Area-Minimal Algorithm for LUT-Based FPGA Technology Mapping with Duplication-free Restriction

Chi-Chou Kao
Department of Information Technology
National Pingtung Institute of Commerce
Pingtung, Taiwan

Yen-Tai Lai
Department of Electrical Engineering
National Cheng Kung University
Tainan, Taiwan

Abstract - Minimum area is one of the important objectives in technology mapping for lookup table-based FPGAs. It has been proven that the problem is NP-complete. This paper presents a polynomial time algorithm which can run in $O(n^3)$ time to generate an efficient solution where n is the total number of gates in the circuit. The proposed algorithm partitions the graph representing the given circuit into subgraphs such that the solution can be obtained by merging the subgraph solutions. The greedy technique is then used to find the solution for each subgraph. It is shown that except for some cases the greedy method can find an optimal solution of a given problem. We have tested our algorithm on a set of benchmark examples. The experimental results demonstrate the effectiveness of our algorithm.

I. Introduction

The merits of low cost and short turnaround time have made field programmable gate arrays (FPGAs) an important technology in VLSI designs. In an FPGA device, a configurable logic block (CLB) contains a k-input lookup table (LUT) and can implement any Boolean function whose input number must not exceed k, k being a function of the CLB hardware constraints on the number of inputs.

The technology mapping problem for LUT-based FPGA is to produce an equivalent circuit for a given circuit using gates that can be implemented with LUTs. The optimization objectives of this problem can be area [1-2], performance [3-5], routability [7-8], and etc. For performance optimization, the algorithm proposed in FlowMap [5] is sophisticated. It assumes that the path delay is directly proportional to the number of logic levels and ensures the optimal solution can be found in polynomial times.

In an area mapping solution, duplication of gates in the given circuit is usually allowed. However, using a two-stage design process, it is possible to find a solution without duplication first, and then find the part that can be duplicated to reduce the total number of LUTs. Thus in this paper, we focus on the first stage of the process in which gates are implemented by one and only one LUT and present an efficient algorithm for finding the area-minimum solution. This proposed algorithm first uses a partitioning algorithm that divides the given circuit system into subsystems such that one subsystem has only one output and the union of the solutions for these subsystem is the solution of the whole system.

Unlike performance optimal mapping, the technology mapping problem for minimizing area cannot be found an optimal mapping solution. It is shown that the problem of area-optimal mapping is NP-complete [9-10]. A greedy approach is used to find the mapping solution for each subsystem. Two kinds of subgraphs of the graph representing a subsystem are selected, one at a time, in the procedure of the greedy method. It is shown that except for some cases this method can find the optimal mapping of the subsystem. The time complexity of this algorithm is bounded by $O(n^3)$, where n is the total number of gates in the given circuit. The experimental results on a number of MCNC benchmarks demonstrate the efficiency of this algorithm.

The remainder of this paper is organized as follows. Section II describes the terminology and problem formulation. The area-optimal algorithm for duplication-free mapping solution is presented in Section III. Experimental results are shown in Section IV. Finally, in Section V we present concluding remarks.

II. Terminology and Problem Formulation

A combinational logic circuit can be represented by a directed acyclic graph (DAG), $G = (V_g \cup V_{io}, E)$. A vertex in V_g represents a logic gate, while a vertex in V_{io} represents a pseudo gate that is either a primary input or a primary output. A directed edge $<i, j>$ exists in E if the output of gate i is the input of gate j. Notice that a primary input vertex has no in-coming edge and a primary output vertex has no out-going edge. Figure 1.*b* shows the corresponding DAG of the combinational circuit illustrated in Figure 1.*a*. A vertex in V_{io} is represented by a square box and a vertex in V_g by a circle in Figure 1.*b*.

Let v and u be two vertices of V_g. If there is a directed path from v to u, v is said to be a *predecessor* of u and u is a *successor* of v. If v is connected to u by a single edge, v is said to be a *fan-in vertex* of u and u is a *fan-out vertex* of v. Let V_s be a subset of V_g and $\overline{V_s} = V - V_s$. A *fan-in signal* of V_s is a signal associated with an edge directed from a $\overline{V_s}$ vertex to a V_s vertex. A *fan-out signal* of V_s is a signal associated with an edge directed from a V_s vertex to a $\overline{V_s}$ vertex. It is defined that *Input*(V_s) represents the set of fan-in signals of V_s. Similarly, *Output*(V_s) represents the set of fan-out signals. The output signal of a gate may feed to more than one gate. The signal is said to have *multiplicity fanout* at the gate. Equivalently, a vertex in G may have more than one out-edge. Such a vertex is called a *multiplicity vertex*. The output signal from a multiplicity vertex propagates through distinct paths and ends at either 1) more than one primary output vertex or 2) one-output vertex. The multiplicity vertices are called *multiple-fanout sources* in the first case and *reconvergent source* in the second case. For example, in Figure 1.*b*, s is a reconvergent source and u and x are multiple-fanout sources. Notice that a vertex can be a multiple-fanout source as well as a reconvergent source.

