Efficient LUT-Based FPGA Technology Mapping for Power Minimization

Hao Li, Wai-Kei Mak, and Srinivas Katkoori
Department of Computer Science and Engineering
University of South Florida, Tampa, FL USA
(hli5,wkmak,katkoori@csee.usf.edu)

Abstract— We study the technology mapping problem for LUT-based FPGAs targeting at power minimization. The problem has been proved to be NP-hard previously. Hence, we present an efficient heuristic to compute low-power mapping solutions. The major distinction of our work from previous ones is that while generating a LUT, we look ahead at the impact of the mapping selection of this LUT on the power consumption of the remaining network. We choose the mapping that results in the least estimated overall power consumption. The key idea is to compute low-power K-feasible cuts by an efficient incremental network flow computation method. Experimental results show that our algorithm reduces both power consumption and area over the previous algorithms reported in the literature.

I. Introduction

Field-programmable gate array (FPGA) is now widely used for VLSI system design and rapid system prototyping due to its short design cycle. An FPGA consists of an array of uncommitted programmable logic blocks, programmable interconnections, and I/O pads. The lookup-table (LUT) based architecture is the most popular architecture. In LUT-based FPGAs, a K-input lookup table (K-LUT) is the basic programmable logic block which can be programmed to implement any Boolean function of up to K variables [3]. The technology mapping problem in LUT-based FPGA designs is to cover a general Boolean network with K-LUTs yielding a functionally equivalent K-LUT network.

Previous LUT-based FPGA technology mapping algorithms can be broadly classified according to their primary optimization objectives: algorithms that minimize area (i.e., minimizing the number of LUTs) [8, 9, 10, 11], algorithms that minimize delay (i.e., minimizing the number of LUTs on a longest path) [4, 5, 12, 14], algorithms that focus on routability [15, 16], and algorithms that minimize both area and delay [19, 20]. However, relatively little work has been done on minimizing power consumption. In one of the works on low-power technology mapping for LUT-based FPGAs, Farrahi and Sarrafzadeh introduced a low-power driven mapping algorithm at the expense of increase in depth and area [7]. A deficiency of their algorithm is that it fails to take the impact of the fanout number of LUTs on the power consumption into account. Wang, Liu, Lai and Wang presented another algorithm to reduce power by using a "cut enumeration" method to generate a predefined number of possible mapping solutions for the sub-circuit rooted at each node [17]. However, it does not guarantee that if more possible mapping solutions are tested, the final mapping solution would consume less power. In this paper, we propose an efficient low-power driven technology mapping algorithm. The main idea is to compute low-power K-feasible cuts to generate LUTs to cover the given circuit. Experimental results show that our algorithm reduces power consumption as well as area over [7, 17] significantly.

The rest of the paper is organized as follows: In Section 2, we give the problem formulation. In Section 3, we review the power estimation model presented in [7, 17]. In Section 4, we present our technology mapping algorithm. In Section 5, we report the experimental results. In Section 6, we draw the conclusions.

II. Problem Formulation

A general Boolean network N can be represented as a directed acyclic graph (DAG) $N(V, E)$, where V is the set of nodes and E is the set of directed edges. Each node in the DAG represents a logic gate, and a directed edge (u, v) exists only when the output of node u is an input of node v. Let $inputs(v)$ denote the set of fanin nodes of node v. A Boolean network N is *K-bounded* if $|inputs(v)| \leq K$ for every node v in N. Given a K-bounded Boolean network, let N_v denote the subnetwork consisting of node v and all its predecessors. A cut $(X_v, \overline{X_v})$ for node v is a partition of the nodes in N_v such that v is in $\overline{X_v}$. The *node cut-size* is the number of nodes in X_v that are adjacent to some nodes in $\overline{X_v}$. If the cut size is no larger than K, it is called a *K-feasible* cut. If $(X_v, \overline{X_v})$ is a K-feasible cut, we can cover all nodes in $\overline{X_v}$ by a single K-input LUT.

