On Nominal Delay Minimization in LUT-Based FPGA Technology Mapping*

Jason Cong and **Yuzheng Ding**
UCLA Computer Science Department, Los Angeles, CA 90024

Abstract

We study the nominal delay minimization problem in LUT-based FPGA technology mapping, where interconnect delay is assumed proportional to net fanout size. We prove that the delay-optimal K-LUT mapping problem under the nominal delay model is NP-hard when $K \geq 3$, and remains NP-hard for duplication-free mapping and tree-based mapping for $K \geq 5$ (but is polynomial time solvable for $K = 2$). We also present a simple heuristic to take nominal delay into consideration during LUT mapping for delay minimization.

1 Introduction

Lookup-table (LUT) based FPGA [8, 10] is a popular architecture in which the basic programmable logic block is a K-input lookup-table (K-LUT), built in SRAM, 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 into a functionally equivalent network of K-LUTs by computing a (not necessarily disjoint) K-LUT covering of the network.

Extensive study has been done on the mapping algorithms for LUT-based FPGAs in recent years. Researchers have focused on area minimization, delay minimization, trade-off between area and delay, and routability optimization, etc.. Efforts have been made both on the development of effective and efficient mapping algorithms, and on the study of the complexity of the mapping problems. It has been shown that if the network is a tree, or if we use *tree-based mapping* (i.e. by decomposing a general network into trees and mapping each tree separately), both the area minimum mapping problem and the depth minimum mapping problem can be solved optimally in strong polynomial time [5, 6]. For general K-bounded Boolean networks, it is shown that the depth minimum mapping problem can be solved optimally in strong polynomial time [1], while the area minimum mapping problem is NP-hard for $K \geq 5$ [5]. If we allow only *duplication-free mapping* (i.e. do not allow node duplication during mapping), it has been shown that both the depth minimum mapping problem and the area minimum mapping problem can be solved optimally in polynomial time for any fixed K [2]. These results are summarized in Table 1.

	tree-based mapping	dup-free mapping	general mapping	
area min.	$O(\min\{nK, n\log n\})$ [5]	$O(n^{K+1})$ [2]	$2 \leq K < 5$	$K \geq 5$
			open	NP-hard [5]
depth min.	$O(\min\{nK, n\log n\})$ [5]	$O(Kmn)$ [1]	$O(Kmn)$ [1]	

Table 1: Summary of previous complexity results on K-LUT mapping of n-node m-edge network

Delay minimization has been an important optimization objective in FPGA mapping because the speed of FPGA designs is usually slower than that of the gate array or standard cell designs due to the extra delay introduced by the programmable interconnects on FPGA chips. Most previous mapping algorithms for delay minimization use the depth of the mapping solution as the measurement of delay, i.e. based on the *unit delay model*, which assumes uniform delay at every logic level. As shown in [1], the depth minimization problem can be solved optimally in polynomial time by efficient network flow computation. However, the assumption made by the unit delay model is usually over-simplified. In LUT-based FPGA designs, although the delay of each LUT is a constant, the interconnect delay of each net may vary considerably. Since interconnect delay contributes a significant portion to the total delay, it is natural to ask whether this portion of delay can be more accurately estimated during mapping.

Experiments in [9] have shown that the interconnect delay in an FPGA chip is closely related to the number of fanouts of the net. This is intuitively true: because an FPGA chip has fixed routing resources, a net with larger fanouts must spread over a larger area and use up more routing resources, and thus has larger delay. It is important to notice that such net structure based delay estimation is *dynamic* since the fanout size of a net may change during mapping.

In this paper we study the LUT-based FPGA technology mapping problem for delay minimization under the *nominal delay model*, which assumes that the net delay is proportional to the net fanout size. This is a very simple case of dynamic delay model. However, even under this model we can show that the delay-optimal mapping problem for K-

*Partially supported by NSF Young Investigator Award MIP9357582, ARPA/CSTO under Contract DABT63-93-C-0055, and grants from AT&T, Fujitsu Laboratories of America, Hewlett-Packard, and Xilinx under the 1993–94 California MICRO Program and NYI matching program.

LUTs becomes NP-hard for $K \geq 3$, and remains NP-hard for $K \geq 5$ even for duplication-free mapping and tree-based mapping (but is polynomial time solvable when $K = 2$). These complexity results are summarized in Table 2.