Assume C_v is the subgraph induced by V_v. The subgraph C_v is said to be a *cone* if there exists a vertex $v \in V_v$ such that for every vertex $u \in C_v$ there is a directed path from u to v in C_v.

The vertex v is called the *tip* of the cone. A cone, C_v, is said to be a *single output cone* (*SO cone*) if $|Output(V_v)| = 1$. A cone, C_v, is said to be *k-feasible* if it is a SO cone and $|Input(V_v)| \leq k$. Since C_v is the induced subgraph of V_v, the set of fan-in signals of C_v is exactly equal to $|Input(V_v)|$. For convenience, $Input(V_v)$ and $Input(C_v)$ are used simultaneously and interchangeable in the rest of this paper. Let $C_u = (V_u, E_u)$ and $C_v = (V_v, E_v)$ be two cones. C_u and C_v are said to be *overlaped* if $V_u \cap V_v \neq \emptyset$; C_u is said to be *covered* with C_v if $V_u \subseteq V_v$.

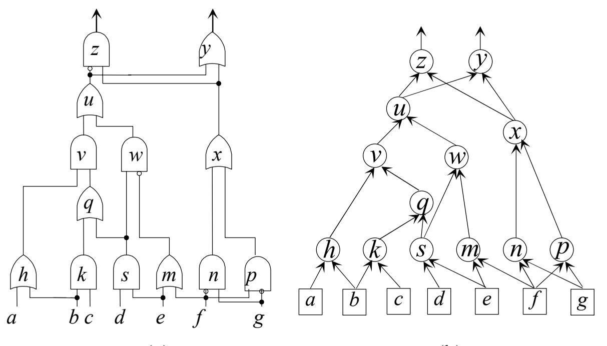

Fig. 1: *a*) A combinational circuit; *b*) The DAG corresponding to the circuit in (*a*).

A SO cone tipped at v, C_v, is called a *primary block* if it covers all SO cones tipped at v and v is either a primary output vertex or a multiple-fanout source. For example, in Figure 1.*b*, the cone, C_x, containing vertices x, n, and p is a primary block.

A network is said to be *k-bounded* if the in-degree of every vertex is less than or equal to k in the network. In the rest of this paper, it is assumed that the given DAG is transformed into a 2-input simple gate network by using the decomposition algorithm [4]. Formally, a collection of k-feasible cones is said to be a *mapping solution* of $G = (V_g \cup V_{io}, E)$ if each vertex in V_g is included in one and only one cone. Therefore, the technology mapping problem for LUT-based FPGA designs can be formulated as a graph-covering problem as follows:

Given a 2-bounded network, find a mapping solution such that the number of cones in the mapping is minimum.

III. Outline of the Mapping Algorithm

The divide-and-conquer approach and greedy method are two major techniques used in the algorithm. The divide-and conquer approach is usually used to reduce the complexity of computation. To use this approach, we must be able to partition the given problem into sub-problems such that each sub-problem can be solved independently and the solutions for the sub-problems can be combined to be the solution of the whole problem.

The proposed algorithm is divided into two steps: 1) partitioning the graph representing the given circuit system into primary blocks, and 2) using the greedy method to find the solution for each primary blocks.

To use greedy method to find the optimal mapping for a primary block G_B, we must find a k-feasible cone C_t such that the optimal solution of G_B is the union of $\{C_t\}$ and the optimal solution of $G_B - C_t$. Subgraphs are selected in the iterations of the greedy procedure. It is shown that in an iteration the selection of a subgraph of a special kind can always lead to an optimal solution. However, a subgraph of this special kind may not exist in the selection procedure. In such a case, a subgraph of the second kind is selected. It is shown that except for some particular cases the selection of the second kind can also lead to finding the optimal solution.

IV. Partitioning the Given DAG into Primary Blocks

According to the algorithm [6], it can be shown that 1) the primary blocks of a given DAG are mutual-exclusive, 2) the union of all primary blocks includes all vertices, and 3) for a SO cone, C_k, there is one and only one primary block, C_w, such that $C_k \subseteq C_w$. Therefore, the mapping solutions for the primary blocks can be combined to be the solution of the whole DAG.

