Delay-Oriented Technology Mapping for Heterogeneous FPGAs with Bounded Resources

Jason Cong and Songjie Xu
Computer Science Department
University of California, Los Angeles, CA 90095
cong@cs.ucla.edu sxu@cs.ucla.edu
http://cadlab.cs.ucla.edu/~cong http://cadlab.cs.ucla.edu/~sxu

Abstract

In order to maximize performance and device utilization, recent generation of FPGAs take advantage of speed and density benefits resulted from *heterogeneous* FPGAs, which provide either an array of homogeneous programmable logic blocks (PLBs), each configured to implement circuits with LUTs of different sizes, or an array of physically heterogeneous LUTs. Some heterogeneous FPGAs do not have limitations on the availability of LUTs of specific sizes within chip capacity due to the configuration flexibility of their PLBs, while others, such as Altera FLEX10K devices [1] and Vantis VF1 FPGAs [11], have limited number of LUTs of certain types (such as embedded memory blocks), which we call *heterogeneous* FPGAs with *bounded* resources. LUTs of different sizes usually have different delays. In this paper, we study the technology mapping problem for delay minimization for *heterogeneous* FPGAs with *bounded* resources. We show that it is NP-Hard for general networks, but can be solved optimally in pseudo-polynomial time for trees. We then present two heuristic algorithms, named BinaryHM and CN-HM, for delay minimization of general networks for *heterogeneous* FPGA designs with *bounded* resources. We have tested BinaryHM and CN-HM on MCNC benchmarks on Altera FLEX10K device family, which can be taken as the heterogeneous FPGAs with 4-LUTs and a limited number of 11-LUTs. The experimental results show that compared with FlowMap using only 4-LUTs, both BinaryHM and CN-HM can reduce more than 20% of the circuit mapping delays, 27% of the 4-LUT area and 10% of the circuit layout delays by making efficient use of the *available* heterogeneous LUTs.

1 Introduction

In a traditional LUT-based FPGA device, the basic programmable logic block is a K-input lookup table (K-LUT) which can implement any Boolean function of up to K variables. In order to maximize performance and device utilization, recent generation of FPGAs take advantage of speed and density benefits resulted from *heterogeneous* FPGAs, which provide either an array of homogeneous programmable logic blocks (PLBs), each configured to implement circuits with LUTs of different sizes, or an array of physically heterogeneous LUTs. For example, the PLBs in Xilinx XC4000 series FPGAs [13], Lucent ORCA2C series FPGAs [10] and the recently announced Vantis VF1 FPGAs [11][1] can be configured to have heterogeneous LUTs. These heterogeneous FPGAs do not have limitations on the availability of LUTs of specific sizes within chip capacity due to their PLB configuration flexibility. On the other hand, Altera FLEX10K devices [1] (see Figure 1) and Vantis VF1 FPGAs (see Figure 2) provide both a logic array of normal K-LUTs and an embedded memory array with a series of embedded memory blocks (EMBs) which, if not used as on-chip memories, can be used to implement logic functions. These heterogeneous FPGAs have limitation on one or several types of LUTs, which we call *heterogeneous* FPGAs with *bounded* resources. For example, in one FLEX10K device chip, there are 3 to 12 EMBs[2] according to the device size, and each EMB can be taken as an 11-LUT. In one VF1 FPGA chip, there are 28 to 48 EMBs according to the device size, and each EMB can be configured to implement any single logic function of 7 inputs and 1 output.

In a heterogeneous FPGA, larger LUTs can cover more gates, but usually have longer delay. Therefore, given a *heterogeneous* FPGA with *bounded* resources, an important problem is how to utilize the *available* heterogeneous LUTs to minimize the overall circuit delay and/or area during technology mapping.

In the past a few years, extensive studies have been done on technology mapping for homogeneous LUT-based FPGAs. A survey of these results can be found in [4]. However, none of these algorithms are able to deal with the delay optimization problem for heterogeneous FPGAs, as they assume the identical capacity and delay for every LUT. In

[1] In XC4000, ORCA2C, and VF1 FPGAs, *PLB* is called *configurable logic block (CLB)*, *programmable function unit (PFU)*, and *configurable building block (CBB)* respectively.

[2] In Altera FLEX10K device family, *EMB* is called *embedded array block (EAB)*.

