A Technology Mapping Algorithm for Heterogeneous FPGAs

Chi-Chou Kao and Yen-Tai Lai
Department of Electrical Engineering
National Cheng Kung University
Tainan, Taiwan

Abstract - In this paper, a technology mapping algorithm is proposed for heterogeneous FPGAs. The technology mapping problem is first formulated as a flow network problem. Then, an algorithm based on the min-cost max-flow algorithm is presented to select a proper set of feasible LUTs for various objectives. The objective, the total area composed of LUTs and routing area, are discussed in the paper. This algorithm has been tested on the MCNC benchmark circuits. Compared with other existing LUT-based FPGA mapping algorithms, the algorithm produces better characteristics.

I Introduction

In a traditional lookup table (LUT)-based FPGA device, the configurable logic blocks (CLBs) are composed of *k*-input LUTs whose input number is constant. To maximize device utilization, *heterogeneous* FPGAs provide an array of homogeneous LUTs of different sizes or an array of physically heterogeneous LUTs. For example, the XC4000 [1] and ORCA2C [2] series FPGAs can be configured to have heterogeneous LUTs. The technology mapping problem for LUT-based FPGAs involves producing an equivalent circuit for a given circuit using only gates that can be implemented with LUTs.

This paper addresses the technology mapping problems for heterogeneous FPGA designs. There are several homogeneous FPGA technology-mapping algorithms for minimum layout size. However, most of these algorithms are unable to deal with heterogeneous FPGAs. A recent work [9] has shown that the area minimization-mapping problem for a tree network can be solved optimally in $O(n^3)$. However, this algorithm is significantly limited because the optimality holds only for the tree. The HeteroMap algorithm [10] was presented to reduce the number of LUTs for heterogeneous LUT-based FPGAs. An algorithm is proposed in this paper. The proposed algorithm can be configured for various objectives, the minimum number of LUTs, the total area, or others. To minimize the total area of a FPGA, two parts: 1) The LUT area and 2) the routing area are considered simultaneously.

The technology mapping problem is formulated first as a flow network problem. An algorithm based on the min-cost max-flow algorithm is then presented to select a proper set of LUTs from the set of feasible LUTs. An enumerating algorithm to generate all feasible LUTs is also presented. We implemented this algorithm and compared the empirical results with other LUT-based FPGA mapping algorithms. The results demonstrate the efficiency of this algorithm.

The remainder of this paper is organized as follows. The terminology and a graph-based formulation of the problem are described in Section 2. An algorithm for solving the problem is shown in Section 3. Section 4 gives an algorithm to generate the set of feasible cones. The objective, the total area, are discussed in Section 5. Experimental results are shown in Section 6. Our concluding remarks are presented in Section 7.

II. Formulation of the Mapping Problem

An FPGA technology mapping problem can be formulated as a graph based problem. 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. Let v and u be two vertices of V_g. 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. *Input*(V_s) is defined as representing the set of fan-in signals of V_s. Similarly, the *Output*(V_s) represents the set of fan-out signals. 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.

Assume that C_v is the subgraph induced by V_s. The subgraph $C_v = (V_v, E_v)$ is said to be a *cone* if vertex $v \in V_v$ exists such that for every vertex $u \in V_v$ 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 *k-feasible* if $|Output(V_v)| = 1$ and $|Input(V_v)| \leqq k$. A vertex u is said to be *covered* with C_v if $u \in V_v$. Because C_v is the induced subgraph of V_v, the set of fan-in signals of C_v is exactly equal to the $|Input(V_v)|$. For convenience, *Input*(V_v) and *Input*(C_v) are interchangeable in the remainder of this paper.

We assumed that a general LUT-based heterogeneous FPGA consists of LUTs of n types. Each LUT of one type has k inputs, $k \in \{k_1, k_2, \ldots, k_n\}$. Homogeneous FPGAs can be viewed as a special kind of heterogeneous FPGA with one and only one type of LUT. The technology mapping problem can be described as: Given a 2-bounded Boolean network, according to the objectives, find a set of feasible cones such that the union of all feasible cones includes all vertices.