	$k = 2$	$K = 3, K = 4$	$K \geq 5$
tree-based mapping	$O(mn)$	open	NP-hard
dup-free mapping	$O(mn)$	open	NP-hard
general mapping	open	NP-hard	

Table 2: Summary of complexity results on K-LUT mapping for nominal delay minimization on n-node m-edge network

These results indicate the difficulty of dynamic delay estimation in FPGA mapping. On the other hand, since net structure is closely related to net delay, it is helpful to use such information, even approximately. In this paper we will also present a simple heuristic to demonstrate the improvement of mapping quality over the depth-optimal mapping by taking nominal delay into consideration.

Due to page limit most of the proofs will be omitted. Interested readers may refer to [3] for details.

2 Problem Formulation

A combinational Boolean network is represented as a directed acyclic graph in which nodes represent logic gates, and edges represent interconnects. A primary input (PI) is represented by a node without incoming edge, and a primary output (PO) is represented by a node without outgoing edge. The set of fanins of gate v is denoted $input(v)$, and the set of *distinct* nodes which supply inputs to the gates in subnetwork H is denoted $input(H)$. Similarly, the set of fanouts of v is denoted $output(v)$, and the set of distinct fanouts of a subnetwork H is denoted $output(H)$. The *level* (or *depth*) of a node v is the number of edges on the longest path from any PI node to v. The $depth$ of a network is the largest node level in the network. A Boolean network is *K-bounded* if $|input(v)| \leq K$ for every node v. In this paper we assume that the networks to be mapped are always K-bounded.

For a node v in the network, a *cone of* v, denoted C_v, is a subgraph of logic gates consisting of v and its predecessors such that any path connecting a node in C_v and v lies entirely in C_v. The *root* of C_v is v. The *fanin cone* of node v, denoted N_v, consists of v and all the predecessors of v. A *fanout-free cone (FFC) at* v, denoted FFC_v, is a cone of v such that for any node $u \neq v$ in FFC_v, $output(u) \subseteq FFC_v$. A *K-feasible cone of* v is a cone C_v such that $|input(C_v)| \leq K$. A *cut* in a fanin cone N_v of node v is a bipartition $(X, \overline{X})$ of N_v such that $\overline{X}$ is a cone of v, and for every PI node $w \in N_v$, $w \in X$. If $\overline{X}$ is a K-feasible cone, the cut is called a *K-feasible cut* .

A K-LUT LUT_v that implements node v *covers* a K-feasible FFC C_v of v. If C_v is not fanout free, the non-root nodes in C_v that have fanouts outside of C_v must be duplicated in order to cover C_v by a K-LUT. Given a K-bounded network, the *technology mapping problem* for K-LUT based FPGA designs is to cover the network with K-feasible FFCs, possibly with node duplications. If node duplication is not allowed, it is *duplication-free mapping*. If the network is first decomposed into trees, and each tree is then mapped separately, it is *tree-based* mapping.

Given a node v, the *nominal delay* associated with v is defined as

$$D(LUT_v) = d_L + |output(v)| \cdot d_N, \qquad (1)$$

where d_L is the delay of a K-LUT, which is a constant for a given technology, and $d_N > 0$ is a constant representing the additional delay due to adding a fanout branch to the net (which is determined by the technology, the placement/routing tools, and the style of the design, etc.). In practice, d_N is usually not a constant. However, even under such a simplified model, the delay minimization problem is much more difficult than the depth minimization problem in LUT mapping.

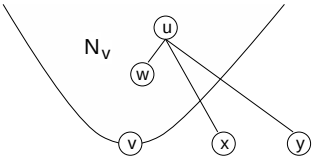

Figure 1: Complication of mapping under the nominal delay model

Under the unit delay model, for any node v, the minimum depth of a K-LUT that implements v only depends on the depth-optimal mapping of the fanin cone N_v of v. This allows the depth-optimal solution to be computed using dynamic programming approach. Under the nominal delay model, however, the minimum delay of a K-LUT that implements v may also depend on the mapping of nodes outside of N_v. Figure 1 illustrates this scenario. Let u be a predecessor of v which has fanouts x and y outside of N_v. If both nodes x and y are packed into one LUT, the nominal delay of node u decreases, since the fanout size of u is decreased by one. On the other hand, if x or y is duplicated, the nominal delay of u will increase. Therefore, delay-optimal mapping of v clearly depends on the mapping of the nodes x and y outside of N_v. This is the inherent difficulty of nominal delay minimization in LUT mapping, which leads to the NP-hardness results in the next section.