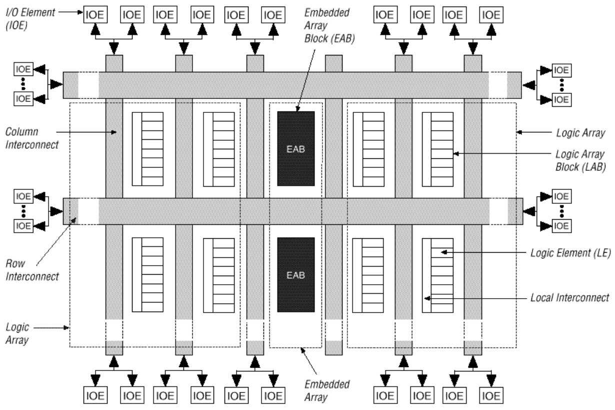

Figure 1: Altera FLEX10K Device Block diagram.

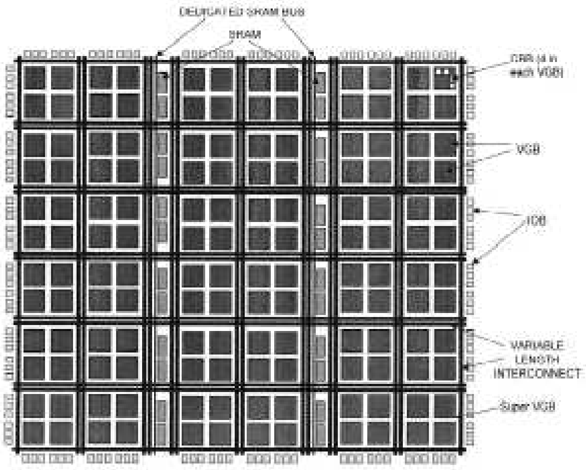

Figure 2: Vantis VF1 Family Architecture.

[8], an approach for technology mapping into heterogeneous LUT-based FPGAs was presented for area minimization, but their architecture assumes a mixture of only two types of LUTs with a fixed ratio in one FPGA chip. The recent work in [9] shows that the problem of mapping a tree network using the minimum number of target PLBs, each with independent LUTs of two different sizes, can be solved optimally in $O(n^3)$ time. However, the optimality holds only for trees, which significantly limits the application of this algorithm. In [5] and [12], it was shown that uncommitted EMBs in heterogeneous FPGAs can be efficiently used to implement logic for area minimization. The algorithm in [5] can further guarantee that the circuit delay will not increase while using available EMBs for area reduction. In [6], the first polynomial-time delay-optimal technology mapping algorithm, named HeteroMap, was presented for heterogeneous FPGA designs, and the optimality of the HeteroMap algorithm holds for general networks. However, HeteroMap cannot be applied to *heterogeneous* FPGAs with *bounded* resources directly, since HeteroMap cannot constrain the use of certain types of LUTs.

In this paper, we formulate a general technology mapping problem for *heterogeneous* FPGAs with *bounded* resources and show that this problem under delay minimization objective is NP-Hard for general networks, but solvable for trees in pseudo-polynomial time. We then present two heuristic algorithms, named BinaryHM and CN-HM, for delay minimization of general networks in *heterogeneous* FPGA designs with *bounded* resources. Both BinaryHM and CN-HM produce favorable results on MCNC benchmarks on Altera FLEX10K device family [1], which can be taken as the heterogeneous FPGAs with 4-LUTs and a limited number of 11-LUTs.

The remainder of this paper is organized as follows. Section 2 presents the problem formulation and preliminaries. Section 3 gives the complexity result for delay optimal mapping in *heterogeneous* FPGA designs with *bounded* resources. In order to solve the problem, Section 4 presents a pseudo-polynomial time optimal algorithm for trees for our problem, while Section 5 presents two heuristic algorithms for general networks. Experimental results and comparative study are presented in Section 6. Section 7 concludes the paper.

2 Problem Formulation and Preliminaries

A Boolean network N 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. Primary input (PI) nodes have no incoming edge and primary output (PO) nodes have no outgoing edge. We use $input(v)$ to denote the set of fanins of gate v. A boolean network is *K-bounded* if $|input(v)| \leq K$ for each node v in the network.

