AN OPTIMAL PERFORMANCE-DRIVEN TECHNOLOGY MAPPING ALGORITHM FOR LUT-BASED FPGAS UNDER ARBITRARY NET-DELAY MODELS

Jason Cong and Yuzheng Ding
Department of Computer Science
University of California, Los Angeles, CA 90024, U.S.A.

Tong Gao
Department of Computer Science
University of Illinois, Urbana Champaign, IL 61801, U.S.A.

Kuang-Chien Chen
Fujitsu America, Inc.
3055 Orchard Drive, San Jose, CA 95134, U.S.A.

ABSTRACT

The field programmable gate-array (FPGA) has become an important technology in VLSI ASIC designs. Most existing performance-driven technology mapping algorithms for Lookup-table (LUT) based FPGA designs are based on unit delay model. In this paper we present an efficient algorithm which finds an *optimal* technology mapping solution with minimum delay under *arbitrary* net delay models for LUT-based FPGA designs. The key step of this algorithm is the computation of a *minimum height K-feasible cut* in a Boolean network with arbitrary net delay, which is carried out through efficient flow computation. The capability of dealing with arbitrary net delay models enables our algorithm to be used not only as a FPGA technology mapper but also as an evaluation tool for delay model studies. We have implemented the algorithm and tested it on a number of MCNC benchmark examples.

1. INTRODUCTION

The field programmable gate array (FPGA) has become an important technology for VLSI ASIC designs in recent years due to the short design cycle and low manufacturing cost. An FPGA architecture consists of the programmable logic blocks, programmable interconnections, and programmable I/O pads. The LUT-based FPGA is a popular architecture used by several FPGA manufacturers, including Xilinx and AT&T [6, 16]. In LUT-based FPGA, the basic programmable logic block is a K-input lookup table (K-LUT) which can implement any Boolean function of up to K variables.

The technology mapping problem in LUT-based FPGA designs is to transform a general Boolean network (obtained by technology independent synthesis) into a functionally equivalent network of K-LUTs. Several optimization objectives for the technology mapping problem have been proposed, including minimizing the number of K-LUTs in the mapping solution [5, 7, 8, 10, 11, 15], minimizing the delay of the mapping solution [2, 3, 4, 9], and maximizing the routability of the mapping solution [1, 13].

This paper studies the problem of performance-driven technology mapping for LUT-based FPGAs, i.e. the technology mapping with the objective of minimizing the delay. Previous mapping algorithms for delay minimization include MIS-pga-delay [9], Chortle-d [4], DAG-Map [2], and FlowMap [3], etc.. MIS-pga-delay combines Boolean synthesis with technology mapping and considers the number of levels of the mapping solution as well as the routing delay in its optimization procedure. Chortle-d uses bin-packing heuristic and several post-processing operations to minimize the number of levels in the mapping solution. DAG-Map minimizes the number of levels of the mapping solution based on Lawler's labeling algorithm. None of these algorithms guarantees the optimality of their mapping results. FlowMap for the first time efficiently solves the problem of optimal mapping for delay minimization for general networks based on efficient flow computation. In FlowMap, the delay is also measured by the number of levels in the mapping solution, as in Chortle-d and DAG-Map.

The delay model that assumes uniform delay for each level of a network is called the *unit delay model*. Under the unit delay model, minimizing delay is equivalent to minimizing the number of levels. In reality, although it is usually true that the delay of each LUT is the same, the interconnection delay between each pair of LUTs may vary considerably. Therefore, more accurate delay models are needed to allow variable delay values for different nets.

There are two important reasons to develop an optimal mapping algorithm that can handle arbitrary net delay models. First, such an algorithm will be more effective in generating good mapping solutions since it is not restricted to any specific delay model. Second, such an algorithm can be used as an effective tool to evaluate various delay models. Due to the lack of such an algorithm, previous study of delay models was sensitive to the choice of the heuristic mapping algorithms. Since a heuristic algorithm may be biased towards certain delay models, it may not reveal the true nature of a delay model. With an algorithm that produces optimal solution under *any* delay model, The delay models can be evaluated accurately based on the optimal mapping results.

In this paper we generalize the idea in FlowMap to develop an efficient algorithm that guarantees delay-optimal mapping solution for general networks under arbitrary net delay model.