3 Complexity Results

Our proof of the NP-hardness of the delay-optimal LUT mapping problem will be based on polynomial time transformations from the **RSAT** problem to the decision version

of the delay-optimal LUT mapping problem under the nominal delay model. We first define the **RSAT** problem, which is a restricted case of the well-known NP-complete problem **SAT**, and remains NP-complete. We shall then transform **RSAT** to the decision version of the nominal delay minimization problems.

Problem: Restricted Satisfiability (**RSAT**)

Instance: A set of n Boolean variables $\mathcal{X}= \{x_1, x_2, ..., x_n\}$ and a collection of m clauses $\mathcal{C} = \{C_1, C_2, ..., C_m\}$, where (a) each clause is the disjunction (OR) of 2 or 3 literals of the variables, (b) each clause contains at most one of x_i and $\overline{x}_i$ for any variable x_i, (c) both literals x_i and $\overline{x}_i$ of any variable x_i appear in some of the clauses, and (d) for any variable x_i, the number of clauses containing either x_i or $\overline{x}_i$ is no more than 3.

Question: Is there a truth assignment of the variables in $\mathcal{X}$ such that $C_j = 1$ for $1 \leq j \leq m$?

By adding the conditions to **SAT** while maintaining equivalence, we have shown

Lemma 1 **RSAT** *is NP-complete.* ∎

Note that conditions (c) and (d) indicate that each literal will appear in either one or two clauses. This is important to our construction of the transformations.

3.1 General Mapping

We first define the decision version of the general delay-optimal mapping problem under the nominal delay model.

Problem: Depth-Bounded LUT Mapping under Nominal Delay Model (**DBLMN**)

Instance: A constant $K \geq 2$, a Boolean network N of p nodes and q edges such that for any $v \in N$, $|input(v)| \leq K$, and three other constants $d_L \geq 0, d_N \geq 1$, and B.

Question: Under the nominal delay model with parameters d_L, d_N, is there a K-LUT mapping solution of N that has delay no more than B?

We shall construct a polynomial time transformation that transforms each instance of **RSAT** to an instance of **DBLMN**. Intuitively, we want to relate the decision of the truth assignment of a Boolean variable in an instance $\mathcal{F}$ of **RSAT** to the decision of a node duplication in the corresponding network N of the **DBLMN** instance. Since determining the truth assignment is difficult, we can show that determining the node duplication is also difficult.

Given an instance $\mathcal{F}$ of the **RSAT** with n variables $x_1, x_2, ..., x_n$ and m clauses $C_1, C_2, ..., C_m$, we construct a K-bounded Boolean network N corresponding to the instance $\mathcal{F}$ as follows.

First, for each variable x_i, we construct a subnetwork $N(x_i)$ that consists of the following nodes: (a) Four internal nodes denoted as $x_i, x_i^+, \overline{x}_i$, and $\overline{x}_i^+$; (b) $5(K-1)-(L_{x_i} + L_{\overline{x}_i})$ PI nodes, where L_{x_i} and $L_{\overline{x}_i}$ are the number of clauses containing x_i and $\overline{x}_i$, respectively; (c) A *super-PI* node N_x, which is a subnetwork to be defined later; and (d) Three PO nodes denoted as O_i^1, O_i^2, O_i^3. The nodes are connected as shown in Figure 2(a), where N_x, $x_i, x_i^+, \overline{x}_i, \overline{x}_i^+$, and O_i^3 form a direct chain, while x_i and $\overline{x}_i$ are connected to O_i^1 and O_i^2, respectively. There are $K-1$ PI nodes connected to each of the PO nodes, $K-1-L_{x_i}$ PI nodes connected to both x_i and x_i^+, and similarly $K-1-L_{\overline{x}_i}$ PI nodes connected to both $\overline{x}_i$ and $\overline{x}_i^+$. Note that for $K \geq 3$, $K-1-L_{x_i}$ and $K-1-L_{\overline{x}_i}$ will never be less than zero, since L_{x_i} and $L_{\overline{x}_i}$ are at most 2. Therefore, for $K \geq 3$ the construction is feasible. We call this subnetwork the *subnetwork of* x_i.

Next, for each clause C_j with 3 literals l_j^1, l_j^2, l_j^3, we construct a subnetwork $N(C_j)$ consisting of the following nodes: (a) Three internal nodes denoted as l_j^1, l_j^2, and l_j^3; (b) $3K-2$ PI nodes; (c) One *super-PO* node N_c, which is a subnetwork to be defined later. The connection is shown in Figure 2(b), where l_j^1, l_j^2, l_j^3 and N_c form a direct chain, and each has K, $K-1$, $K-1$, and 0 PI nodes connected to it, respectively.