In this paper, we focus on the technology mapping problem for LUT-based FPGAs to minimize power dissipation. The input is a K-bounded Boolean network, the output is a mapping into K-input LUTs with low-power dissipation. The area of the mapped circuit is measured by the number of LUTs used. In the following section, we will review the general approach to estimate the power dissipation for a circuit mapped into a LUT-based FPGA.

III. Power Estimation Model

In this section, we review the power estimation model used in [7, 17], which estimates the power consumption of a mapped

circuit consisting of LUTs only. The majority of power consumption in CMOS circuits is due to the dynamic power dissipation [18]. Dynamic power dissipation P_d occurs because of the switching activity of the circuit, which results in the charging/discharging of the load capacitance.

The *equilibrium probability* of a signal v, denoted as $p(v)$, is defined as the probability that signal v has the value 1. And the *transition density* of a signal v, denoted as $d(v)$, is defined as the number of times that signal v changes its value in unit time.

Let $y = f(x_1, x_2, \ldots, x_n)$ be a Boolean function. The *Boolean difference* of y with respect to x_i, denoted as $\frac{\partial y}{\partial x_i}$, is defined as:

$$\frac{\partial y}{\partial x_i} = y|_{x_i=0} \oplus y|_{x_i=1} \tag{1}$$

Then the transition density at the output of a gate can be calculated as follows:

$$d(y) = \sum_{i=1}^{n} p\left(\frac{\partial y}{\partial x_i}\right) d(x_i) \tag{2}$$

Using this theorem we can compute the transition density of any node in the network when the transition densities at the primary inputs are given. The proof of this theorem can be found in [13].

Once the transition density of every gate has been computed, the power consumption of a given circuit can be calculated as:

$$P_d = \frac{1}{2} \sum_i [\, C_i \cdot V_{dd}^2 \cdot d(i) \,] \tag{3}$$

where C_i is the load capacitance of gate i, V_{dd} is the power supply voltage, and $d(i)$ is the output transition density of gate i.

For LUT-based FPGAs, this formula still holds except that the transitions happen only at the input/output of each LUT. Given a mapping solution, for a LUT rooted at node v and with input nodes $u_1, u_2, \ldots, u_k$, the power dissipation due to the LUT, $P(v, \{u_1, u_2, \ldots, u_k\})$, is given by:

$$P(v, \{u_1, u_2, \ldots, u_k\}) = K_p \cdot C_{out} \cdot d(v) + \sum_{i=1}^{k} K_p \cdot C_{in} \cdot d(u_i) \tag{4}$$

where $K_p = 0.5 \cdot V_{dd}^2$ is a constant, C_{out} is the output capacitance of the LUT, C_{in} is the input capacitance of the LUT.

Now if we are given a Boolean network, we estimate the minimum power dissipation when it is mapped into K-input LUTs as follows. Define a LUT-cover rooted at node v as a LUT that covers all the nodes in a K-feasible cone rooted at v. Let S_v denote the set of all possible LUT-covers rooted at node v and $inputs(L)$ denote the set of nodes that provide inputs to a LUT-cover L in S_v. We use the following recursive formula to estimate the minimum power consumption [1] of a LUT-subnetwork covering N_v:

$$EP(v) = \min_{L \in S_v} \{P(L) + \sum_{u \in inputs(L)} EP(u)\} \tag{5}$$

[1] Note that $EP(v)$ is actually an upper bound on the power consumption of any LUT-subnetwork covering N_v when there are reconvergent fanouts within N_v.