By efficiently computing a *minimum height K-feasible cut* for each node in the network, we are able to compute an optimal mapping for each node, hence obtain the optimal mapping solution for the entire network by dynamic programming. We have implemented our algorithm and tested it on a set of MCNC benchmark circuits under a simple non-unit delay model.

The remainder of this paper is organized as follows. Section 2 gives a precise problem formulation and some preliminaries. Section 3 presents our depth optimal technology mapping algorithm. Experimental results are presented in Section 4. Section 5 concludes the paper.

2. PROBLEM FORMULATION & PRELIMINARIES

A general Boolean network can be represented as a directed acyclic graph (DAG) where each node represents a logic gate, and a directed edge (i, j) exists if the output of gate i is an input of gate j. A primary input (PI) node has no incoming edge and a primary output (PO) node has no outgoing edge. We use $input(v)$ to denote the set of nodes which are fanins of gate v. Given a subgraph H of the Boolean network, $input(H)$ denotes the set of *distinct* nodes outside H which supply inputs to the gates in H. For a node v in the network, a *K-feasible cone at v*, denoted C_v, is a subgraph consisting of v and its predecessors such that any path connecting a node in C_v and v lies entirely in C_v, and $|input(C_v)| \le K$. A Boolean network is *K-bounded* if $|input(v)| \le K$ for each node v.

We assume that each programmable logic block in an FPGA is a K-input lookup-table (K-LUT) that can implement any K-input Boolean function. Thus, each K-LUT can implement any K-feasible cone of a Boolean network. The technology mapping problem for K-LUT based FPGAs is then to cover a given Boolean network with K-feasible cones[1]. Fig. 1 shows an example of mapping a Boolean network into a 3-LUT network. Note that we allow these cones to overlap, which means that the nodes in the overlapped region can be duplicated when generating K-LUTs. In fact, our algorithm is capable of duplicating nodes automatically when necessary, in order to achieve delay optimization. A technology mapping solution S is a DAG where each node is a K-feasible cone (equivalently, a K-LUT) and the edge (C_u, C_v) exists if u is in $input(C_v)$.

The delay of an FPGA circuit is determined by two factors: the delay in K-LUTs and the delay in the interconnection paths. Each K-LUT contributes a constant delay (the access time of a K-LUT) independent of the function it implements. The output of a K-LUT T can be the input of one or more K-LUTs which are the *fanouts* of T. We call the interconnection net that connects T to its fanouts the *fanout net* of T. The delay of the fanout net of a K-LUT T is determined by several factors, including the parasitic resistance and capacitance of the interconnection wires and programmable switches, as well as the total load capacitance of the fanout K-LUTs that are driven by T. Therefore, the delay of a fanout net usually varies from net to net. Several delay models have been proposed to capture variable net delay, including the *nominal delay model*

[1] The PI and PO nodes are not covered.

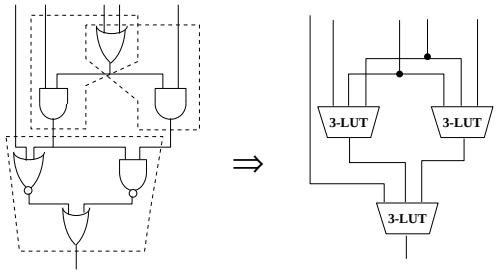

Fig. 1 Mapping a Boolean network to a K-LUT network (where K=3).

[12] which assumes the fanout net delay be proportional to the number of fanouts.

The delay of a path from a PI node to a PO node in a K-LUT network is the sum of the delay of all the K-LUTs along the path and their fanout nets (including the fanout nets of the PIs). The delay of a K-LUT network is the largest delay of any path from a PI to a PO in the network. We say that a mapping solution is *optimal* if its delay is minimum. The objective of our algorithm is to find an optimal mapping solution.

Given a network $N = (V(N), E(N))$ with a source s and a sink t, a *cut* $(X, \bar{X})$ is a partition of the nodes in $V(N)$ such that $s \in X$ and $t \in \bar{X}$. The *node cut-set* of $(X, \bar{X})$, denoted as $C(X, \bar{X})$, is the set of nodes in X that are adjacent to some node in $\bar{X}$, i.e.