To partition the given DAG into primary blocks, we label the vertices in the given DAG such that the vertices in a primary block have the same label. Starting from the primary output vertices, a traversal of the graph in topological order can label the vertices. A vertex is labeled with the same tag as its fan-out vertices if all its fan-out vertices have the same label; it is labeled with new tag otherwise. Figure 2 shows the result of labeling vertices of the graph in Figure 1.*b*. It is seen that in Figure 2 the fan-out vertices of vertex u have different labels and therefore vertex u is labeled with a new tag.

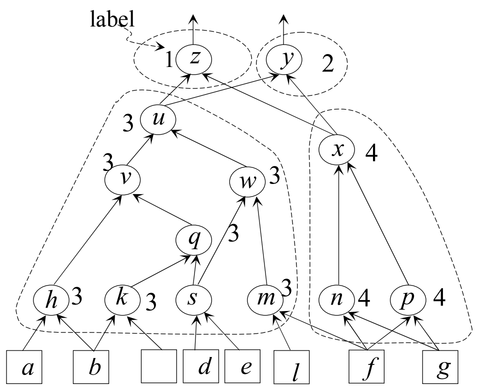

Fig. 2. Partitioning a given DAG into primary blocks.

According the labeling rule, a vertex that is labeled with a new tag must be a primary output vetex or a multiple fanout source. Hence, the subgraph induced by the vertices with the same label must be a SO cone tipped at a primary output vertex or a multiple-fanout source. Therefore, the subgraph induced by the vertices with the same label is a primary block. In this procedure, every edge is traversed once. The time complexity is bounded by $O(e)$.

V. The Mapping Solution for a Primary Block

In this subsection, we will use the greedy method to find the mapping solution for a primary block G_B. It is discussed whether the selection of a k-feasible cone in each iteration of the greedy method will lead to finding the optimal mapping solution or not.

A cone is called a *floor cone* if all its fan-in vertices are primary input vertices. In Figure 3.*a*, the cone including the vertex r and all its predecessors is a floor cone.

***Theorem* 1:** There exists an optimal mapping solution, M_B, in G_B

such that every feasible floor cone is covered with a cone in M_B.

***Proof*:** Assume that C_w is a feasible floor cone tipped at w and it is not covered with a single cone in an optimal mapping solution M_B. Let C_v be the cone in M_B that covers w. There are two cases with v and w: 1) $v = w$ and 2) $v \neq w$. Consider the first case. Assume $C_v \neq C_w$. Because C_w includes all predecessors of w, $C_v \subset C_w$. Let $G_v = C_w - C_v$ and S_v be the set of cones covering G_v in M_B. It is obvious that $\bigcup_{C \in S_v} C = G_v$. If we replace C_v and all the cones in S_v with C_w, the number of cones in the new mapping solution decreases. This is a contradiction to the assumption that M_B is an optimal mapping solution. Therefore, if $v = w$, then $C_v = C_w$. The second case is shown in Figure 4. Let $C'_w = C_w \cap C_v$, $G_u = C_w - C'_w$, and $C'_v = C_v - C'_w$. Since w is the only vertex in C_w that has fan-out signal to C'_v and $|Output(G_u)| \geq 1$, $|Input(C'_v)| = |Input(C_v)| - |Output(G_u)| + 1 \leq |Input(C_v)|$. Therefore, C'_v is a k-feasible cone. Let S_C be the set of cones covering G_u in M_B. Since $|Output(G_u)| \geq 1$, $|S_C| \geq 1$. Accordingly, if we replace C_v and the cone in S_C with C'_v and C_w, the total number of cones in M_B does not increase. ■

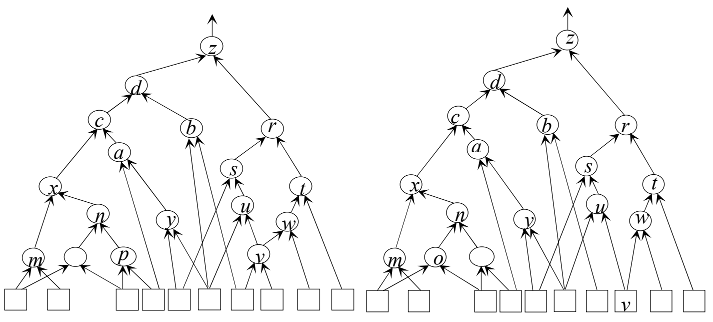

Fig. 3. *a*) A given DAG ; *b*) A new DAG derived from (*a*) by selecting a critical floor cone as a cone in the optimal mapping solution (k =5).