If the clause C_j contains only 2 literals l_j^1 and l_j^2, the subnetwork $N(C_j)$ is constructed with the following nodes: (a) Two internal nodes l_j^1, l_j^2, and a dummy internal node d_j; (b) $4K-3$ PI nodes; (c) Two PO nodes d_j^1 and d_j^2; (d) One super-PO node N_c. The connection as shown in Figure 2(c) is similar to the case of 3-literal clause, except that the additional $K-1$ PI nodes and two PO nodes are attached to the dummy node d_j.

The super-PI node N_x and the super-PO node N_c are shown in Figure 2(d). N_x is a three level subnetwork consisting of 3 internal nodes, 4 PO nodes, and $7K-8$ PI nodes, and has one outgoing edge. N_c is a four level subnetwork with 3 internal nodes, 4 PO nodes, and $7K-7$ PI nodes, and has one incoming edge. The subnetworks N_x and N_c are used to balance the delay along different paths in N, which is necessary to the proof.

Finally, we connect the subnetworks $N(C_j)$, $j = 1, 2, ..., m$, with the subnetworks $N(x_i)$, $i = 1, 2, ..., n$. Let l_j^r ($1 \leq r \leq 3$) be a literal in C_j. If for some variable x_i, $l_j^r = x_i$, we connect the internal node l_j^r of subnetwork $N(C_j)$ to the internal node x_i of subnetwork $N(x_i)$. Similarly, if $l_j^r = \overline{x}_i$, we connect node l_j^r of $N(C_j)$ to node $\overline{x}_i$ of $N(x_i)$. We call x_i (or $\overline{x}_i$) the *variable node of* l_j^r, and call l_j^r a *literal node of* x_i (or $\overline{x}_i$). Such connections are illustrated in Figure 3, and also in Figures 2(a–c) (in dashed lines).

It is clear that the transformation defined above takes $O(K(m+n))$ time. Examples of transformations can be found in [3]. Regarding the network N obtained through this transformation, we can see that N is K-bounded when $K \geq 3$, and the only ways of packing more than one node into a K-LUT are to pack x_i into the K-LUT of x_i^+ (denoted as $pack(x_i)$), and to pack $\overline{x}_i$ into the K-LUT of $\overline{x}_i^+$ (denoted

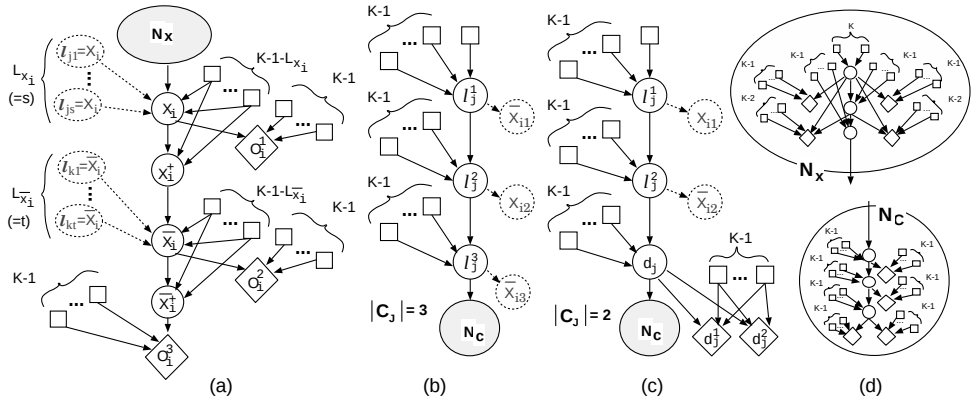

Figure 2: Transformation of **RSAT** instance to **DBLMN** instance: (a) $N(x_i)$ for variable x_i; (b) $N(C_j)$ for clause C_j with 3 literals; (c) $N(C_j)$ for clause C_j with 2 literals; (d) *Super-nodes* N_x and N_c