Finding the maximum flow with minimum cost in a flow

network can solve this problem. Given a directed acyclic graph, $G = (V_g \cup V_{io}, E)$, let V_c be the set of feasible cones in G. According to G, a flow network $G_{asn} = (V_g \cup V_c \cup \{s,t\}, E_{bp} \cup E_{st})$ is constructed in which:

1) V_g is the set of vertices representing gates,
2) a vertex in V_c represents a feasible cone,
3) s and t represent the source and the sink respectively,
4) there is an edge $e_{ij} \in E_{bp}$ directed from a vertex $i \in V_g$ to a vertex $j \in V_c$ if i is a vertex in the k-feasible cone associated with j and the capacity $cap(e_{ij})$=1,
5) there is an edge $e_{si} \in E_{st}$ directed from s to every vertex $i \in V_g$, and the capacity $cap(e_{si})$=1, and
6) there is an edge $e_{jt} \in E_{st}$ directed from each vertex $j \in V_c$ to t, and its capacity, $cap(e_{jt})$, is equal to the number of vertices in the cone $C_j = (V_j, E_j)$ associated with j.

Figure 1.*b* illustrates a flow network constructed from the DAG shown in Figure 1.*a*. It is assumed that a LUT in the heterogeneous FPGA has k-inputs, $2 \le k \le 4$. In G_{asn}, an edge is said to be *saturated* if the flow through it is equal to its capacity. Since a V_c vertex has only one out-edge, for convenience, a V_c vertex is also said to be *saturated* if its out edge is saturated. On the other hand, if the flow through a V_c vertex is zero, the V_c vertex is said to be *empty*.

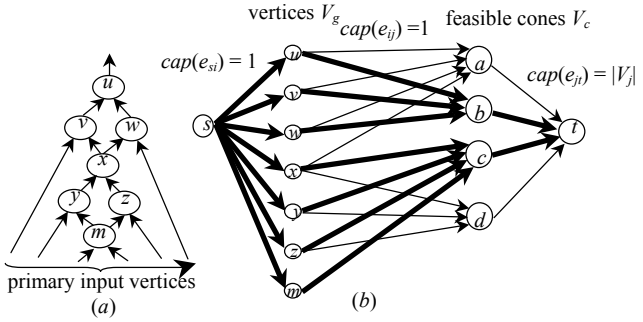

Fig. 1. *a*) A given DAG; *b*) The flow network constructed from (*a*).

***Theorem* 1:** A maximum flow in G_{asn} is equivalent to the solution of a technology mapping problem if the total flow is equal to $|V_g|$ and every V_c vertex is either saturated or empty.

***Proof*:** Since there is one and only one edge directed from s to each V_g vertex and its capacity is one, the upper bound of the total flow is the total number of vertices in V_g. If the maximum flow is equal to the total number of vertices in V_g, the mapping includes all of the vertices in V_g. If we select the set of feasible cones corresponding to the vertices, V_c is saturated. By the construction rules, every gate is included in one and only one selected cone.■

Consider the flow shown in Figure 1.*b*. The bold lines from a V_c vertex to t are saturated edges; the others are zero. Since the total flow is equal to $|V_g|$, by Theorem 1, the set of cones induced by $\{u, v, w\}$ and $\{x, y, z, m\}$ can be selected to be an optimal mapping solution. There are several ways to generate a set of feasible cones [4]. If the set of feasible cones is not rich enough, it may not be possible to obtain a feasible solution. An algorithm to generate the set of feasible cones is presented in section 4.

III. An Algorithm for Finding the Min-cost Max-flow as Mapping Solution

The mapping solution is not unique. Given an objective, we can define a cost function to assign a weight on each edge in G_{asn} and find the min-cost max-flow as the solution. There are several optimization objectives in the technology mapping process. Two objectives, the minimum number of LUTs and the total area, are discussed in Section 5.*A* and *B*. In this section, it is assumed the weight of cost on every edge is given.

There is an algorithm [13-14] that can find the maximum flow with minimum cost. However, a solution found by this algorithm does not ensure the flow through the sink edges is either saturated or zero. The bold lines in Figure 2 illustrate the min-cost max-flow network. It is seen that the flow through the vertex a is neither saturated nor zero.

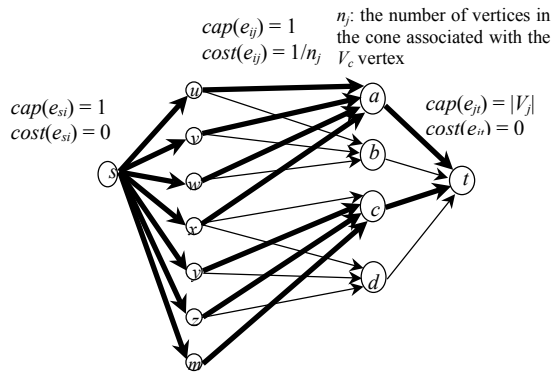

Fig. 2. A min-cost max-flow.