A LUT-based *heterogeneous* FPGA with *bounded* resources consists of c types of LUTs of K_1-LUT, K_2-LUT, $\ldots$, and K_c-LUT ($K_1 < K_2 < \ldots < K_c$), with the delays d_1, d_2, $\ldots$, and d_c ($d_1 < d_2 < \ldots d_c$, and d_1, d_2, $\ldots$, d_c may not be integer), and with resource bound B_1, B_2, $\ldots$, and B_c for each type of LUT, respectively. Without loss of generality, d_1 is scaled to 1 in remaining discussions. Some of B_i's can be ∞. Homogeneous FPGAs can be viewed as the special heterogeneous FPGA with only one type of LUTs.

For a circuit mapped into a heterogeneous FPGA, we assume different access delays for heterogeneous LUTs but a constant delay for the interconnection[3], which is called *het-*

[3]The constant interconnection delay can be counted into the LUT delays such that the interconnection delay can be set to zero. In general, interconnection delays are highly dependent on the placement result. We choose to consider interconnection delay as constant as there is no good delay model available so far which is able to accurately estimate interconnection delay in the logic synthesis phase. See

erogeneous LUT-delay model [6]. The unit-delay model used in [2] is a special case of heterogeneous LUT-delay model in homogeneous FPGAs.

Given these definitions, the technology mapping problem for *heterogeneous* FPGAs with *bounded* resources can be formulated as follows: *Given a K_1-bounded Boolean network N and the heterogeneous FPGA with bounded resources, transform N to an equivalent LUT network N' by making use of the available heterogeneous LUT resources such that the circuit delay and/or area are minimized.*

In this paper, our primary objective is to minimize the circuit mapping delay under the heterogeneous LUT-delay model through technology mapping. Therefore, a mapping solution is said to be *optimal* if the mapping delay is minimized. The corresponding technology mapping problem for delay minimization for *heterogeneous* FPGAs with *bounded* resources is abbreviated as the **DM-HM-BR** Problem.

3 Complexity of Problem DM-HM-BR

In this section, we shall investigate the computational complexity of the **DM-HM-BR** problem. In order to simplify the description, we assume that there are two types of LUTs in a heterogeneous FPGA, K_1-LUT without resource limitation and r K_2-LUT ($K_1 < K_2$, r is a variable), with delay ratio of $1 : d$ ($d > 1$). We first define the decision version of the **DM-HM-BR** problem.

Problem: Delay-Bounded heterogeneous LUT mapping with bounded resources (**DB-HM-BR**)

Instance: Three integers, K_1, K_2 ($K_1 < K_2$) and r ($r \geq 0$), two real numbers, d ($d > 1$) and B, and a K_1-bounded Boolean network N.

Question: Under the heterogeneous LUT-delay model with $d_1 = 1$ and $d_2 = d$, is there a mapping solution of N with any number of K_1-LUTs and no more than r K_2-LUTs, which has delay no more than B?

We shall show that the **DB-HM-BR** problem is NP-complete for $K_1 \geq 5$. The proof of the NP-completeness for the **DB-HM-BR** problem is based on the polynomial time transformation from the 3-Satisfiability (**3SAT**) problem, a well-known NP-complete problem, to the **DB-HM-BR** problem. First, we define the **3SAT** problem,

Problem: 3-Satisfiability (**3SAT**)

Instance: A set of Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$ and a collection of m clauses $C = \{C_1, C_2, \ldots, C_m\}$, where (i) each clause is the disjunction (OR) of 3 literals of the variables and (ii) each clause contains at most one of x_i and $\bar{x}_i$ for any variable x_i in X.

Question: Is there a truth assignment of the variables in X such that $C_j = 1$ ($j = 1, 2, \ldots, m$)?

Section 7 for more discussions.

We shall construct a polynomial time transformation that transforms each instance of **3SAT** to an instance of **DB-HM-BR**. We shall relate the decision of the truth assignment of a Boolean variable in an instance $\mathcal{F}$ of **3SAT** to the decision of the K_2-LUT implementation on a node in the corresponding network N of the **DB-HM-BR** instance. Since determining the truth assignment is difficult, we can show that determining the K_2-LUT implementation is also difficult.

Given an instance $\mathcal{F}$ of the **3SAT** with r variables x_1, x_2, ..., x_r and m clauses C_1, C_2, ..., C_m, we construct a K_1-bounded Boolean network N corresponding to the instance $\mathcal{F}$ as follows.