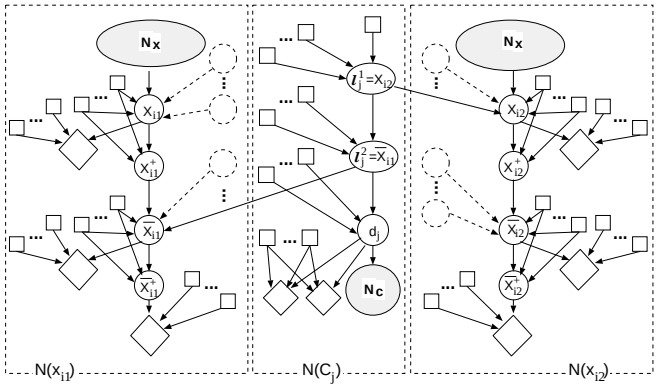

Figure 3: Transformation of **RSAT** instance to **DBLMN** instance: Connections between $N(x_i)$ and $N(C_j)$

as $pack(\overline{x}_i)$), for $1 \leq i \leq n$. Moreover, every node must be implemented by a K-LUT regardless such packing (i.e. if such a packing happens, node duplication must happen). By linking the variable assignment of $x_i = 0$ for the **RSAT** instance $\mathcal{F}$ with the duplication and packing of node x_i in the network N (and assignment of $x_i = 1$ with the duplication and packing of $\overline{x}_i$), we can show the NP-completeness of the **DBLMN** problem. We start with a lemma about the transformation from a satisfiable **RSAT** instance to a delay-bounded **DBLMN** instance.

Lemma 2 *If $\mathcal{F}$ is satisfiable, then N has a mapping solution of delay $7d_L + 15d_N$.* ∎

The proof is obtained by examining three types of PI to PO paths, i.e. paths entirely inside $N(x_i)$, paths entirely inside $N(C_j)$, and paths from $N(C_j)$ to $N(x_i)$, and calculating the largest possible path length. We omit the analysis due to page limit.

In order to derive a truth assignment that satisfies $\mathcal{F}$ from a delay-bounded mapping solution of N, we first analyze the characteristics of such a mapping solution.

Lemma 3 *In a mapping solution of N with delay bound $7d_L + 15d_N$, at least one of the operations $pack(x_i)$ and $pack(\overline{x}_i)$ must be performed for each i, $1 \leq i \leq n$.* ∎

Lemma 4 *In a mapping solution of N with delay bound $7d_L + 15d_N$, for each $N(C_j)$, $1 \leq j \leq m$, at least one of the operations $pack(l_j^1)$, $pack(l_j^2)$, and (when $|C_j| = 3$) $pack(l_j^3)$* cannot *be performed.* ∎

According to Lemmas 3 and 4, if a mapping solutions satisfies delay bound $7d_L + 15d_N$, it will be similar to the one we constructed in the proof of Lemma 2, except that for some x_i, it is possible that both $pack(x_i)$ and $pack(\overline{x}_i)$ are performed. In such a case, however, the largest delay of a path in $N(x_i)$ is smaller than $7d_L + 15d_N$, and one of the packing operations is not necessary, hence can be ignored in the construction of a truth assignment for $\mathcal{F}$ from the mapping solution. This implies

Lemma 5 *If N has a mapping solution S of delay no more than $7d_L + 15d_N$, then $\mathcal{F}$ is satisfiable.* ∎

The proofs of the above lemmas are omitted due to page limit.

Therefore, by setting $B = 7d_L + 15d_N$, we can show

Lemma 6 **DBLMN** *is NP-complete for $K \geq 3$.*

Proof It is easily seen the the transformation takes polynomial time. Moreover, **DBLMN** is in NP because given any K-LUT mapping solution, we can easily verify if its delay is bounded by B. Finally, according to Lemmas 2 and 5, the **RSAT** instance $\mathcal{F}$ has YES answer if and only if the **DBLMN** instance N has YES answer. Therefore, the

NP-completeness of **RSAT** implies that **DBLMN** is NP-complete. ∎

Based on Lemma 6, we have

Theorem 1 *The delay-optimal K-LUT mapping problem under the nominal delay model is NP-hard when* $K \geq 3$. ∎

Note that the construction of N does not apply when $K = 2$, so the complexity of the problem is still open for $K = 2$.

3.2 Duplication-Free Mapping & Tree-Based Mapping

We have proved the NP-hardness of the delay-optimal general LUT mapping problem under the nominal delay model by relating the truth assignment of Boolean variables in an **RSAT** instance to the duplication of nodes in the mapping of corresponding network. An interesting question is whether node duplication is the only cause of the difficulty. In this subsection we answer this question negatively, i.e. even when node duplication is prohibited, or the mapping is restricted to be tree-based, the delay-optimal LUT mapping under nominal delay is still NP-hard when $K \geq 5$. This result indicates that the delay minimization problem is more difficult than the area minimization problem, which is polynomial time solvable for duplication-free mapping or tree-based mapping.