The flow path in G_{asn} found by the min-cost max-flow algorithm must be changed such that the V_c vertex is either saturated or zero. When the flow path is changed, an algorithm which increases the minimum cost bound is needed. It has been proven that this problem is a NP-complete problem [13]. We will construct a max flow solution by a greedy algorithm. The strategy of this algorithm is as follows:

1) Find the minimum-cost maximum-flow in the G_{asn}.
2) The cones associated with the saturated V_c vertices are selected to be in the mapping solution.
3) A V_c vertex, j, is called a *candidate vertex* if j is non-saturated and the cone corresponding to j includes no V_g vertices covered by selected cones. The total cost increases if we change the path of flow through other non-saturated V_c vertices such that j becomes saturated. The increase of total cost to force j saturated is denoted $\Delta C(j)$. For every candidate vertex, j, calculate $\Delta C(j)$.
4) According to the calculation in step 3, find the candidate vertex, j, whose $\Delta C(j)$ is minimum and change the path of flow to make j be saturated.
5) Select the cone corresponding to j to be in the mapping solution. If there exist candidate vertices, go to step 3.

6) If every V_g vertex is covered, a mapping solution is obtained. Otherwise, no mapping solutions exist.

The min-cost max-flow algorithm [13-14] first finds a maximum flow in G_{asn}. Then it constructs an auxiliary graph and iteratively reduces the total cost by finding directed cycles with negative costs in the auxiliary graph.

We need an approach to calculating $\Delta C(j)$ of a candidate vertex j. Let $C_j=(V_j, E_j)$ be the cone corresponding to j and V_s be a subset of V_j such that in G_{asn} the flow coming from a V_s vertex goes into j. Let $\overline{V_s} = V_j - V_s$ and $x \in \overline{V_s}$. Assume in G_{asn} e_{xj} is the edge directed from x to j and the flow from x to a vertex y passes along the edge e_{xy}. If the flow passing along e_{xy} is changed to pass along e_{xj}, the increase of total cost is equal to cost(e_{xj})–cost(e_{xy}). Accordingly,

$$\Delta C(j) = \sum_{x \in \overline{V_s}} (\cos t(e_{xj}) - \cos t(e_{xy}))$$

where e_{xj} is the edge directed from x to j and e_{xy} is the edge passed by the flow from x to y.

Consider the example in Figure 2. It is seen that the vertex c is saturated. Hence we select the cone induced by {x, y, z, m} which is associated with c. The vertex b is a candidate vertex. On the other hand, d is not a candidate vertex because the cone associated with d includes {x, y, z} which are covered V_g vertices. To make b saturated, the flow along the three edges, <u, a>, <v, a>, and <w, a>, must be redirected to pass through b. To change the flow in each of these three edges, the cost increases 1/3−1/4=1/12. Hence, $\Delta C(b)$ =1/4. Since b becomes saturated, the cone induced by {u, v, w} is selected to cover the other vertices in the mapping solution.

IV An Algorithm to Enumerate All Feasible Cones

This section describes an algorithm to enumerate all feasible cones. Given a DAG, $G = (V_g \cup V_{io}, E)$, we generate the feasible cones tipped at each vertex in V_g. Let $v \in V_g$ be a vertex and $CN(v)$ denote the set of cones C_v tipped at v and $|Input\ (C_v)| \le k$. Let $Fin\ (v)$ be the set of fan-in vertices of v and $Pcn(v) = \bigcup_{v' \in Fin(v)} CN(v')$. Since a cone must be connected, a cone in $CN\ (v)$ must be the union of {v} and a subset of $Pcn(v)$. Therefore, we only need to inspect the cones in every $CN\ (v')$, $v' \in Fin\ (v)$.

Accordingly, starting from every primary output vertex, all feasible cones can be found in an order of post-order traversal of G. Let V_v be the union of {v} and a subset of $Pcn(v)$. The subgraph induced by V_v is a k-feasible cone if and only if $|Input\ (V_v)| \le k$ and $|Output(V_v)| = 1$. To find $CN\ (v)$, we must check the union of {v} and every subset of $Pcn(v)$.