For any PI node u, we assume that $EP(u)$ is equal to zero. For example, for the mapping solution shown in Fig. 1,

$$\begin{aligned} EP(j) &= K_p \cdot C_{out} \cdot d(j) + K_p \cdot C_{in} \cdot [d(a) + d(b) + d(c)] \\ &\quad + EP(a) + EP(b) + EP(c) \\ &= K_p \cdot C_{out} \cdot d(j) + K_p \cdot C_{in} \cdot [d(a) + d(b) + d(c)] \end{aligned}$$

as $EP(a) = EP(b) = EP(c) = 0$, and

$$\begin{aligned} EP(l) &= K_p \cdot C_{out} \cdot d(l) + K_p \cdot C_{in} \cdot [d(j) + d(k)] \\ &\quad + EP(j) + EP(k) \end{aligned}$$

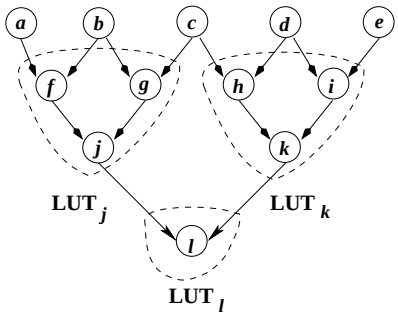

Fig. 1. A network rooted at l. Nodes *a*, *b*, *c*, *d*, and *e* are PI nodes.

IV. Power Minimization Algorithm

In this section, we present our power minimization algorithm called *PowerMinMap* (PMM). The input to the algorithm is a K-bounded gate-level network. If the netlist is not K-bounded then we will first decompose the network to satisfy this property. The equilibrium probabilities and transition densities of the primary inputs are given. The PMM algorithm consists of three phases:

1. Transition density propagation,
2. Computation of $EP(v)$ for every node v,
3. Mapping generation.

In the first phase, we compute the transition density for all the internal nodes as well as PO nodes in a topological order. Note that by doing so, we would have also propagated the equilibrium probabilities.

A. Computation of $EP(v)$

The EP values of all nodes are computed in a topological order starting from the PIs. For each node v, a sequence of *T-bounded K-feasible* cuts is computed based on the EP values of v's ancestors. $EP(v)$ is computed based on the cut that yields the minimum power consumption. A *T-bounded K-feasible* cut is a cut whose input size is no larger than K such that the EP value on any input to the cut is no more than T. This term is further explained in detail below.

A.1 T-Bounded K-Feasible Cut

While generating a LUT rooted at node v, we would like to minimize the sum of the power consumption of the fanins feeding into this LUT. However, it has been proved that the problem of finding a technology mapping solution with minimum power consumption is NP-hard [6]. Hence we propose an efficient method to compute a low-power K-feasible cut by generating a sequence of *T-bounded K-feasible* cuts using incremental network flow computation.

Definition: A T-bounded K-feasible cut $(X_v, \overline{X_v})$ for node v is a K-feasible cut such that the maximum EP value of the nodes in X_v that provide inputs to the nodes in $\overline{X_v}$ is no larger than a threshold T.

Definition: A T-bounded K-feasible cut is a *low-power K-feasible cut* for node v, if it results in the smallest power consumption among the sequence of T-bounded K-feasible cuts computed for v.

For a given non-PI node v, a low-power K-feasible cut is computed as follows: We start with the largest EP value in v's ancestors as a threshold T, and use a network flow based approach to check if there exists a T-bounded K-feasible cut. We convert subnetwork N_v into a node-capacitated flow network by assigning infinite flow capacity to those nodes having EP values larger than the threshold and unit flow capacity to other nodes. Then, we compute a max-flow in the constructed network. If the value of the max-flow computed is no larger than K, we will repeat with the next largest EP value in v's ancestors as the new threshold value, so on and so forth. Each time we find a new K-feasible cut, we compute its total power consumption using Equation (5). We retain the cut that yields the smallest power consumption so far. When the value of the max-flow exceeds K, we terminate and the estimated power for node v, $EP(v)$, is set to the smallest power consumption recorded. This is so because we will not be able to find a K-feasible cut for any smaller threshold.