For this proof, we again construct a polynomial time transformation from an instance of **RSAT** to an instance of **DBLMN**. The network for the **DBLMN** instance is carefully constructed so that duplication-free mapping or tree-based mapping can be easily enforced. Since node duplication is not allowed in either case, we link the Boolean variable assignment to the possible packing of multiple fanouts of the same node into a single K-LUT, and show that even the choice of packing operations itself is difficult. Due to page limit, we leave out the construction and the proof, and state the following theorem directly.

Theorem 2 *The delay-optimal duplication-free K-LUT mapping problem and The delay-optimal tree-based K-LUT mapping problem under the nominal delay model are both NP-hard when* $K \geq 5$. ∎

The construction of the network for the proof of the above result is feasible only when $K \geq 5$. For $K < 5$, the complexity problem is open except for $K = 2$, which will be solved in polynomial time in the next subsection.

3.3 Optimal Duplication-Free Mapping for $K = 2$

As defined in [2], the *maximum fanout-free cone* (MFFC) of a node v, denoted $MFFC_v$, is an FFC of v such that for any node $w \neq v$, $w \in MFFC_v$ if $output(w) \subseteq MFFC_v$. One property of MFFC is that a network can be decomposed into a set of disjoint MFFCs. The MFFC decomposition is important because every duplication-free mapping solution contains a mapping solution of each MFFC in the decomposition. In the case of area minimization, this property allows us to map each MFFC independently and compose an optimal solution of the entire network from the optimal solutions of the MFFCs [2].

In the case of nominal delay minimization, in general this method no longer works because the delay of some input nets to an MFFC can be changed due to the mapping of other MFFCs that share the same input net. However, if $K = 2$, we can still combine the optimal mapping solutions of the MFFCs to obtain an optimal mapping solution for the entire network. Specifically, we have shown that the following simple algorithm will produce the delay-optimal mapping solution if there is no cut of size one existing in the network[1]:

1. Decompose the network into disjoint MFFCs;
2. For each MFFC $MFFC_v$, do
   - 2.1. Find the maximum-volume[2] min-cut $(X, \overline{X})$ of cut-size 2, and assign $LUT_v := \overline{X}$;
   - 2.2. If $MFFC_v - LUT_v \neq \vee$, recursively map $MFFC_v - LUT_v$ using this algorithm.

The proof of its optimality, omitted due to page limit, is based on the fact that for $K = 2$ and in the absence of the cuts of size one, any K-feasible cut is a min-cut; and the following property of the maximum-volume min-cut proved in [1],

Lemma 7 *For any cone* C_v *of a node* v*, there exists a unique maximum-volume*[3] *min-cut* $(X, \overline{X})$ *of* C_v *such that for any other min-cut* $(X', \overline{X'})$ *of* C_v, $\overline{X'} \subset \overline{X}$. ∎

The cut computation in step 2.1 can be implemented using the augmenting path algorithm as used in FlowMap [1], which takes linear time in terms of the number of edges. In he worst case the mapping for $\Theta(n)$ nodes must be computed, so the complexity is $O(mn)$. This gives

Theorem 3 *For* $K = 2$*, the delay-optimal duplication-free mapping problem under the nominal delay model can be solved in* $O(mn)$ *time, where* m *and* n *are the number of edges and nodes in the network, respectively.* ∎

4 Considering Nominal Delay in Mapping

In this section, we present a simple heuristic to consider nominal delay in LUT mapping for delay minimization. We use the dynamic programming approach similar to that used in FlowMap [1]. We compute the delay-minimal mapping of each node according to a topological ordering of nodes starting from the PIs. The computation of delay-minimal mapping LUT_v of node v depends on the the delay-minimal mapping of predecessors of v computed in the previous

[1] A cut $(X, \overline{X})$ of size one can be eliminated in a pre-processing step by collapsing the nodes in $\overline{X}$ into the root of the cone.

[3] The *volume* of a cut $(X, \overline{X})$ is defined to be $|\overline{X}|$.

steps. After we have computed the delay-minimal mapping LUT_v of v, we also record the delay to the output of LUT_v (excluding the fanout delay of LUT_v) in the mapping solution, and denote it as $l(u)$.

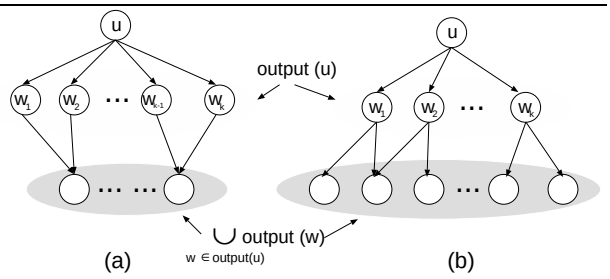

Figure 4: Nominal delay estimation.
(a) $|\bigcup_{w \in output(u)} output(w)| \leq |output(u)|$;
(b) $|\bigcup_{w \in output(u)} output(w)| \geq |output(u)|$.