$$C(X, \bar{X}) = \{x : (x, y) \in E(N), x \in X \text{ and } y \in \bar{X}\}$$

The *node cut-size* of $(X, \bar{X})$, denoted as $n(X, \bar{X})$, is the number of nodes in $C(X, \bar{X})$. A cut $(X, \bar{X})$ is *K-feasible* if its node cut-size is no more than K, i.e., $n(X, \bar{X}) \le K$. Assuming that each edge (u, v) has a non-negative capacity $c(u, v)$. Then, the *edge cut-size* of $(X, \bar{X})$, denoted $e(X, \bar{X})$, is the sum of the capacities of the edges that go from X to $\bar{X}$, i.e.

$$e(X, \bar{X}) = \sum_{u \in X,\ v \in \bar{X}} c(u, v)$$

Moreover, assuming that there is a given label $l(v)$ associated with each node v. Then, the *height* of a cut $(X, \bar{X})$, denoted $h(X, \bar{X})$, is defined to be the maximum label of the nodes in $C(X, \bar{X})$. Fig. 2 shows a cut $(X, \bar{X})$ in a network with given node labels. Assuming that each edge has unit capacity, then,

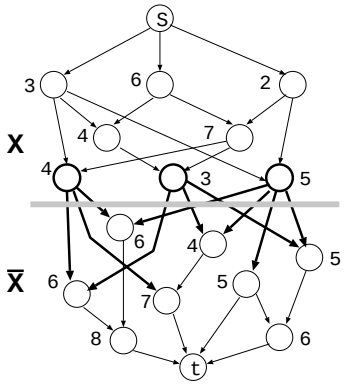

Fig. 2 A 3-feasible cut of edge cut-size 10, node cut-size 3, and height 5.

$n(X, \bar{X}) = 3$, $e(X, \bar{X}) = 10$, and $h(X, \bar{X}) = 5$. The highlighted nodes (edges) form the node cut-set (edge cut-set).

3. DELAY OPTIMAL TECHNOLOGY MAPPING ALGORITHM FOR LUT-BASED FPGAS

Our algorithm is applicable to any *K-bounded* Boolean network. Given a general Boolean network as input, if it is not K-bounded, We first transform it into a 2-input simple gate network using the algorithm discussed in [2]. Note that the optimality of our algorithm holds not only for such 2-input simple gate networks, but also for any K-bounded general Boolean network.

The mapping algorithm presented in this paper is a generalization of FlowMap [3] which produces optimal mapping solution under unit delay model. The algorithm has two phases. In the first phase, it computes a label for each node which reflects the delay of the K-LUT implementing that node in an optimal mapping solution. In the second phase, it generates the K-LUT mapping solution based on the node labels computed in the first phase. The details are discussed in the following subsections.

3.1. The Labeling Phase

Given a K-bounded Boolean network N, let N_t denote the subnetwork consisting of node t and all the predecessors of t. We define the *label* of t, denoted as $l(t)$, to be the delay of the optimal K-LUT mapping solution of N_t. Clearly, the delay of the K-LUT that contains t in an optimal mapping solution of N is at least $l(t)$, and the maximum label of the POs of N determines the delay of the optimal mapping solution of N. The first phase of our algorithm computes such labels for all the nodes in N, according to the topological order starting from the PIs.

Denote the delay of a K-LUT as D_T. For each node u other than the PO nodes, denote the fanout net delay of u as $D(u)$. For the clarity of the algorithm, instead of computing the $l(u)$ directly, we compute $L(u) = l(u) + D(u)$ which can be trivially converted to $l(u)$. For simplicity we will call $L(u)$ the *label* in the remaining of this subsection.

First, for each PI node t, we know that

$$L(t) = D(t). \tag{1}$$

Suppose t is the current node being processed, and t is not a PI or PO. Then, for each node $u \neq t$ in N_t, the label $L(u)$ has been computed. By including in N_t an auxiliary node s and connecting s to all the PI nodes in N_t, we obtain a network with s as the source and t as the sink. For simplicity we still denote it as N_t. Let $LUT(t)$ be the K-LUT that implements node t in a K-LUT mapping solution of N_t, and let X denote the set of nodes in $LUT(t)$ and X denote the remaining nodes in N_t. It is easy to see that $(X, \bar{X})$ forms a K-feasible cut between s and t in N_t, because the number of input nodes of $LUT(t)$ is no more than K. Moreover, let u be a node in $C(X, \bar{X})$. Clearly, u must be implemented by a K-LUT $LUT(u)$ in this mapping solution of N_t. Since $l(u)$ is the minimum delay that any K-LUT implementing u can have, the delay of $LUT(u)$ is at least $l(u)$, and the delay of $LUT(t)$ is at least $L(u) + D_T$. More precisely, if this mapping is optimal, then

$$\begin{aligned} L(t) &\geq \max\{L(u) : u \in C(X, \bar{X})\} + D_T + D(t) \\ &= h(X, \bar{X}) + D_T + D(t). \end{aligned} \tag{2}$$