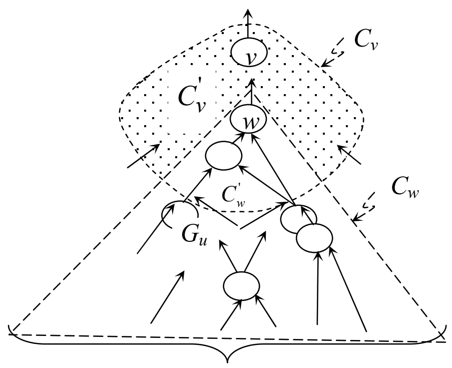

Fig. 4. A feasible floor cone is not covered with a cone in an optimal mapping solution in G_B.

Let C_v be a k-feasible floor cone tipped at v. If the floor cone tipped at any successor of v is not k-feasible, C_v is called a *saturated floor cone*. In Figure 3.*a*, the floor cone including the set of the vertices $\{x, m, n, o, p\}$ is an example of a saturated floor cone. The floor cone including the set of the vertices $\{n, o, p\}$ is a floor cone but not a saturated floor cone because x is a successor of n and the floor tipped at x is a k-feasible floor cone. A saturated floor cone C_v is said to be a *critical floor cone* if it cannot be covered with any other feasible cones. For example, in Figure 3.*a* C_v including the vertex v is a critical floor cone. However, in Figure 3.*a*, the floor cone including the set of the vertices $\{x, m, n, o, p\}$ is a saturated floor cone but not a critical floor cone.

***Theorem* 2:** If there exists a critical floor cone in G_B, it can be selected to be in the optimal mapping solution.

***Proof*:** This is a corollary of Theorem 1. Assume C_r is a critical floor cone. According to Theorem 1, there exist an optimal mapping solution that has a cone, C_s, covering C_r. Since no feasible floor cone is larger than the critical floor cone, $C_s = C_r$.■

According to Theorem 2, the critical floor cone C_v in Figure 3.*a* can be selected to be in the optimal mapping solution. On the other hand, if there are no critical floor cones, we must find another kind of cone that can be selected to be in the mapping solution. A vertex is called a *leading vertex* if 1) each fan-in vertex is either the tip of a saturated floor cone or a primary input vertex, and 2) at least one fan-in vertex is the tip of a saturated floor cone. For example, in Figure 3.*b* the vertex c is a leading vertex.

***Lemma* 3:** There must exist a leading vertex in G_B.

***Proof*:** As shown in Figure 5, let u be a fan-out vertex of a saturated floor cone C_w and U_i be the set of fan-in vertices of u. It is to be shown that if every floor cone tipped at a vertex in U_i is feasible, u is a leading vertex. Otherwise, there must exist a leading vertex that is a predecessor of u.
Consider the case that every floor cone tipped at a vertex in U_i is feasible. Let $v \in U_i$, $v \neq w$, and C_v be a feasible floor cone tipped at v. There are two sub-cases: 1) v is not a multiplicity vertex and 2) v is a multiplicity vertex. In the first sub-case, the successor of w is also the successor of v. Since C_w is a saturated floor cone, the floor cone tipped at a successor of v is not feasible. Hence, C_v is a saturated floor cone. Consider the second sub-case. Assume C_v is not a saturated floor cone and there exists a saturated floor cone, C_x, tipped at a successor of v. Obviously, C_x must cover u and C_w. However, it is a contradiction to the hypothesis that C_w is a saturated floor cone and it cannot be covered with any other saturated floor cones. Therefore, C_v is a saturated floor cone and u is a leading vertex.
Next we have to consider the case that at least one floor cone tipped at a vertex in U_i is infeasible. Let $v \in U_i$ and C_v be an infeasible floor cone covering v and all its predecessors. If u is not a leading vertex, there must in G_B exist an infeasible floor cone C_v. Since C_v is a floor cone and any floor cone tipped at a successor of v is not feasible, there are saturated floor cones in C_v. The case is shown in Figure 6. Let u' be a fan-out vertex of a saturated floor cone in C_v. According to the previous discussion in G_B, if u' is not an out-reach vertex, there must in C_v exist an infeasible floor cone C'_v. Let u'' be a fan-out vertex of a saturated floor cone in C'_v. If u'' is not an out-reach vertex, there is an infeasible floor cone C''_v in C'_v. Recursively, a leading vertex whose every fan-in vertex is the tip of a feasible floor cone can be found. ■