When nominal delay is concerned, as we pointed out in Section 2, the mapping of node v may depend on the mapping of some nodes outside of fanin cone N_v of v, which may have not been processed (since they may appear later in the topological ordering). This is due to the fact that the fanout net delay of the nodes in N_v may not be known. In order to solve this problem, we *estimate* the fanout net delays of each node u in N_v by

$$D_N(u) = d_N \cdot |output(u)| + \alpha \cdot (|\bigcup_{w \in output(u)} output(w)| - |output(u)|), \quad (2)$$

where α is a positive constant for adjusting the relative weight of the second term. The first term depends on the fanout size of u. If it is large, it is likely that LUT_u will also have large fanout size, thus, large nominal delay. The second term is a correction of the simple estimation by the first term. In the second term, $|\bigcup_{w \in output(u)} output(w)|$ is the total fanout size of the fanouts of u. If it is smaller than $|output(u)|$, the fanouts of fanouts of u converges (see Figure 4(a)), thus some fanouts of u may be packed together in one or more LUTs. Thus, the nominal delay of LUT_u will decrease. On the other hand, if $|\bigcup_{w \in output(u)} output(w)|$ is larger, the fanouts of fanouts of u diverge (see Figure 4(b)). In this case, some fanouts of u are likely to be duplicated. Thus, the nominal delay of LUT_u will increase. Note that the value of $D_N(u)$ may be corrected later on when partial mapping solution is known.

Given the fanout delay estimation $D_N(u)$ of every node u in N_v and the delay to the fanout net of LUT_u in the delay-minimal mapping solution, we can compute the delay-minimal mapping of node v as follows. We compute a K-feasible cut $(X_v, \overline{X}_v)$ such that the *height* of this cut, defined as

$$H(X_v, \overline{X}_v) = \max\{l(u) + D_N(u) \mid u \in X_v\}, \quad (3)$$

is minimum. Note that $H(X_v, \overline{X}_v)$ corresponds to the maximum delay to the input of $\overline{X}_v$ (which will be the LUT implementing v). Such a cut can be computed using the minimum-height K-feasible cut algorithm introduced in FlowMap [1].

The minimum-height K-feasible cut algorithm was originally used in FlowMap for depth-optimal mapping under the unit delay model. The minimum-height K-feasible cut at each node can be found in $O(Km)$ time, where m is the number of edges in the network. Later on, the algorithm was generalized to delay-minimum mapping when each net has a fixed pre-assigned delay of arbitrary value [4]. The generalized minimum-height K-feasible cut algorithm runs in $O(Km \log n)$ time, where m and n are the number of edges and nodes in the network, respectively. We use the generalized minimum-height K-feasible cut algorithm in our mapping of node v and use $D_N(u)$ as the fanout delay estimation of each node in N_v. After we find the minimum-height K-feasible cut $(X_v, \overline{X}_v)$ in N_v, we set $LUT_v = \overline{X}_v$. For some node u in N_v, it may have two or more fanouts packed into a single input to LUT_v. Therefore, we update the fanout net delay estimation of every node in $input(LUT_v)$. Finally, we compute the delay to the output of LUT_v in the mapping solution and record it in $l(v)$.

After we have computed delay-minimal mapping of each node, we shall generate all the necessary LUTs starting from the POs using the same procedure as in FlowMap, which can be done in $O(m)$ time. Since we need to compute a minimum-height K-feasible cut at each node, the time complexity of our algorithm is $O(Kmn \log n)$. Since the time complexity is bounded by a low order polynomial, this algorithm can handle circuits of large sizes very efficiently, and can be invoked several times for the same design with different choices of α in the $D_N(u)$ estimation to obtain the best result.