Since we want to find the minimum label for t that corresponds to the minimum possible delay of any mapping solution of N_t, we need to minimize the righthand side of (2), which is equivalent to minimizing the *height* of the cut $(X, \bar{X})$, if $L(u)$ is regarded as the label of u. Therefore, we need to compute a *minimum height K-feasible cut* [2] $(X^*, \bar{X^*})$ in N_t, which will give t the label

$$L(t) = h(X^*, \bar{X^*}) + D_T + D(t). \tag{3}$$

When t is a PO node, since t does not have to be covered by any K-LUT and it has no fanout, we have

$$L(t) = l(t) = \max\{L(u) : u \in input(t)\}. \tag{4}$$

Based on the above discussion, we have

Lemma 1. Let $L(t)$ be the label computed by Eqs. (1), (3), and (4), then $l(t) = L(t) - D(t)$ gives the minimum delay of any mapping solution of N_t. □

Clearly, the unit delay model is a special case where $D(u)$ is always a constant; by including it into D_T, we can assume $D(u) = 0$, so $l(u) = L(u)$.

The computation of a minimum height K-feasible cut is more complicated in the case of arbitrary delay model, since $L(t)$ may not be monotonic along a path from PI to PO due to the arbitrary net delay $D(u)$. However, we can still identify the possible range of $L(t)$.

Lemma 2. Let t be a node in a network N, t is not a PI or PO. Then,

$$L(t) \leq \max\{L(u) : u \in input(t)\} + D_T + D(t), \tag{5}$$

$$L(t) \geq \max\{D(u) : u \in N_t,\ u \text{ is a PI}\} + D_T + D(t), \tag{6}$$

$$L(t) \geq \max\{L(u) - D(u) : u \in N_t,\ u \text{ is not a PI}\} + D(t). \tag{7}$$

Proof Since $(N_t - \{t\}, \{t\})$ is a K-feasible cut in N_t, (5) is true. Since a PI node cannot be included in any K-LUT, (6) is true. To see (7) is true, consider the two possible cases of u in an optimal mapping solution of N_t.

Case 1: u is contained in $LUT(t)$. In this case, the cut $(X, \bar{X})$ defined by $LUT(t) = \bar{X}$ also induces a K-feasible cut $(Y, \bar{Y})$ in N_u, where $C(Y, \bar{Y}) = C(X, \bar{X}) \cap N_u$. Thus, $h(Y, \bar{Y}) \leq h(X, \bar{X})$, which implies that the minimum height of a K-feasible cut in N_u is no more than $h(X, \bar{X})$. Therefore, $L(u) - D_T - D(u) \leq L(t) - D_T - D(t)$, which implies that $L(t) \geq L(u) - D(u) + D(t)$.

Case 2: u is outside of $LUT(t)$. In this case, either u is implemented by a K-LUT in the mapping solution, which implies that $L(t) \geq L(u) + D_T + D(t) > L(u) - D(u) + D(t)$; or u is covered by the K-LUT implementing another node v that is outside of $LUT(t)$, which implies that $L(v) \geq L(u) - D(u) + D(v)$, and $L(t) \geq L(v) + D_T + D(t)$, and these imply that $L(t) \geq L(u) - D(u) + D(t)$. Therefore,

[2] We exclude the cuts $(X, \bar{X})$ where $\bar{X}$ contains a PI node. Our algorithm to be shown later on guarantees that such kind of cuts are not generated.