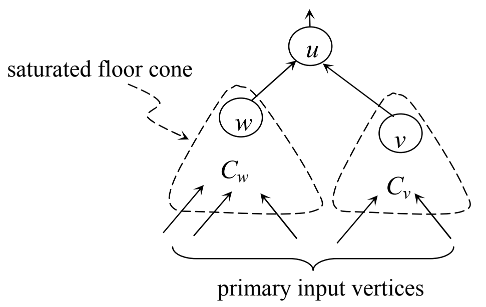

Fig. 5. Every floor cone tipped at a vertex in U_i is feasible.

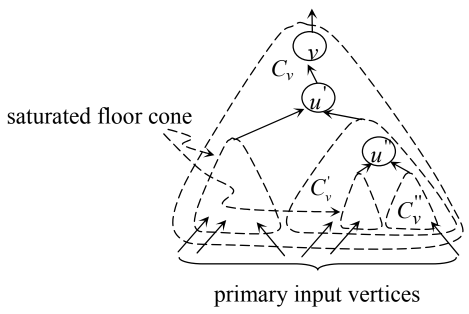

Fig. 6. There exists a floor cone tipped at a vertex in U_i is infeasible.

Let S_v be a set of saturated floor cones whose fan-out vertex is a leading vertex v. A cone $C_i \in S_v$ is called the *prime cone* if $|Input(C_i)|$ is the largest in S_v. A prime cone is said to be *critical* if there exist no feasible cone covering the prime cone and other saturated floor cones. For example, in Figure 3.*b*, the vertex *c* is a leading vertex and the floor cone tipped at *x* is the critical prime cone.

***Theorem* 4:** If there are no critical floor cones , then a critical prime cone can be selected to be in the optimal mapping solution.

***Proof*:** By Lemma 3, there must exist a leading vertex. Let C_w be a critical prime cone tipped at *w*. C_w is a feasible floor cone. According to Theorem 1, there exists an optimal mapping solution M_B which has a cone C_v covering C_w. Let *v* be the tip of C_v. Since no critical floor cones exist, $v \neq w$. Hence, there is a path from *w* to *v*. Equivalently, *v* is a successor of *w*. As in Figure 7, assume that the leading vertex has a fan-in signal from a feasible floor cone C_u and $|Input(C_u)| \leq |Input(C_w)|$. Since C_w is a critical prime cone, there exist no feasible cones to cover both C_u and C_w. In other words, C_u must be a cone in M_B. Let $C_v' = C_u \cup (C_v - C_w)$. $|Input(C_v')| = |Input(C_u)| + 1 \leq |Input(C_w)| + 1 = |Input(C_v)| \leq k$. Hence, we can obtain a new mapping solution by replacing C_v and C_u with C_w and C_v'. The new mapping solution will have the same number of cones. Equivalently, the new mapping solution is also an optimal one. ■

According to Theorem 2 and 4, the greedy method can be used to find the optimal solution. Figure 8 illustrates the procedure of finding an area-optimal mapping solution. We first find the leading vertex *c* and the prime cone C_x including the set of the vertices $\{x, m, n, o, p\}$. The cone C_x is selected to be in the mapping solution. A new DAG $G_B = G_B - C_x$ is obtained. In the new iteration of recursive process, G_B is set to be $G_B - C_x$. The tip of C_x is considered as a primary input vertex of new G_B. The vertex *d* is the leading vertex in the new G_B. The saturated floor cone C_c including the set of the vertices $\{c, a, y\}$ is a prime cone and is selected to be in the mapping solution. Again set G_B to be $G_B - C_c$. The subgraph induced by the set of vertices $\{b, d\}$ is selected to be in the mapping solution. It is seen that the mapping solution is an optimal one including three cones.

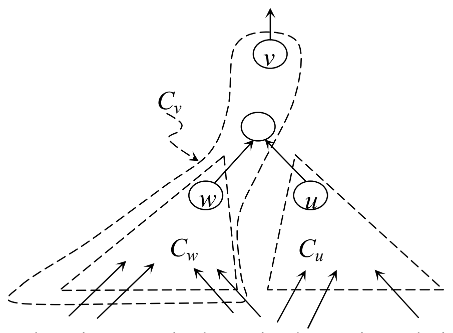

Fig. 7. The prime cone in the optimal mapping solution.