5 Experimental Results

We have implemented the heuristic algorithm described in the preceding section and tested it on a set of six mid-size MCNC benchmark circuits which were also used in [1]. These circuits were chosen because each of them can be mapped into a single Xilinx XC3000 series FPGA chip [10], and be placed and routed using Xilinx design tools, so that we were able to measure the real delays of the mapping solutions (the second entry in Table 3 shows the type of Xilinx XC3000 chip used for each example). The circuits have been optimized for delay minimization using technology independent synthesis and decomposed into two-input simple gate networks. Each circuit was mapped using the heuristic algorithm into 5-LUTs. Then, we packed the LUTs into XC3000 series 2-output CLBs, whenever possible, using a maximum matching algorithm. Finally, we used the Xilinx apr program to place and route the mapping solutions, and used the Xilinx xdelay program to measure the actual delays. We selected the smallest possible FPGA chips that

circuit	XC3000	FlowMap			new heuristic			
name	part#	#clb	ndly	adly (*ns*)	α	#clb	ndly	adly (*ns*)
9sym	3020PC68	50	8.8	96.3	0.06	50	8.7	94.3
C880	3090PQ208	166	15.7	209.1	0.06	195	14.3	208.3
alu2	3064PC84	120	17.3	204.3	0.03	149	17.1	200.1
apex7	3042PP132	66	8.7	99.4	0.02	65	6.3	93.2
count	3020PC68	59	7.5	83.5	0.02	60	7.1	79.8
vg2	3020PC68	34	5.9	83.0	0.03	39	5.1	78.1

Table 3: Experimental results.
(ndly = nominal delay, adly = actual delay after routing. For nominal delay, $d_L = 1, d_N = 0.1$)

can accommodate the circuits and complete the routing. For comparison, we also mapped the circuits using FlowMap. The same FPGA chips were used, although FlowMap often uses fewer CLBs and sometimes can fit into smaller chips. We set $d_N = 0.1d_L$ in Eq.(2) in our nominal delay formulation. For each circuit we tried 10 different α values ($\alpha = 0.1d_N, ..., 1.0d_N$), and kept the one that produced the solution with the smallest nominal delay for placement and routing. Table 3 summarized these results. As one can see, with proper choice of parameters in the delay estimation model, our simple heuristic based on nominal delay estimation can produce better mapping solutions than the depth-optimal mapping algorithm.

6 Conclusion

We have studied the LUT-based FPGA technology mapping problem for delay minimization under the nominal delay model, a very simple case of net structure based dynamic delay model. Contrary to the fact that the depth minimization problem in LUT mapping is polynomial time solvable, we have shown that the nominal delay minimization problem is NP-hard for general LUT mapping when $K \geq 3$, and remains NP-hard for duplication-free mapping and tree-based mapping when $K \geq 5$ (but is polynomial time solvable for $K = 2$). Despite such difficulty, it is still beneficial to consider nominal delay during LUT mapping for delay minimization. We have demonstrated this using a simple heuristic.

Accurate delay modeling is very important in FPGA technology mapping. Our complexity results shown in this paper indicates that dynamic delay model is difficult to use directly. Currently we are working on more effective static approximation of dynamic delay minimization. An alternative to dynamic delay minimization is iterative static delay minimization via feedback from placement and routing, which we are also working on.

References

[1] J. Cong and Y. Ding, "An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *IEEE Trans. on CAD*, Vol. 13, No. 1, Jan. 1994, 1–12.

[2] 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, June 1994, 137–148.

[3] J. Cong and Y. Ding, "On Nominal Delay Minimization in LUT-Based FPGA Technology Mapping," *Tech. Report 940022*, Computer Science Department, UCLA, USA, May 1994; to appear in *INTEGRATION the VLSI Journal*.

[4] J. Cong, Y. Ding, T. Gao, and K. C. Chen, "An Optimal Performance-Driven Technology Mapping Algorithm for LUT based FPGAs under Arbitrary Net-Delay Models," *Proc. Int'l Conf. on CAD and Computer Graphics*, Beijing, China, Aug. 1993, 599–603.

[5] A. Farrahi and M. Sarrafzadeh, "Complexity of the Lookup-Table Minimization Problem for FPGA Technology Mapping," *IEEE Trans. on CAD*, Vol. 13, No. 11, Nov. 1994, 1319–1332.

[6] R. J. Francis, J. Rose, and Z. Vranesic, "Fast Technology Mapping for Lookup Table-Based FPGAs," *Proc. 28th ACM/IEEE Design Automation Conference*, San Francisco, CA, USA, June 1991, 613–619.

[7] M. Garey and D. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness*, Freeman and Company, San Francisco, CA, USA, 1979.

[8] D. Hill, "A CAD System for the Design of Field Programmable Gate Arrays," *Proc. 28th ACM/IEEE Design Automation Conference*, San Francisco, CA, USA, June 1991, 187–192.

[9] M. Schlag, 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*, Oxford, UK, Sept. 1991, 201–213.

[10] *The Programmable Gate Array Data Book*, Xilinx, Inc., San Jose, CA, USA, 1992.