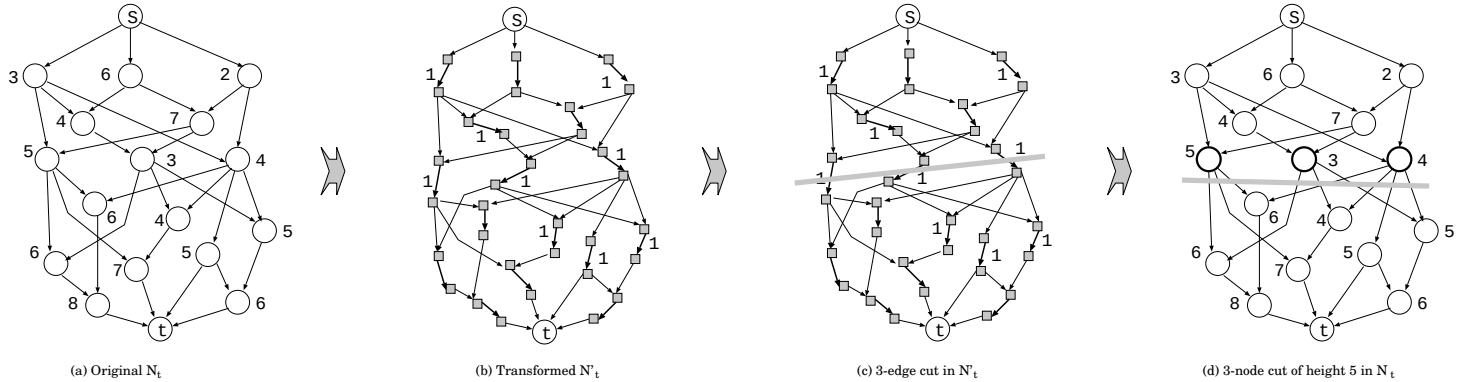

Fig. 3 Network transformation for computing a K-feasible cut of height H in N_t ($H = 5$, $K = 3$).

in either case we have $L(t) \geq L(u) - D(u) + D(t)$, so (7) is true. □

Given the above bounds, we can derive the range of the minimum height any K-feasible cut for N_t may have. If we can determine whether N_t has a K-feasible cut of height H or not, we can perform a binary search over the possible height values to determine the minimum height $H_{\min}$ and hence the label $L(t)$.

To compute a K-feasible cut of height H in N_t, we apply a *node-splitting* transformation on N_t to convert the problem into a standard edge-cut computation problem. Specifically, we obtain a network N'_t as the following: N'_t has the source s and sink t of N_t, and for each of the other nodes u in N_t, N'_t contains two nodes u' and u'', where u' inherits all the incoming edges of u and u'' inherits all the outgoing edges of u. In other words, u is *split* into two nodes. Moreover, u' and u'' are connected by an edge $<u', u''>$ which is called a *bridge edge*, and labeled as u. Finally, we assign capacities to the edges as follows. For each bridge edge $<u', u''>$ labeled u satisfying $L(u) \leq H$ in N_t, the capacity of $<u', u''>$ is one; for all other edges, the capacity is infinite.

Lemma 3. N_t has a K-feasible cut of height no more than H if and only if N'_t has a cut whose edge cut-size is no more than K.

Proof Suppose N_t has a K-feasible cut $(X, \bar{X})$ of height H. Then, the size of $C(X, \bar{X})$ is no more than K, and the label of any node in $C(X, \bar{X})$ is no more than H. Therefore, the set of bridge edges in N'_t corresponding to the nodes in $C(X, \bar{X})$ all have capacity one, which form an edge cut-set of size no more than K in N'_t.

On the other hand, if N'_t has a cut $(X', \bar{X}')$ such that $e(X', \bar{X}') \leq K$, clearly, the edge cut-set only contains the bridge edges that correspond to the nodes in N_t with labels no more than H, since the capacities of all other edges are infinite. Because this is an edge cut-set, the corresponding node set in N_t forms a node cut-set $C(X, \bar{X})$ for some cut $(X, \bar{X})$. Clearly, $n(X, \bar{X}) = e(X', \bar{X}')$. Therefore, $(X, \bar{X})$ is a K-feasible cut in N_t of height no more than H. □

This proof also shows how to convert an edge cut in N'_t to a node cut in N_t. Fig. 3 illustrates the entire transformation. In Fig. 3(b), all the edges whose capacities are not specified have infinite capacity. Fig. 3(c) shows the cut computed from (b), and in (d) it is converted back to a cut in the original network (a).