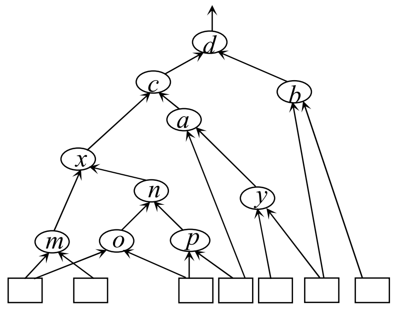

Fig. 8. An example of finding an optimal mapping solution by the greedy method (k =5).

If a prime is not critical, selecting it may lead to a non-optimal solution. For example, in Figure 9 the prime cone C_y including the vertex *y* is not critical. If we select C_y to be in the mapping solution, a new DAG $G_B = G_B - C_y$ is obtained. In the new iteration of recursive process, G_B is set to be $G_B - C_y$. The tip of C_y is considered as a primary input vertex of new G_B. The vertex *d* is the leading vertex in G_B. The saturated floor cone C_c tipped at *c* is a critical prime cone and is selected to be in the mapping solution. Again set G_B to be $G_B - C_c$. The subgraph induced by the set of vertices $\{b, d\}$ is selected to be in the mapping solution. It is seen that the mapping solution includes three cones. However, if we use the cones including the sets of the vertices $\{a, b, c, d, y\}$ and $\{m, n, o, p, x\}$ to cover G_B, the number of the cones needed is only two. Hence, the greedy method can not be used to find the optimal solution in Figure 9. Nevertheless, it is difficult to determine whether a prime cone is critical or not. Its time complexity can be exponential. In practice, most prime cones are critical in our experience. Even more, a good mapping solution can be obtained in the experimental results shown in Section VI if there exist no critical cones and we select a prime cone in the mapping solution.

Based on the above discussion, we need an algorithm to find the critical floor cones and the prime cone. To find those cones, we must find the saturated floor cones that can be generated by inspecting all *k*-feasible floor cones.

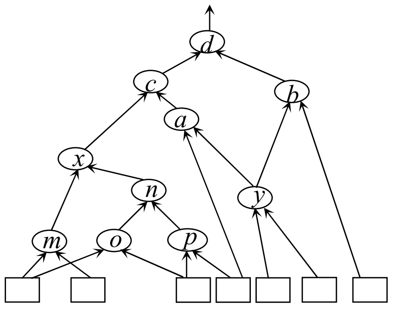

Fig. 9. A feasible cone covers the leading vertex and its all predecessors (k =5).

It is easy to find all k-feasible floor cones. Let $Pre(v)$ denote the predecessors of v and $Fin(v)$ denote the set of fan-in vertices of v. Let $SP(v) = \bigcup_{u\in Fin(v)} Pre(u)$. Clearly, $Pre(v) = \{v\}\cup SP(v)$. Accordingly, starting from the tip of a primary block G_B, all k-feasible floor cones can be found by traversing G_B in post-order. Assume that C_v is a floor cone induced by $Pre(v)$. Recall C_v is k-feasible if and only if $|Input(C_v)| \le k$ and $|Output(C_v)| = 1$. The procedure to find the k-feasible floor cones in G_B is as follows:

```
Algorithm Generating Floor Cones:
    GenerateFloorCones ($G_B$)
    Comment: $G_B = (V, E)$ is a DAG
    begin
        $stack \leftarrow \emptyset$ ;
        for every vertex $v$ in $V$ do begin
            $timesVisited(v) \leftarrow 0$
            if ($v$ is a primary input vertex) then push $v$ onto
             $stack$
        end of for-loop
        while stack $\neq \emptyset$ do begin
            $vx \leftarrow$ pop($stack$) ;
            $Pre(v) = \{v\}\cup SP(v)$ ;
            Construct $C_{vx}$ which is the induced subgraph by
            $Pre(v)$ ;
            if ( $|Input(C_{vx})| \le k$ and $|Output(C_{vx})| = 1$) then
                add $C_{vx}$ to the list of all $k$-feasible floor cones
          for every fan-out vertex of $vx$, $vs$, do begin
            $timesVisited(vs) \leftarrow timesVisited(vs) + 1$ ;
            if ($timesVisited(vs)$ = indegree of $vs$) do
                push $vs$ onto $stack$ ;
          end of for-loop
          end of while-loop
    end of GenerateAllFloorCones
```

***Theorem* 5:** The complexity of generating all k-feasible floor cones in G_B is $O(n^2)$, where n is the total number of vertices.