Illustrative Example:

Fig. 2(a) illustrates how a sequence of T-bounded K-feasible cuts is determined for node v for K=3. Next to each node (except v), the transition density (d) and EP value are shown in d/EP format. For simplicity, we assume that $K_p \cdot C_{out}$ and $K_p \cdot C_{in}$ are equal to 1. The threshold value is first set to 5.2, which is the largest EP value (for node h) of the nodes in $N_v - \{v\}$. The corresponding min-cut computed is Cut I. According to Equation (5), the estimated power consumption for Cut I is:

$$EP_{Cut\,I}(v) = [d(v) + d(g) + d(h)\,] + EP(g) + EP(h) \\ = d(v) + 11.4$$

We store Cut I as the best cut at this point. Then threshold T is lowered to 4, the next highest EP value (for node f) in $N_v - \{v\}$, and Cut II is the min-cut computed. The estimated power for Cut II is:

$$EP_{Cut\,II}(v) = [d(v) + d(e) + d(f) + d(g)\,] \\ +EP(e) + EP(f) + EP(g) = d(v) + 15$$

Since d(v)+15 is larger than d(v)+11.4, we move on to the next smaller EP value as the threshold. The threshold is lowered to 3.5 (for node e) and Cut III is the min-cut computed. Since the cut size is larger than 3, we will terminate. Hence, the mapping induced by Cut I is the most desirable choice for node v. Fig. 2(b) shows the corresponding mapping solution to cover N_v.

A.2 Incremental Network Flow Computation

In this section, we will describe how we may use incremental network flow method to efficiently find the low-power K-feasible cut for a node v.

We first construct the flow network with the largest EP value in N_v as the threshold. Once we have found the first T-bounded K-feasible cut, we have to repeat with the second largest EP value as the threshold. However, there is no need to build a new flow network to compute the new cut. We only need to update the residual network for all node v such that $EP(v)$ is equal to the old threshold value.

We use the same example as in Section A.1. Suppose we want to compute a low-power 3-feasible cut for node v in Fig. 2(a). The EP values of the ancestors of node v in decreasing order are 5.2, 4, 3.5, 3, and 0. First, the threshold is set to 5.2, therefore $\overline{X_v}$ can have any node as its input node. The required flow network is shown in Fig. 3(a). Here we used a standard network transformation technique, known as *node-splitting*, that transforms a node-capacitated network into an edge-capacitated network such that any existing edge cut computation algorithm can be applied. A minimum cut of size 2 is computed by maximum flow computation, the residual network is shown in Fig. 3(b). This cut corresponds to Cut I in Fig. 2(a). Since Cut I is 3-feasible, so we lower the threshold to 4 to compute a new cut $(X_v, \overline{X_v})$ such that no node with EP value greater than 4 can be an input node to $\overline{X_v}$. To compute such a new cut, we may simply update the residual network by changing the flow capacity of node h whose EP value is 5.2 from unit to infinity. If there exists a new augmenting path in the updated residual network, it can be found efficiently.

We associate each node v with two flags: $FS(v)$ equals to

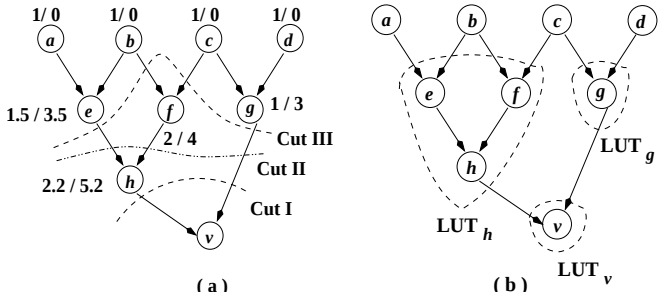

Fig. 2. (a) Selecting a 3-feasible cut for node v (K=3). (b) A mapping solution.