Let $d' = \lfloor d \rfloor$. First, for each variable x_i, we construct a subnetwork $N(x_i)$ that consists of the following nodes:

- $d' + 3$ internal nodes, denoted as x_i, $\bar{x}_i$, a_i^1, a_i^2, ..., $a_i^{d'}$, and b_i;
- $(3 \cdot K_1 - 1)$ PI nodes;
- One PO node, denoted as O_{x_i}.

The nodes are connected as shown in Figure 3, where PI_0, a_i^1, a_i^2, ..., $a_i^{d'}$, x_i, $\bar{x}_i$, b_i and O_{x_i} form a direct chain. Every internal node in $N(x_i)$ and PO O_{x_i} have K_1 fanins. Let $S = \{a_i^1, a_i^2, \ldots, a_i^{d'}, x_i, \bar{x}_i\}$, then $\forall s_1, s_2 \in S$ ($s_1 \in input(s_2)$), $|input(s_1)| = K_1$, $|input(s_2)| = K_1$, and $|input(s_1) \cup (input(s_2) - \{s_1\})| = K_1 + 1 > K_1$. Therefore, no packing can be done among a_i^1, a_i^2, ..., $a_i^{d'}$, x_i, $\bar{x}_i$, b_i and O_{x_i} with K_1-LUTs. The outputs of *variable nodes* x_i and $\bar{x}_i$ are connected to the corresponding *literal nodes* in the clause subnetworks to be constructed.

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:

- Four internal nodes, denoted as l_j^1, l_j^2, l_j^3 and e_j;
- $(3 \cdot \lfloor \frac{K_1-1}{2} - 1 \rfloor + K_1 - 1)$ PI nodes (K_1 is odd and $K_1 \geq 5$, or K_1 is even and $K_1 \geq 8$) or $(3 \cdot \lfloor \frac{K_1-1}{2} - 1 \rfloor + K_1)$ PI nodes ($K_1 = 6$);
- One PO node, denoted as O_{c_j}.

The nodes are connected as shown in Figure 4, where e_j and $(K_1 - 1)$ PI nodes form the fanin set of O_{c_j}, and the three literal nodes, l_j^1, l_j^2, and l_j^3 form the fanin set of e_j. For each literal node, there is one fanin coming from the corresponding variable node in the variable network. For instance, if $l_j^1 = x_{i_1}$ ($1 \leq i_1 \leq r$), node x_{i_1} in $N(x_{i_1})$ will be the fanin of l_j^1. For each literal node, there are another $\lfloor \frac{K_1-1}{2} - 1 \rfloor$ fanins formed by PI nodes. The subnetwork $N(c_j)$ in the case that K_1 is odd and $K_1 \geq 5$, or K_1 is even and $K_1 \geq 8$ is illustrated in Figure 4(a). For the case of $K_1 = 6$, another PI is added as another fanin to one of the literal nodes, *i.e.* l_j^1, as depicted in Figure 4(b).

After connecting subnetworks $N(c_j)$, $j = 1, 2, \ldots, m$, with the subnetworks $N(x_i)$ ($i = 1, 2, \ldots, r$), the network