The union of a subset of $Pcn(v)$ can be equivalent to the union of another subset of $Pcn(v)$. Assume C_x and C_y are two cones in $CN\ (v')$ and C_z is the union C_x and C_y. Every cone in $CN\ (v')$ includes v'. If C_z is a feasible cone, C_z must include v' and $C_z \in CN\ (v')$. Otherwise, C_z is not a feasible cone and $|Input\ (C_z)| > k$. In this case, $|Input(\{v\} \cup C_z)| > k$ and $\{v\} \cup C_z$ must not be a feasible cone. In other words, the union operation in $CN\ (v')$ is closure. Let a *separation selection* of $Pcn(v)$ be a subset of $Pcn(v)$ in which every member is selected separately from a $CN\ (v') \in Pcn(v)$, $v' \in Fin\ (v)$. Therefore to find $CN(v')$, we can just check whether $|Input(V_{ss})| \le k$ and $|Output(V_{ss})| = 1$, where V_{ss} is the union of {v} and every separation selection of $Pcn(v)$.

V The Optimization Objective

The total area is the most important objective in the technology mapping process.

A large LUT must use a large area. A mapping with the minimum number of cones may not lead to the minimum total area because the size of the LUT grows exponentially proportional to the number of cone inputs. It is better to optimize the total area.

Assume that C_j is a cone associated with a V_c vertex and A_j is the total area needed for using a logic block to implement C_j. Similar to the cost function defined to minimize the total number of LUTs, if the cost of the edges between a V_g vertex and the V_c vertex is set to $cost(e_{ij}) = A_j/n_j$ and the cost of the other edges are zero, the total area can be minimized by finding the min-cost max-flow in G_{asn} where n_j is the number vertices included in C_j.

Recall that the FPGA area includes the area for the LUTs and the area for routing. Let A_b be the area for a logic block, and A_c be the area needed for interconnection if a cone C_j is selected. Then $A_j = A_b + A_c$. An approach for calculating A_b, and a method for estimating A_c are needed.

To calculate A_b the area for a LUT-based logic block, we used the logic block model shown in the literature 3 [1]. Let BA be the bit area required to store a static RAM bit and FA be the fixed area required to implement the D flip-flop and all of the other associated circuitry. The area for a logic block, A_b, is then:

$$A_b = BA \times 2^k + FA. \qquad (1)$$

To estimate the routing area, we used the model proposed in the literature [3]. We can consider the needed routing area to be the space taken by the routing tracks on two of the four sides of the logic block, as shown in Figure 3.

Let N_p be the number of pins. Assume that the pitch of a routing track is approximated as the square root of the area required by a bit. The dimension of a channel is then $N_p \times \sqrt{BA}$. The area for interconnection is:

$$A_c = (N_p)^2 \times BA + 2\sqrt{BA \times 2^k + FA} \times N_p \times \sqrt{BA}. \qquad (2)$$

According to (1) and (2), the total area of a logic block is:

$$A_j = \{BA \times 2^k + FA\} + \{(N_p)^2 \times BA + 2\sqrt{BA \times 2^k + FA} \times N_p \times \sqrt{BA}\}. \qquad (3)$$

According to the experimental results shown in [3], N_p must be at least k+1 and proportional to the total number pins of CLBs.

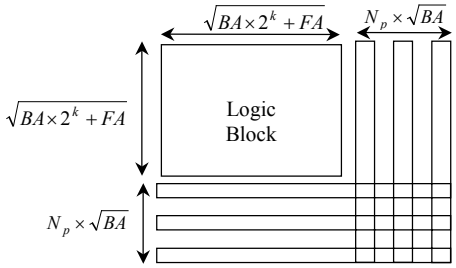

Fig. 3. A model for estimating the routing area.

IV Experimental Results

We used C language to implement the proposed algorithm on a SUN ULTRA SPARC workstation and tested several circuits from the MCNC logic synthesis benchmark set. To produce a more accurate area analysis for heterogeneous FPGA, testing was accomplished using FlowMap [5], HeteroMap [10], and the proposed algorithm on XC4000 series FPGAs that can implement circuits with 4-LUTs and 5-LUTs. Using 1.25μm CMOS technology, *BA* was estimated about 400μm^2 and *FA* was 5100μm^2. Therefore, the total area of a 4-LUT was estimated as $\{400\times2^4+5100\}+\{(5)^2\times400+2\sqrt{400\times2^4+5100}\times5\times\sqrt{400}\}$ = 42948μm. The results are shown in Table 1. It confirms effectiveness of the proposed algorithm in terms of the area.