According to the Max-flow Min-cut Theorem [14], N'_t has a cut whose edge cut-size is no more than K if and only if the maximum flow between s and t in N'_t has value no more than K. Since we are only interested in testing if the maximum flow is of value K or smaller, we apply the augmenting path algorithm in N'_t to compute a maximum flow. (For details of the augmenting path algorithm, see [14].) From the construction of N'_t, it can be seen that each augmenting path in the flow residual graph of N'_t from s to t increases the flow by either one or infinity. If we can find a path of infinite capacity, or $K + 1$ augmenting paths, then the maximum flow in N'_t has value more than K, and we can conclude that N'_t does not have a cut $(X', \bar{X}')$ with $e(X', \bar{X}') \leq K$. Otherwise, the residual graph is disconnected before we find the $(K + 1)$-th augmenting path, and the disconnected residual graph induces a cut of edge cut-size no more than K. Moreover, we can find such a cut $(X', \bar{X}')$ by performing a depth first search starting at the source s, and including in X' all the nodes which are reachable from s. Since finding an augmenting path takes $O(m)$ time, where m is the number of edges in the residual graph of N'_t (which is in the same order as the number of edges in N_t), we can determine in $O(Km)$ time whether N'_t has a cut of edge cut-size no more than K and find one if such a cut exists. Such a cut $(X', \bar{X}')$ in N'_t induces a K-feasible cut $(X, \bar{X})$ of height no more than H in N_t.[3]

In practice, we can simplify N'_t as the following. For any node u in N'_t other than s and t, if u has a single outgoing edge $<u, v>$ of infinite capacity, it cannot be in any edge cut-set, and any path that reaches u must go to v. In this case, u can be collapsed into v and the edge $<u, v>$ can be removed. Similarly, if u has a single incoming edge $<v, u>$ of infinite capacity, u can also be collapsed into v. This may reduce the size of the network significantly. For the example shown in Fig. 3(b), such collapsing will reduce the number of nodes from 34 to 14, and reduce the number of edges from 48 to 25.

Being able to determine if a K-feasible cut of height H exists, by binary search we can easily determine the smallest $H = H_{\min}$ such that a K-feasible cut of height $H_{\min}$ exists. Let the cut be $(X, \bar{X})$, then, the implementation of t in an optimal mapping solution of N_t is given by $LUT(t) = \bar{X}$, and $L(t) = H_{\min} + D_T + D(t)$. In other words, $l(t) = H_{\min} + D_T$. Note that the height of any cut must be a node label according to the definition. Therefore, the range of the binary search is

the set of different labels $L(u)$ for $u \in N_t$, restricted by Lemma 2. In any case, the number of different labels never exceeds the number of nodes in N_t. Based on the above discussion, we have

Theorem 1. A minimum height K-feasible cut in N_t can be found in $O(Km\log n)$ time where m and n is the number of edges and nodes in N_t. □

Note that although a K-feasible cut can also be computed by trivially enumerating all possible combinations of K or less nodes, it yields a complexity of $O(n^K)$, which is significantly larger when n or K is not very small.

Applying Theorem 1 to each node in N in the topological order in the labeling phase, we have

Corollary 1. The labels of all the nodes in N can be computed in $O(Kmn\log n)$ time, where n and m are the number of nodes and edges in N, respectively. □

In current technology, K is usually 4 or 5. Moreover, for a K-bounded network, $m = O(Kn)$. Therefore, the complexity of the labeling phase of our algorithm is $O(n^2\log n)$ in practice.

3.2. The Mapping Phase

In the labeling phase, for each node u we have computed $LUT(u)$, the K-LUT implementing u in an optimal mapping of N_u. The second phase of our algorithm is to generate the K-LUTs in the optimal mapping solution. Since this phase is the same as that in FlowMap [3], we only give a brief description. We generate K-LUTs only for nodes which have fanouts to POs or other K-LUTs. During the mapping phase, we maintain a list L of nodes which have to be implemented by K-LUTs. Initially, L contains those nodes that have fanouts to the PO nodes. Then, we repeatedly remove a node u from L, introduce the K-LUT $LUT(u)$ computed in the first phase as the K-LUT implementing u, and add into L all the nodes in $input(LUT(u))$, which are fanins of $LUT(u)$. (The nodes that are never added into L do not need to be implemented since they are completely covered by the K-LUTs implementing other nodes.) The mapping phase ends when L only contains PI nodes. It is easy to see that the resulting network is logically equivalent to the original network, and the whole second phase takes linear time. Moreover, it is not hard to see that the delay from any PI to a K-LUT $LUT(u)$ is no more than $l(u) = L(u) - D(u)$. Therefore, the mapping solution is optimal. (A formal proof for similar conclusion under unit delay model can be found in [3].)