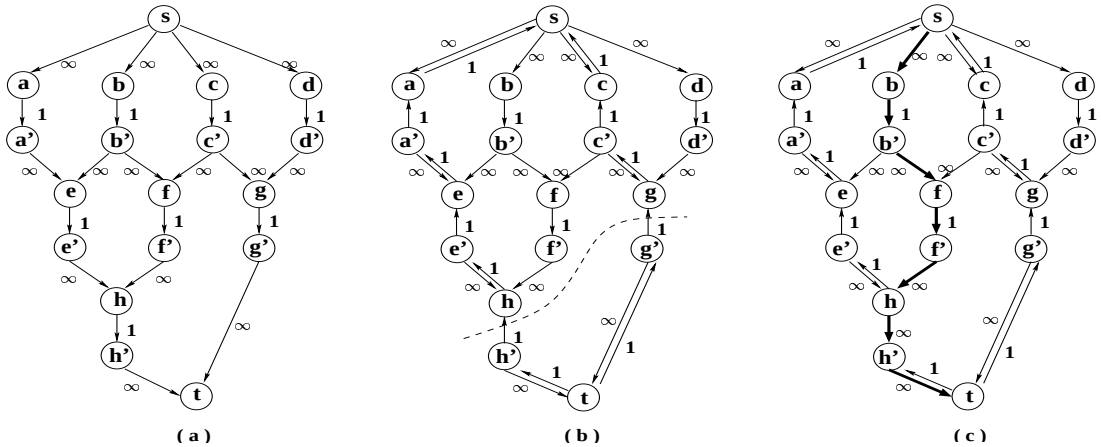

Fig. 3. (a) The flow network for T=5.2 (b) Residual network corresponding to a maximum flow. (c) The updated residual network with a flow augmenting path shown in bold edges.

TRUE indicates that v is reachable from source s and $FT(v)$ equals to TRUE indicates that there exists a path from v to sink t in the residual network. Note that the initial values of $FS(v)$ and $FT(v)$ can be derived simultaneously while traversing the network to compute the min-cut of the original network. When we update the flow capacity of a node v, we also check if its flags have to be updated. For example, $FS(h)$ is TRUE and $FT(h)$ is FALSE in Fig. 3(b). After node h's flow capacity is updated, $FT(h)$ needs to be updated to TRUE as in Fig. 3(c). If $FS(v$ and $FT(v)$ are both equal to TRUE, we know that there exists an augmenting path. In this case, we will augment the flow, traverse the network again to compute a new min-cut and the new FS and FT values for each node. Since we will terminate once the max-flow value exceeds K, we will traverse the network at most K times.

B. Mapping Generation

This phase generates a K-LUT mapping solution of the whole circuit. The mapping solution is generated in a bottom up manner starting from the PO nodes. If a LUT needs to be generated rooted at v, a K-feasible cut is recomputed for node v based on the cost values of its ancestors.

Initially, $cost(u)$ is set to $EP(u)$ for all node u. When we generate a LUT rooted at node v, the power dissipation of subnetwork N_u for all node u that provides input to LUT_v will be counted. So in order to avoid counting the power dissipation of subnetwork N_u again if another LUT is generated that also receives input from node u, we have to reset $cost(u)$ to zero after the first time it is counted. When we compute the low-power K-feasible cut in this phase, we use the dynamically updated cost values.

We also introduce a technique to further improve the quality of the mapping. If a low-power K-feasible cut found by the flow method has a cut size less than K, we will check if it is possible to reduce the sum of the power consumption of the fanins by replacing one node in the node cut set with its fanin such that the cut size still does not exceed K. We call this *cut frontier refinement*. For example, assume Cut 1 shown in Fig. 4 is a 4-feasible cut computed for node v. Since the cut size is only 3, we can increase the cut size to 4. If $cost(l)+cost(m) < cost(i)$, we know that a LUT generated according to Cut 2 consumes less power. Similarly, we also check if choosing the node cut set $\{i, m, n, k\}$ or $\{i, j, n, o\}$ can save more power over Cut 2. We do this recursively until we cannot increase the cut size any more. If there exist cuts that use less power than the K-feasible cut computed, we will replace it with the one that reduces the most power.