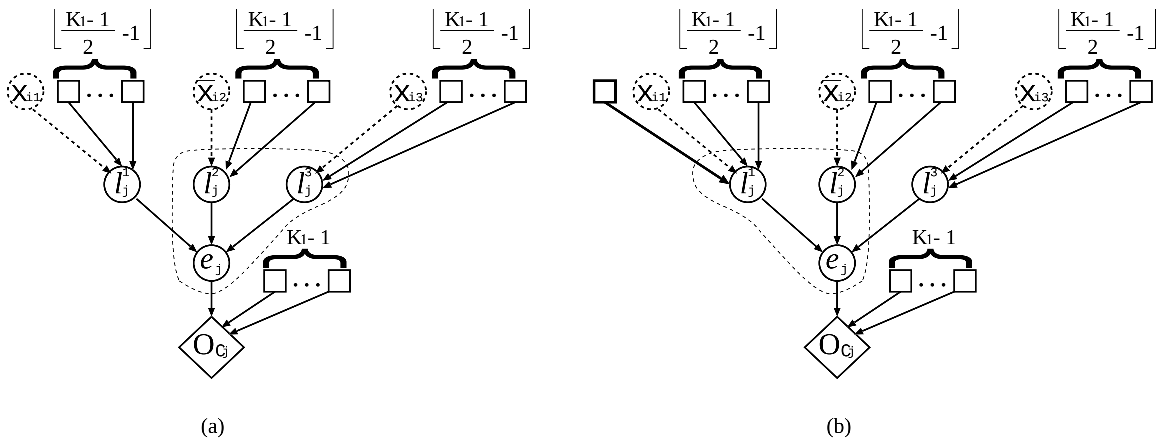

Figure 4: Transformation of **3SAT** instance to **DB-HM-BR** instance: Construction of $N(C_j)$ for variable C_j and the possible packing operation on e_j: (a) When K_1 is odd and $K_1 \geq 5$, or K_1 is even and $K_1 \geq 8$; (b) When $K_1 = 6$.

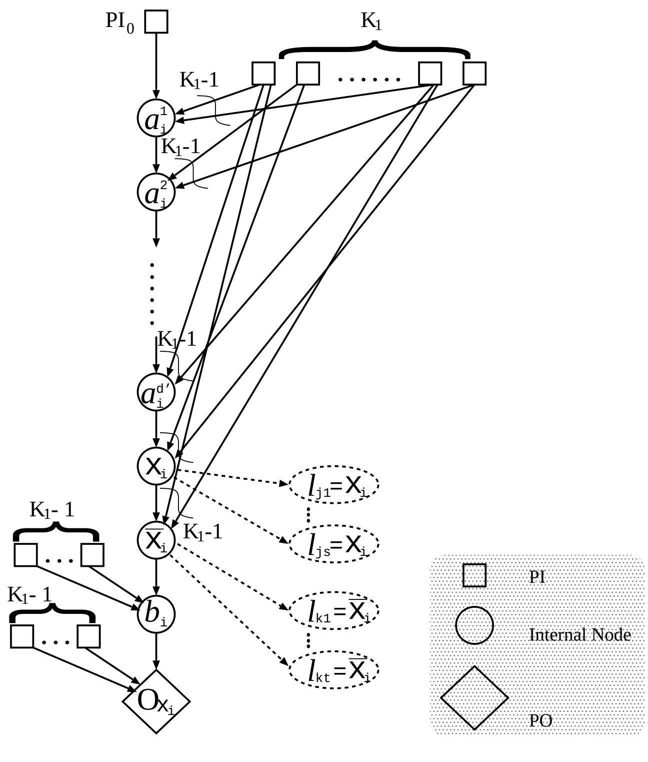

Figure 3: Transformation of **3SAT** instance to **DB-HM-BR** instance: Construction of $N(x_i)$ for variable x_i.

N obtained is K_1-bounded when $K_1 \geq 3$. It is clear that the transformation defined above takes $O(K_1 \cdot (r \cdot d' + m))$ time. The following NP-completeness proof remains valid for $K_1 \geq 5$.

We define packing a_i^1, a_i^2, $\ldots$, and $a_i^{d'}$ into x_i to form a K_2-LUT as $pack\_K_2(x_i)$, and packing a_i^1, a_i^2, $\ldots$, $a_i^{d'}$ and x_i into $\bar{x}_i$ to form a K_2-LUT as $pack\_K_2(\bar{x}_i)$. We now show that it is impossible to pack all the three literal nodes in $N(C_j)$ to e_j using a K_1-LUT. If K_1 is an odd number, each of the literal nodes has $\frac{K_1-1}{2}$ fanins. If all the three literal nodes are packed into e_j, the total number of fanins for e_j is $\frac{3\cdot(K_1-1)}{2}$, which is larger than K_1 when $K_1 \geq 5$. If K_2 is an even number, each of the literal nodes has $\frac{K_1}{2} - 1$ fanins. If all the three literal nodes are packed into e_j, the total number of fanins for e_j is $\frac{3\cdot K_1}{2} - 3$, which is larger than K_1 when $K_1 \geq 8$. Similarly we can show that with the subnetwork constructed for $K_1 = 6$ in Figure 4(b), it is impossible to pack all the three literal nodes in $N(C_j)$ to e_j. Therefore, it is impossible to pack all the three literal nodes in $N(C_j)$ to e_j using a K_1-LUT for $K_1 \geq 5$. However, we can always pack two of the three literal nodes in $N(C_j)$ to e_j to form a K_1-LUT, denoted as $pack\_K_1(e_j)$ (see Figure 4).

