four examples, we cannot generate $\mathcal{E}^{Kn}$ due to both time and space limitations. The results show that the average sizes of our flow networks

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

circuit	TurboMap				SeqMapII-opt				SeqMapII-heu				
	LUT	FF	Φ	CPU	LUT	FF	Φ	CPU	LUT	FF	Φ	CPU	Φ_{FM}
bbara	13	9	3	0.5	12	9	3	180.8	19	9	4	2.8	4
dk15	7	2	1	0.3	7	2	1	155.1	23	3	6	4.6	1
dk16	104	18	14	11.2				***	113	15	14	71.5	14
dk17	8	5	1	0.4	6	3	1	293.0	14	4	4	7.3	2
kirkman	57	17	5	0.7	65	23	5	5373.3	66	22	5	4.8	6
ex1	93	16	8	1.6	89	24	8	55653.0	89	24	8	9.1	8
s1	60	7	7	0.9	62	12	7	22435.7	62	12	7	12.6	7
sse	52	12	6	0.4	50	16	6	4508.4	50	16	6	6.6	7
keyb	80	13	9	1.5	89	14	9	78534.4	89	14	9	17.4	10
styr	173	10	16	8.4				***	190	16	16	36.6	17
sand	179	36	15	13.8				***	211	27	15	96.4	16
planet1	222	26	18	20.3				***	236	42	18	118.9	19
scf	342	26	13	32.5				***	343	101	13	101.9	14
s9234	525	238	4	102.0				***	499	282	4	226.6	5
s5378	490	292	4	92.3				***	457	280	4	145.1	4
s15850.1	1541	811	8	511.5				***	1525	844	8	732.4	8
s38417	3988	2552	6	4056.8				***	4002	2706	6	9599.7	8
total	7934	4090	138	4855.2	—	—	—	—	7988	4417	147	11595.1	150
ratio	1	1	1	1	—	—	—	—	1.01	1.08	1.07	2.39	1.09

Table 1. Performance comparison between TurboMap and SeqMapII. Entry "*" means no result after running 24 hours.**

are only slightly larger than $\mathcal{E}^6$ and 823 times smaller than $\mathcal{E}^{Kn}$. This result, together our efficient label update and SCC partition, give an explanation why TurboMap is significantly faster than SeqMapII-opt and even 2 times faster than SeqMapII-heu.

5. Conclusions

We present a new algorithm for technology mapping with retiming for optimal clock period as an improvement of the SeqMapII algorithm by Pan and Liu [13]. We showed the monotone property of node labels and propose an efficient label update with single K-cut computation based on partial flow networks. SCC partition is used to further speedup the algorithm. The experimental results show that TurboMap is 2×10^4 times faster than SeqMapII. In our future work, we want to develop an efficient algorithm to compute the initial state of the mapping solution. Also we are in the process of integrating the TurboMap algorithm into RASP [7].

6. Acknowledgment

The authors thank very much Professors Peichan Pan and C. L. Liu for providing 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

[1] 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.

[2] 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.

[3] J. Cong and Y. Ding. FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. *IEEE Trans.*

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
subtotal			2907	1341	1153373	1103982	2024	1147
ratio			1	1	396.8	823.3	0.7	0.9

Table 2. Comparison of the sizes of expanded circuits used by TurboMap and SeqMapII. Entry "*" means no results after running the algorithm for 24 hours.**

on Computer-Aided Design of Integrated Circuits And Systems, 13(1):1–12, 1994.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] J. Cong and C. Wu. *An Improved Algorithm for Technology Mapping with Retiming for Lookup-Table Based FPGAs.* UCLA-CSD 960012, Technique Report, 1996.

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

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

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.