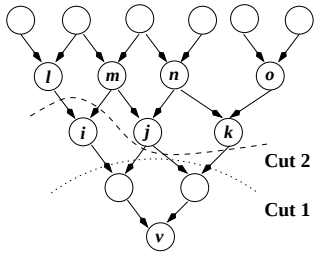

Fig. 4. Cut frontier refinement for N_v (K=4).

The complete algorithm of PowerMinMap is shown in Fig. 5. Line 0 – 3 is the first phase that computes the transition density of each non-PI node in the DAG. Line 4 – 8 is the second phase that computes $EP(v)$, the estimated minimum power consumption of a LUT-subnetwork covering N_v, for every node v. Line 9 – 21 is the last phase where the mapping solution is generated. Once a LUT is generated, we perform cut frontier refinement and update the cost values of its fanins.

Algorithm PowerMinMap
Input: General Boolean network N
Output: A mapping of the network into K**-LUTs**

```
                        /* phase 1 */
0  L ← list of non-PI nodes in topological order from PIs to POs;
1  for each node v ∈ L
2     compute d(v);
3  endfor
                        /* phase 2 */
4  for each node v ∈ L
5     compute a sequence of T-bounded K-feasible cuts for node v;
6     EP(v) ← power consumption of the low-power K-feasible cut;
7     cost(v) ← EP(v);
8  endfor
                        /* phase 3 */
9  Q ← a queue of all POs;
10 while Q ≠ ∅
11    v ← dequeue (Q);
12    compute a low-power K-feasible cut for node v;
13    perform cut frontier refinement;
14    generate a LUT rooted at v;
15    for each node u ∈ inputs(LUT_v)
16       if node u ∉ Q and node u ∉ PIs
17          enqueue (u, Q);
18          cost(u) ← 0;
19       endif
20    endfor
21 endwhile
End-algorithm
```

Fig. 5. Pseudocode of PowerMinMap Algorithm

C. Computational Complexity

Theorem: Given a general Boolean network *N*, PowerMinMap generates a low-power mapping solution in $O(Kmn + n^2 \log n)$ time, where n and m are the number of nodes and edges in N, respectively.

Proof: In phase one of PMM, it takes $O(n)$ time to compute the output transition density for each non-PI node v. In phase two, it takes $O(n' \log n')$ time to sort the EP values of the nodes in $N_v - \{v\}$, where n' is the number of nodes in N_v. When we compute a set of T-bounded K-feasible cuts to find the low-power K-feasible cut in N_v, we need to traverse N_v at most K times. Traversing the network once takes $O(m' + n')$ time, where m' is the number of edges in N_v. So it takes a total of $O(n' \log n') + O(Km' + Kn') = O(Km' + n' \log n')$ time to compute a low-power K-feasible cut in N_v. As m' and n' are bounded by m and n, phase two takes $O(n \cdot (Km + n \log n)) = O(Kmn + n^2 \log n)$ time to compute $EP(v)$ and $cost(v)$ for all node v. Phase three differs from phase two in that low-power K-feasible cut is recomputed for L nodes, where L is the number of LUTs generated in the mapping solution ($L \leq n$). And cut frontier refinement is also performed which takes $O(K)$ time. So phase three also takes $O(Kmn + n^2 \log n)$ time. Thus, the overall runtime for PMM is $O(n) + O(n^2 \log n + Kmn) + O(n^2 \log n + Kmn)$, which is $O(n^2 \log n + Kmn)$. If m and n are of the same order and K is fixed, the total time is $O(n^2 \log n)$.