Circuits	Proposed algorithm			FlowMap			HeteroMap		
	# CLB	Total Area (*mm*)	CPU (s)	# CLB	Total Area (*mm*)	CPU (s)	# CLB	Total Area (*mm*)	CPU (s)
5xp1	12	3095	5.6	20	5159	0.2	16	4127	0.2
9sym	44	11349	11.0	45	11607	0.3	48	12381	0.6
9symml	42	10833	15.4	47	12123	0.3	44	11349	0.6
C880	90	23214	55.3	183	47202	2.3	151	38948	13.3
alu2	77	19861	86.6	128	33016	1.5	114	29404	13.8
count	31	7996	2.7	59	15218	0.4	51	13155	0.6
duke2	96	24762	27.5	140	36111	1.2	127	32758	2.3
rd84	25	6448	20.3	34	8770	0.4	31	7996	0.9
vg2	14	3611	1.0	31	7996	0.2	27	6964	0.3
Total	431	111170	225.4	687	177201	6.8	609	157082	32.6
Comparison	1		1	+59%		-97%	+41%		-86%

Table I. Comparison of Area and CPU Time on XC4000 series FPGAs

Circuits	Proposed algorithm			FlowMap			HeteroMap		
	Total Area (*mm*)	Block Area (*mm*)	Routing Area (*mm*)	Total Area (*mm*)	Block Area (*mm*)	Routing Area (*mm*)	Total Area (*mm*)	Block Area (*mm*)	Routing Area (*mm*)
9sym	2416	702	1714	3093	932	2161	3216	924	2292
9symml	2628	772	1856	2967	888	2079	3171	926	2245
apex4	33720	9308	24413	51712	15380	36332	48881	13367	35514
alu2	7814	2195	5619	9722	2859	6862	9446	2632	6814
des	37338	11157	26181	45448	13676	31772	46916	13722	33194
rd84	1487	416	1071	2596	752	1844	2341	635	1705
Total	85404	24550	60855	115537	34487	81050	113970	32206	81764
Comparison	1	1	1	+35%	+40%	+33%	+33%	+31%	+34%

Table II. Comparison of Area on a Give Heterogeneous FPGA

A heterogeneous FPGA can consist of three or more types of LUTs. We select a heterogeneous FPGA architecture that consisted of three types of LUTs with input sizes of four, five, and six respectively. Table 2 shows the comparison results. Compared with FlowMap and HeteroMap, the proposed algorithm reduces 35% and 33% of the mapping area.

References

[1] Xilinx: "The programmable logic data book," Xilinx Inc., San Jose, CA, 1997.

[2] Lucent Technologies: "ORCA OR2C-A/OR2T-A series FPGAs data sheet," Lucent Technologies Inc., Allentown, PA, 1996.

[3] Rose, J. S., Francis, R. J., Lewis, D., and Chow, P.: "Architecture of programmable gate array: The effect of logic block functionality on area efficiency," IEEE Journal of Solid State Circuits, Vol. 25, No 5, Oct. 1990, pp. 1217-1225.

[4] Schlag, M., Kong, J., and Chan, P. K.: "Routability-driven technology mapping for lookup table-based FPGAs," IEEE Trans. on Computer-Aided Design of Integrated Circuits and System pp.13-26 Vol.13 No. 1, Jan. 1994.

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

[6] Ahuja, R., Thomas, K., Magnanti, L., and Orlin, J. B.: "Network flows theory, algorithms, and applications," Prentice-Hall International Editions, 1993.

[7] Evans, J. R., and Minieka, E.: "Optimization algorithms for networks and graphs," Marcel Dekker, INC., 1992.

[8] He, J., and Rose, J.: "Technology mapping for heterogeneous FPGAs," presented at the ACM Int. Workshop FPGA, Feb. 1994.

[9] Korupolu, M. R., Lee, K. K., and Wong, D. F.: "Exact tree-based FPGA technology mapping for logic blocks with independent LUT," Proc, 35th ACM/IEEE Design Automation Conference, pp. 708-711, June 1998.

[10] Cong, J., and Xu, S.: "Delay-optimal technology mapping for FPGAs with heterogeneous LUT," UCLA Computer Science Dept. Tech. Report CSD-TR980015, 1998.

[11] 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.

[12] 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.

[13] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," *New York, W. H. Freeman*, 1979.