In summary, we have

Theorem 2. For any K-bounded Boolean network N, the algorithm produces a K-LUT mapping solution with the minimum delay under arbitrary net delay model in $O(Kmn\log n)$ time, where n and m are the number of nodes and edges in N. □

4. EXPERIMENTAL RESULTS

We have implemented the above algorithm as an extension of the FlowMap algorithm. The new mapping algorithm is named FlowMap-d, written in C language on SUN SPARC workstations. We have tested it under a simple non-unit net delay model: we assume that the delay of a K-LUT is a constant, and that the delay of the fanout net of a node is proportional to the number of its fanouts nodes. Such a delay model is called the *nominal delay model* [12]. Moreover, in order to make a fair comparison with FlowMap, we assume that the K-LUT delay is larger than any fanout net delay[4], so that it is guaranteed that the mapping solution of FlowMap-d also has the minimum number of levels. Therefore, the difference between the solution of FlowMap-d and that of FlowMap is merely due to the net selection along critical paths, which is determined by their different assumptions on net delay.

We have tested the algorithm on four mid-size MCNC benchmark circuits such that the mapping solution of each circuit can be implemented by a single FPGA chip of Xilinx 3000 series, which consists of 5-LUTs [16]. The initial circuits are 2-input simple gate networks, which are the same as those used in [3]. For each of the four circuits, we applied FlowMap and FlowMap-d (under the nominal delay model defined in the preceding paragraph) separately, and then used FlowPack [3] to minimize the area for both solutions. For each of the final mapping solutions, we have computed both the estimated delay under the nominal delay model and the real delay obtained through placement and routing using Xilinx XACT design tool. The results are summarized in Table 1, where we show the estimated delay under the nominal delay model (normalized to 100 for each FlowMap solution), and the real delay (in nanoseconds) measured as the worst case pad-to-pad delay using Xilinx tool.

As can be seen from the table, both algorithms generate solutions of the same number of levels, which are minimum in this sense. FlowMap-d uses more 5-LUTs than FlowMap, since FlowMap is less restricted in selecting the mapping solution (FlowMap computes *any* mapping solution of the minimum level, while FlowMap-d must compute the one with both the minimum level and the minimum net delay). However, FlowMap-d produces solutions with smaller delays. Since the delay model is very primitive, the delay prediction is not very accurate. We expect that more accurate delay models may lead to solutions with even smaller delays. For instance, we may incorporate the information obtained from an estimated placement solution. The estimation of the fanout net size can also be improved by taking the node duplication into consideration.

5. CONCLUSION

In this paper we have presented an efficient algorithm to carry out performance-driven technology mapping for LUT-based FPGA designs. Our algorithm is applicable to general Boolean networks and guarantees optimal solution under arbitrary net delay models. We have implemented our algorithm and tested it on a set of MCNC benchmark examples.

The capability of taking arbitrary delay models allows our algorithm to be used as an evaluation tool for the study of

[3] It is clear that for the resulting cut $(X, \bar{X})$ in N_t, $\bar{X}$ does not contain any PI nodes since any outgoing edge of the source s in N'_t has infinite capacity.

[4] In general, this assumption may not always be true. We have also tested the models without this restriction. However, in those cases the delay difference between the solutions of FlowMap-d and FlowMap also depends on the K-LUT delay assumption, making the comparison more complicated.

Table 1. Comparison of FlowMap-d with FlowMap for Delay minimization										
circuit	**FlowMap**				**FLowMap-d**				**Improvement**	
	#LUTs	**#levels**	**estimated delay**	**real delay**	**#LUTs**	**#levels**	**estimated delay**	**real delay**	**estim. delay**	**real delay**
9sym	65	5	100	101.7	69	5	97.3	99.4	2.7%	2.3%
alu2	162	8	100	192.9	196	8	98.9	187.9	1.1%	2.6%
count	77	3	100	89.1	79	3	95.7	77.0	4.3%	13.6%
vg2	46	4	100	78.8	53	4	57.8	78.3	42.2%	0.7%
total	350	20	400	462.5	397	20	349.7	442.6	-	-

delay models. The modeling of FPGA interconnection delay is very important. The nominal delay model considers the driving load as a primary factor that affects the delay. There are many other factors. The placement and global routing result determines the wire length and the number of programmable switches of each net, while the channel routing result determines, in the case of Xilinx 3000 series, the number of pass transistors each connection will go through, which result in considerable capacitance. Accurate prediction of such factors in the mapping stage is a difficult task. Previous mapping algorithms are either restricted to unit delay model, or do not guarantee the optimality of their solutions. Therefore, they cannot be used to compare various delay models on a fair basis. In contrast, our algorithm generates optimal solutions under arbitrary net delay models, thus makes the accurate comparison possible.