V. Experimental Results

We have implemented the PMM algorithm using C language and experimented with the same set of MCNC benchmark circuits as reported in [7, 17]. Given a general Boolean network, we first optimize it with the optimization script "rugged" within SIS. Then we decompose the circuit into 2-bounded network using the "DMIG" command.

We assume that $V_{dd} = 5V$ and all PI nodes have equilibrium probability p = 0.5, and transition density d = 10,000. The capacitances C_{in} and C_{out} are set to 10 pF each. And we map the circuits into 5-input LUTs. When we estimate the power consumption of a mapping solution, the external load capacitance of PO nodes (which is not known *a priori*) is ignored since the power consumption due to a given external load is independent of the mapping. Note that the above assumptions have been used in [7, 17]. The results of PMM algorithm are shown in Table I. For comparison, we quote the best result reported for each circuit in [17]. We also include the results from [7].[2] On the average, our algorithm reduces the power consumption by 18.5% and 12.2% compared with [7] and [17], respectively. Besides, it uses 9.5% and 10.6% fewer LUTs than [7] and [17], respectively. Note that in two cases (rd84 and vg2), PMM used more LUTs than [7], but it still achieved power saving. This shows that reducing the number of LUTs does not always reduce the power consumption as well.

Our algorithm is able to achieve greater power saving since it can predict the impact of generating a single LUT on the mapping of the LUTs generated afterwards. Hence, we can choose the mapping that yields smaller power consumption. Besides, we notice that LUTs already generated also affect the mapping of the subnetwork which has not been covered yet. By dynamically updating the cost value of each node, we have up-to-date power estimation information and therefore, we can implement the rest of the circuit with better LUT covering.

VI. Conclusions

We studied the technology mapping problem to minimize power dissipation for LUT-based FPGAs. We presented an algorithm that minimizes power as well as area. We used an efficient incremental network flow computation method to compute low-power K-feasible cuts that optimizes the total power consumption of the generated K-LUTs. Our algorithm estimates the power consumption for a set of possible choices and choose the one that yields the best solution. Experimental results show that our algorithm achieved both power and area savings over two previous algorithms targeting at minimizing power.

[2] We cannot compare the delays because delay information is not reported in [7, 17].

TABLE I
COMPARISON OF POWERMINMAP WITH [17] AND [7] (PWR: mW)

CKT	**Wang *et al.*[17]**		**Farrahi *et al.*[7]**		**PowerMinMap**		**Percentage Savings (%)**			
							Vs. [17]		Vs. [7]	
	LUTs	PWR	LUTs	PWR	LUTs	PWR	LUTs	PWR	LUTs	PWR
5xp1	26	188	25	182	23	168	11.5	10.6	7.7	7.8
9sym	87	553	62	365	60	340	31	38.5	3.2	6.8
9symml	62	446	58	376	56	349	9.7	21.7	3.4	7.2
c499	98	1061	91	1076	74	983	24.5	7.4	18.7	8.6
c880	116	746	111	1060	106	718	8.6	3.8	4.5	32.2
alu2	128	874	146	836	124	815	3.1	6.8	15.1	2.5
apex6	192	1110	237	1404	183	1012	4.7	8.8	22.8	27.9
apex7	68	379	69	450	65	349	4.4	7.9	5.8	24.0
count	31	159	31	227	31	159	0	0	0	29.9
duke2	152	538	190	478	145	443	4.6	17.6	23.7	7.3
misex1	13	97	16	106	12	92	2.3	5.2	25	13.2
rd84	33	261	27	344	32	243	3	6.9	-18.5	29.4
rot	234	1306	238	1749	224	1203	4.3	7.9	5.9	31.2
vg2	32	182	25	60	26	158	18.7	13.2	-4	1.3
z4ml	7	56	5	80	5	41	28.6	26.8	28.6	48.8
Average							10.6	12.2	9.5	18.5

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows - Theory, Algorithms, And Applications*, Prentice Hall, 1993.