By linking the variable assignment of $x_i = 1$ for the **3SAT** instance $\mathcal{F}$ with $pack\_K_2(x_i)$, and linking the assignment of $x_i = 0$ with $pack\_K_2(\bar{x}_i)$, we can show

Theorem 1 *$\mathcal{F}$ is satisfiable if and only if N has a mapping solution of delay $d+3$ using K_1-LUTs and no more than r K_2-LUTs.*

The proof is given in Appendix.

Corollary 1 DB-HM-BR *is NP-complete for $K_1 \geq 5$.*

Proof: **DB-HM-BR** is in NP because given any mapping solution with K_1-LUTs and r K_2-LUTs, we can easily verify if its delay is bounded by B. Moreover, the transformation from **3SAT** instance to **DB-HM-BR** instance takes polynomial time. Finally, according to Theorem 1, the **3SAT** question has YES answer if and only if the **DB-HM-BR** question has YES answer. Therefore, the NP-completeness of **3SAT** implies that **DB-HM-BR** is NP-complete. ■

Corollary 2 *The* **DM-HM-BR** *problem is NP-hard for $K_1 \geq 5$.*

Note that the construction of the polynomial time transformation does not apply when $K_1 \leq 4$. Therefore, the complexity of the problem is still open for $K_1 \leq 4$.

4 Delay Optimal Mapping for Trees

Although the **DM-HM-BR** problem is NP-hard for general DAGs, we shall show in this section that it can be solved optimally in pseudo-polynomial time for trees using the dynamic programming technique.

Assume that there are two types of LUTs in a heterogeneous FPGA, K_1-LUT without resource limitation and r K_2-LUTs ($K_1 < K_2$), with delay ratio of $1 : d$ ($d > 1$). Given a tree T, we want to compute the mapping solution for T with minimum mapping delay under the heterogeneous LUT-delay model by using the *available* heterogeneous LUT resources. The algorithm is based on the cut generation for trees. Assume that the root t of T has f fanin nodes v_1, v_2, ..., v_f ($f \leq K_1$). Let T_{v_i} denote the subtree in T rooted at v_i ($1 \leq i \leq f$). Clearly, any cut of size H in T induces an H_i-cut of T_{v_i}, with $\sum_{i=1}^{f} H_i = H$, and vice versa. Let $C_T(H)$ denote the set of cuts of size H in T, and define $C_T(1) = t$, [3] showed

$$C_T(H) = \bigcup_{\sum_{i=1}^{f} H_i = H} (C_{T_{v_1}}(H_1) \times C_{T_{v_2}}(H_2) \times \ldots \times C_{T_{v_f}}(H_f)) \tag{1}$$

It was shown in [3] that based on the recursive equation 1, all the cuts of size H in a tree can be generated, and the number of cuts generated according to this equation is bounded by a constant depending only on H, which is the $(H-1)$th *Catalan number* [7], denoted c_{H-1}, where $c_H = \frac{1}{H+1}\binom{2H}{H}$. The total number of H-*feasible* cuts in a tree is thus bounded by $\sum_{i=0}^{H-1} c_i$.

Using dynamic programming, for each node v in T from leaves to root t in topological order, we want to compute $l_v(p)$ for each $p = 0, 1, \ldots, r$, which is the minimum delay of node v with p K_2-LUTs used in the mapping solution of T_v. The topological order guarantees that every node is processed after all of its predecessors have been processed. For each node v, we first generate all K_2-*feasible* cuts in T_v, which include all K_1-*feasible* cuts in T_v as well. Note that node v will be implemented either by K_1-LUT or K_2-LUT. We first assume that v is implemented by K_1-LUT. For each K_1-*feasible* cut c in T_v, we enumerate all the p K_2-LUTs' distributions among the trees rooted at the cut nodes $v_1, v_2, \ldots, v_s$ ($s \leq K_1$) of this K_1-*feasible* cut c, then obtain the minimum delay of v using this K_1-*feasible* cut by the following formular

$$l_{1c_v}(p) = \min_{\sum_{j=1}^{s} p_j = p} \{\max_{1 \leq j \leq s} l_{v_j}(p_j) + 1\} \tag{2}$$