Since delay estimate relies on information about placement and routing, which is not available in the mapping stage, an iterative approach that combines mapping with placement and routing will be beneficial. Unlike the Boolean optimization based algorithms, our algorithm is very efficient even for large scale circuits, hence is affordable to be integrated with placement and routing in such an iterative procedure.

The development of accurate delay model is beyond the scope of this paper. The simple delay model used in our experiments only serves the testing purpose. Currently, using our delay optimal mapping algorithm, we are studying more accurate delay models for FPGA interconnections.

ACKNOWLEDGMENT This research is partially supported by a grant from Xilinx Inc. under the State of California MICRO program No.92-030.

REFERENCES

[1] Bhat, N. and D. Hill, ‘‘Routable Technology Mapping for FPGAs,’’ *First Int’l ACM/SIGDA Workshop on Field Programmable Gate Arrays*, pp. 143-148, Feb. 1992.

[2] Chen, K. C., J. Cong, Y. Ding, A. B. Kahng, and P. Trajmar, ‘‘DAG-Map: Graph-based FPGA Technology Mapping for Delay Optimization,’’ *IEEE Design and Test of Computers*, Sep. 1992.

[3] Cong, J. and Y. Ding, ‘‘An Optimal Technology Mapping Algorithm fo Delay Optimization in Lookup-Table Based FPGA Designs,’’ *Proc. IEEE Int’l Conf. on Computer-Aided Design*, Nov. 1992.

[4] Francis, R. J., J. Rose, and Z. Vranesic, ‘‘Technology Mapping for Delay Optimization of Lookup Table-Based FPGAs,’’ *MCNC Logic Synthesis Workshop*, 1991.

[5] Francis, R. J., J. Rose, and Z. Vranesic, ‘‘Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs,’’ *Proceedings 28th ACM/IEEE Design Automation Conference*, pp. 613-619, 1991.

[6] Hill, D., ‘‘A CAD System for the Design of Field Programmable Gate Arrays,’’ *Proc. ACM/IEEE Design Automation Conference*, pp. 187-192, 1991.

[7] Karplus, K., ‘‘Xmap: A Technology Mapper for Table-lookup Field-Programmable Gate Arrays,’’ *Proc. 28th ACM/IEEE Design Automation Conference*, pp. 240-243, 1991.

[8] Murgai, R., et al, ‘‘Logic Synthesis Algorithms for Programmable Gate Arrays,’’ *Proc. 27th ACM/IEEE Design Automation Conf.*, pp. 620-625, 1990.

[9] Murgai, R., N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, ‘‘Performance Directed Synthesis for Table Look Up Programmable Gate Arrays,’’ *Proc. IEEE Int’l Conf. Computer-Aided Design*, pp. 572-575, Nov., 1991.

[10] Murgai, R., N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, ‘‘Improved Logic Synthesis Algorithms for Table Look Up Architectures ,’’ *Proc. IEEE Int’l Conf. Computer-Aided Design*, pp. 564-567, Nov., 1991.

[11] Sawkar, P. and D. Thomas, ‘‘Technology Mapping for Table-Look-Up Based Field Programmable Gate Arrays,’’ *ACM/SIGDA Workshop on Field Programmable Gate Arrays*, pp. 83-88, Feb. 1992.

[12] Schlag, M., P. Chan, and J. Kong, ‘‘Empirical Evaluation of Multilevel Logic Minimization Tools for a Field Programmable Gate Array Technology,’’ *Proc. 1st Int’l Workshop on Field Programmable Logic and Applications*, Sept. 1991.

[13] Schlag, M., J. Kong, and P. K. Chan, ‘‘Routability-Driven Technology Mapping for Lookup Table-Based FPGAs,’’ *Proc. 1992 IEEE International Conference on Computer Design*, Oct. 1992.

[14] Tarjan, R. E., *Data Structures and Network Algorithms*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania (1983).

[15] Woo, N.-S., ‘‘A Heuristic Method for FPGA Technology Mapping Based on the Edge Visibility,’’ *Proc. 28th ACM/IEEE Design Automation Conference*, pp. 248-251, 1991.

[16] Xilinx, *The Programmable Gate Array Data Book*, Xilinx, San Jose (1992).