***Proof*:** In the procedure, every edge is traversed once. For every vertex v, we must check whether the union of $\{v\}$ and $SP(v)$ is a feasible cone or not. Therefore the complexity of generating all k-feasible floor cones is bounded by $O(n^2)$.■

***Theorem* 6:** Given a 2-bound network, the time complexity of the area-optimal mapping algorithm is $O(n^3)$ where n is the number of vertices.

***Proof*:** The PrimaryBlock_Mapping algorithm must be executed recursively to find the cone of a mapping solution in a given primary block. One iterative in the recursion finds one cone. The maximum depth of recursion is less than or equal to n. The time complexity of finding a cone is $O(n^2)$. Therefore, for a primary block, the upper bound of the complexity of the greedy method is $O(n \cdot n^2)$. The time complexity of partitioning the graph into primary blocks is $O(e)$ where e is the total number of edges. The total number of edges is less than n^2. Hence, the total time complexity of the area-optimal mapping algorithm is $O(n^3)$.■

VI. Experimental Results

The proposed algorithm was implemented in C language on a SUN SPARC 2. We set k to be five. Testing was accomplished by having the algorithm experimentally design several circuits from the MCNC logic synthesis benchmark set. Prior to use of these algorithms, a MIS-II [12] environment was used to reduce the complexity of the given network and produce a 2-bounded general Boolean network. Table I presents the comparison of run-times of the proposed algorithm and the DFMap algorithm [6], which can generate optimal duplication-free area-mapping solution, i.e., gates are implemented by one and only one LUT, by using the dynamic programming method. It is seen that the proposed algorithm takes less 117% CPU time than the DFMap on average. Furthermore, for the special circuits such as 9symml, the DFMap takes more 60 times than the proposed algorithm.

Table I. Comparison of the CPU run-time with DFMap

CPU Time (seconds)			
Circuit	ckt size	Proposed algorithm	DFMap
9symml	151	0.19	12.2
C3540	956	0.20	5.5
alu2	275	0.05	1.2
alu4	540	0.08	1.8
apex6	699	0.06	2.0
C880	302	0.03	0.5
rot	436	0.03	0.6
i7	340	0.03	0.6
C499	370	0.02	1.0
duke2	181	0.03	0.3
rd84	108	0.02	0.3
C5315	1454	16.89	26.0
C6288	2353	0.30	0.3
C7552	2118	57.45	71.7
s1196	481	0.03	0.8
s1494	558	0.09	3.4
s5378	1074	0.19	5.2
des	2244	0.22	4.2
Total		78.76	170.8
Comparison		1	+117%

We compare the number of LUTs and depths generated by the proposed algorithm with those obtained by other FPGA technology mapping algorithms such as the DFMap [6], Chortle-crf [2], and FlowMap [5]. Among the algorithms, the proposed algorithm and the DFMap are duplication-free mapping and Chortle-crf and FlowMap allow duplication of gates. Table II presents the experimental results. The results show that the number of LUTs generated by the proposed algorithm is close to the optimal area-mapping solution found by the DFMap and reduces 23% LUTs number compared with FlowMap. Because the Chortle-crf allows duplication of gates, our algorithm generates 20% more LUTs number than the Chortle-crf.

Table II. Comparison of the number of LUTs (A) and depths (D) with DFMap, Chortle-crf, and FlowMap

Circuit \ Mapper	Proposed algorithm		DFMap		Chortle-crf		FlowMap	
	A (L_n)	D	A	D	A	D	A	D
9symml	73	9	69	7	60	9	77	5
C3540	425	18	399	21	319	16	549	10
alu2	125	14	120	16	111	19	171	9
alu4	230	14	220	18	196	21	300	10
apex6	229	14	220	18	193	18	300	10
C880	117	9	117	12	94	14	161	7
rot	202	9	201	12	203	14	266	7
i7	146	4	146	4	107	2	139	2
C499	66	5	66	5	70	6	74	4
duke2	131	8	129	9	116	8	190	4
rd84	46	7	44	7	40	7	44	4
C5315	473	12	448	11	439	12	690	8
C6288	1425	90	1425	90	494	29	760	22
C7552	676	11	645	10	644	10	741	7
s1196	204	12	200	12	166	10	211	5
s1494	230	7	219	6	188	8	260	4
s5378	483	8	473	8	420	9	527	5
des	1081	10	1068	10	950	11	1552	5
Total	9231	274	9111	288	7365	232	11398	155
Comparison	1	1	-1%	+5%	-20%	-15%	+23%	-43%

Among the algorithms in Table II, our algorithm is aimed primarily at minimum number of LUTs, while CutMap and FlowMap focus on minimizing the depth of the mapping solution. The experimental results show that CutMap and FlowMap generate better solution than our algorithm in depth. These comparisons show that a good mapping algorithm that should first consider the primary objective and then preferably allow controllable trade-off among all objectives. Otherwise, the mapping results will be of low quality for the primary objective.