[2] C. Bhat, and N. N. Chiplunkar, "Routability-driven technology mapping for lookup table-based FPGAs", in *12th International Conference on VLSI Design*, pp. 390-393, Jan. 1999.

[3] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field Programmable Gate Arrays*, Kluwer Academic Publishers, 1995.

[4] K.-C. Chen, J. Cong, Y. Ding, A. Kahng, and P. Trajmar, "DAG-Map: graph-based FPGA technology mapping for delay optimization", *IEEE Design and Test of Computers*, pp. 7-20, 1992.

[5] J. Cong, and Y. Ding, "An optimal technology mapping algorithm for delay optimization in look-up table based FPGA designs", in *International Conference on Computer Aided Design*, pp. 213-218, Nov. 1992.

[6] A. H. Farrahi, and M. Sarrafzadeh, "Complexity of the lookup-table minimization problem for FPGA technology mapping", in *IEEE Transactions on Computer-Aided Design*, vol. 13(11), pp. 1319-1332, Nov. 1994.

[7] A. H. Farrahi, and M. Sarrafzadeh, "FPGA technology mapping for power minimization", in *4th International Workshop on Field Programmable Logic and Applications*, pp. 66-77, September 1994.

[8] R. J. Francis, J. Rose, and K. Chung, "Chortle: A technology mapping for lookup table-based field programmable gate arrays", in *27th ACM/IEEE Design Automation Conference*, pp. 613-619, June 1990.

[9] R. J. Francis, J. Rose, and Z. Vranesic, "Chortle-crf: fast technology mapping for lookup table-based FPGAs", in *28th ACM/IEEE Design Automation Conference*. pp. 227-233, June 1991.

[10] K. Karplus, "Xmap: A technology mapper table-lookup field-Programmable gate arrays", in *28th ACM/IEEE Design Automation Conference*, pp. 240-243, June 1991.

[11] Y.-T. Lai, M. Pedram, and S. Sastry, "BDD based decomposition of logic functions with application to FPGA synthesis", in *30th ACM/IEEE Design Automation Conference*, pp. 230-235, June 1993.

[12] R. Murgai, N. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Improved logic synthesis algorithms for table look up architectures", in *International Conference on Computer Aided Design*, pp. 564-567, Nov. 1991.

[13] F. Najm, "Transition Density: A new measure of activity in digital circuits", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pp. 310-323, Feb. 1993.

[14] P. Sawkar, and D. E. Thomas, "Technology for table-look-up based field programmable gate arrays", in *ACM/SIGDA Workshop on Field Programmable Gate Arrays*, pp. 83-88, Feb. 1992.

[15] M. Schlag, J. Kong, and P. K. Chan, "Routability-driven technology mapping for lookup table-based FPGAs", in *IEEE International Conference on Computer Design*, pp. 86-90, October 1992.

[16] N. Togawa, M. Sato, and T. Ohtsuki, "Maple: A simultaneous technology mapping, placement and global routing algorithm for field-programmable gate arrays", in *International Conference on Computer Aided Design*, pp. 155-163, 1994.

[17] Z.-H. Wang, E.-C. Liu, J. Lai, and T.-C. Wang, "Power minimization in LUT-based FPGA technology mapping", in *ASP-DAC*, pp. 635-640, 2001.

[18] N. Weste, and K. Eshraghian, *Principles of CMOS VLSI Design: A System Perspective*, Addison-Wesley Reading, 1993.

[19] J. Cong, and Y. Ding, "On area/depth trade-off in LUT-based FPGA technology mapping", *IEEE Trans. on VLSI Systems*, vol. 2, no. 2, pp. 137-148, June 1994.

[20] J. Cong, and Y.Y. Hwang, "Simultaneous depth and area minimization in LUT-based FPGA mapping", in *Proc. Int. Symp. on Field Programmable Gate Arrays*, pp. 68-74, 1995.