Through checking all the K_1-*feasible* cuts in T_v, we can compute $l_{1v}(p)$ for each $p = 0, 1, \ldots, r$, which is the minimum delay of node v with p K_2-LUTs used in the mapping solution of T_v if v is implemented by an K_1-LUT. In a similar way, by assuming that v is implemented by K_2-LUT, we check each K_2-*feasible* cut c in T_v and enumerate all the $p-1$ K_2-LUTs' distributions among the trees rooted at the cut nodes $v_1, v_2, \ldots, v_s$ ($s \leq K_2$) of this K_2-*feasible* cut c, then obtain the minimum delay of v using this K_2-*feasible* cut by the following formular

$$l_{2c_v}(p) = \min_{\sum_{j=1}^{s} p_j = p-1} \{\max_{1 \leq j \leq s} l_{v_j}(p_j) + d\} \tag{3}$$

Through checking all the K_2-*feasible* cuts in T_v, we can compute $l_{2v}(p)$ for each $p = 0, 1, \ldots, r$, which is the minimum delay of node v with p K_2-LUTs used in the mapping solution of T_v if v is implemented by an K_2-LUT. Therefore, $l_v(p) = \min\{l_{1v}(p), l_{2v}(p)\}$ for each $p = 0, 1, \ldots, r$. For the root t, $l_t(r)$ gives the minimum mapping delay of T using K_1-LUTs and no more than r K_2-LUTs.

Since the cut generation takes $O(\sum_{i=0}^{K_2-1} c_i)$ time, where c_i is the ith *Catalan number*, and the p K_2-LUTs' distribution among the trees rooted at the cut nodes of each K_j-*feasible* cut ($1 \leq j \leq 2$) of T_v takes $O(p^{K_j})$ time, the complexity of the above algorithm is $O(n \cdot \sum_{i=0}^{K_2-1} c_i \cdot r \cdot (r^{K_1} + r^{K_2}))$, where n is the number of nodes in T.

It is not hard to see that this algorithm can be easily extended for heterogeneous FPGAs with c types of LUTs ($c > 2$), q of which have resource limitations, specified by r_1 K_1-LUTs, r_2 K_2-LUTs and r_q K_q-LUTs respectively. We shall then compute $l_v(p_1, p_2, \ldots, p_q)$ ($0 \leq p_1 \leq r_1$, $0 \leq p_2 \leq r_2$, ..., $0 \leq p_q \leq r_q$), for each node v in tree T, instead of $l_v(p)$ ($0 \leq p \leq r$), and the complexity becomes $O(n \cdot \sum_{i=0}^{K_w-1} c_i \cdot \prod_{j=1}^{q} r_j \cdot \sum_{i=1}^{c} \prod_{j=1}^{q} {r_j}^{K_i})$, where $K_w = \max_{1 \leq i \leq c} K_i$. This algorithm is considered to be polynomial if the sizes of the heterogeneous LUTs are taken as the constants. Therefore, the **DM-HM-BR** problem can be solved optimally in pseudo-polynomial time for trees.

5 Delay-Oriented Mapping for DAGs

In Section 4, we showed that the **DM-HM-BR** problem can be solved optimally for trees in polynomial time. However, most of the combinational circuits are general DAGs instead of trees. If we decompose the general DAG into independent trees before mapping, in order to solve the **DM-HM-BR** problem, we have to try all possible distributions of the LUTs with bounded resources among the independent trees, which will result in very high complexity. The final mapping solution will not be optimal either, since no LUT can go across trees. Therefore, we do not intend to use the optimal tree mapping algorithm for general DAGs. Instead, we developed two efficient heuristics to solve this problem, which will be presented in the following subsections. We start with a brief review of the HeteroMap algorithm presented in [6], which will be used in our two heuristics to compute the delay-optimal heterogeneous LUT mapping solution without resource constraints under the given heterogeneous LUT delay ratio.

5.1 Review of the HeteroMap algorithm

HeteroMap [6] is a technology mapping algorithm that computes delay-optimal mapping solutions in polynomial time for heterogeneous FPGAs without resource constraints. Taking different delays of heterogeneous LUTs into consideration, HeteroMap computes the minimum mapping delay of a circuit based on a series of minimum height K-feasible cut computations at each node in the circuit. For a heterogeneous FPGA consisting of K_1-LUTs, K_2-LUTs, ..., and K_c-LUTs, HeteroMap computes the minimum delay mapping solution in $O(\sum_{i=1}^{c} K_i \cdot n \cdot m \cdot \log n)$ time for a circuit netlist with n gates and m edges.