To demonstrate the effectiveness of our algorithm, Table III presents the number of critical floor cones and prime cones of every benchmark. According to the above discussions in Section II and III, a selection of a prime cone whose fan-out vertex is not a critical leading vertex can lead to non-optimal solution. In Table III, we find the number of the prime cones whose leading vertex is non-critical does not exceed 6% of the total LUTs number. Even more, the circuits C880, i7, rot, and C499 generate optimal solutions. Therefore, we believe that good mapping solutions can be obtained by using our algorithm.

References

[1] J. Francis, J. Rose, and K. Chungm "Chortle: A Technology Mapping Program for Lookup Table-Based Field programmable Gate Arrays," *Proc, 27th ACM/IEEE Design Automation Conference*, pp. 613-619, June 1990.

[2] J. Francis, J. Rose, and Z. Vranesic, "Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs," *Proc, 28th ACM/IEEE Design Automation Conference*, pp. 248-251, June 1991.

[3] R. J. Francis, J. Rose, and Z. Vranesic, "Technology Mapping for Lookup Table-Based FPGAs for performance," *Proc. IEEE International Conf. Computer Aided Design*, pp. 568-571, Nov. 1991.

[4] J. Cong, Y. Ding, A. Kahug, and P. Trajmar, "An Improved Graph-Based FPGA Technology Mapping Algorithm for Delay Optimization," *Proc. ICCD*, pp. 154-158, Oct.1992.

[5] J. Cong, Y. Ding, "FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, pp. 1-11 Vol.13 No. 1, January 1994.

[6] J. Cong. And Y. Ding, "On area/depth trade-off in LUT-Based FPGA technology mapping," *IEEE Transactions on VLSI Systems*, pp. 137-148 Vol.2 No. 2, June 1994.

[7] N. B. Bhat, and D. D. Hill, "Routable Technology Mapping for LUT FPGAs," *Proc. ICCD*, pp. 95-98, Oct. 1992.

[8] M. Schlag, J. Kong, and Pak K. Chan, "Routability-Driven Technology Mapping for Lookup Table-Based FPGAs", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System*, pp.13-26 Vol.13 No. 1, January 1994.

[9] H. Farrahi and M. Sarrafzadeh, "Complexity of the Lookup-Table Minimization Problem for FPGA Technology Mapping," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System* pp.1319-1332 Vol.13 No. 11, November 1994.

[10] S. Zhang, D. Michael Miller, and J. C. Muzio, "Notes on Complexity of the Lookup-Table Minimization Problem for FPGA Technology Mapping," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System* pp.1588-1590 Vol.15 No. 12, December 1996.

[11] R.Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Improved Logic Synthesis Algorithms for Table Look Up Architectures," *Proc. IEEE International Conf. Computer-aided Design*, pp. 564-567, Nov.1991.

[12] R. K. Brayton, R. Rudell, and A. L. Sangiovanni-Vincentelli, "MIS: A Multiple-level Logic Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and System* pp.1062-1081, Nov. 1987.

Table III. Circuit optimization using the greedy method

Circuit	critical floor cones		prime cones whose leading vertex is critical		prime cones whose leading vertex is non-critical	
	A (C_N)	Ratio ($\frac{C_N}{L_N}$)	A (P_{CN})	Ratio ($\frac{P_{CN}}{L_N}$)	A (P_{NCN})	Ratio ($\frac{P_{NCN}}{L_N}$)
9symml	60	82%	9	12%	4	5%
C3540	182	43%	27	6%	26	6%
alu2	65	52%	11	9%	5	4%
alu4	114	50%	22	10%	10	4%
apex6	112	49%	24	10%	9	4%
C880	57	49%	17	15%	0	0%
rot	61	30%	21	10%	1	0%
i7	102	70%	30	21%	0	0%
C499	8	12%	8	12%	0	0%
duke2	42	32%	8	6%	2	2%
rd84	24	52%	2	4%	2	4%
C5315	257	54%	15	3%	25	5%
C6288	0	0%	0	0%	0	0%
C7552	315	47%	4	1%	31	5%
s1196	90	44%	12	6%	4	2%
s1494	153	67%	35	15%	11	5%
s5378	178	37%	32	7%	10	2%
des	497	46%	172	16%	13	1%