5.2 The BinaryHM algorithm

In this section, we shall present our first heuristic algorithm, named BinaryHM, to solve the **DM-HM-BR** problem. Assume that there are two types of LUTs in a heterogeneous FPGA, K_1-LUT without resource limitation and r K_2-LUT ($K_1 < K_2$), with delay ratio of $1 : d$ ($d > 1$). For a Boolean network N, although computing its minimum mapping delay is an NP-hard problem, as shown in Section 3, we can determine the lower and upper bounds of the minimum mapping delay, denoted $D_{MM}(N)$, as follows: let $D_{FM}(N)$ be the minimum mapping delay obtained by FlowMap using only K_1-LUTs, and $D_{HM}(N)$ be the minimum mapping delay obtained from HeteroMap using K_1-LUTs and K_2-LUTs, then

$$D_{HM}(N) \leq D_{MM}(N) \leq D_{FM}(N) \quad (4)$$

$D_{FM}(N)$ is an upper bound of $D_{MM}(N)$ since FlowMap does not use any K_2-LUT. $D_{HM}(N)$ is a lower bound of $D_{MM}(N)$ with the delay ratio of d, as HeteroMap uses as many K_2-LUTs as *necessary* to achieve the minimum delay for each node in N. However, if we increase the delay ratio of K_1-LUT *vs.* K_2-LUT from the original delay ratio, d, all the way to $D_{FM}(N)$, HeteroMap will tend to use K_2-LUTs on the nodes which have more delay reduction with the K_2-LUT implementation over the K_1-LUT implementation, and we also expect to see that HeteroMap will generate mapping solutions using fewer and fewer K_2-LUTs. With $K_1 = 4$ and $K_2 = 11$, Figure 5 shows how the number of 11-LUTs used in the mapping solution of HeteroMap decreases with the increase on the delay ratio of an 11-LUT and a 4-LUT. When the delay ratio equals to $D_{FM}(N)$, HeteroMap will produce exactly the same mapping solution as FlowMap, as using any K_2-LUT will not lead to better mapping solution. Therefore, by doing binary search on the delay ratio of K_1-LUT *vs.* K_2-LUT with the range from the original delay ratio of d to $D_{FM}(N)$, BinaryHM will finally converge to a mapping solution, where no more than r K_2-LUTs are used and the mapping delay of N, whose range is defined by Eqn. 4, is minimized. For a netlist N with n nodes and m edges, the delay ratio range $(D_{FM}(N) - d)$ is at most n. If the granularity of the binary search over the delay ratio of K_1-LUT *vs.* K_2-LUT is selected to be g, BinaryHM will go through the HeteroMap algorithm for $\log \frac{n}{g}$ times. Therefore, the complexity of the BinaryHM algorithm is $O(\log \frac{n}{g} \cdot (K_1 + K_2) \cdot n \cdot m \cdot \log n)$. For the experimental results reported in Section 6, the value of g is set to be 0.1.

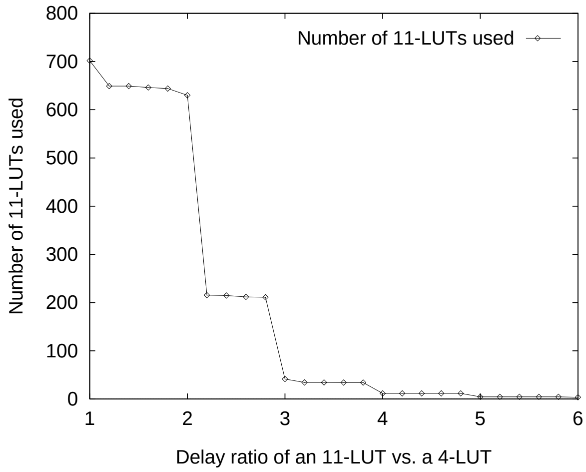

Figure 5: The number of 11-LUTs used by HeteroMap with resepect to different delay ratios between an 11-LUT and a 4-LUT.

5.3 The CN-HM algorithm

Our second heuristic, named CN-HM, is a post-mapping approach. Given an original unmapped network, FlowMap is first applied to map N into a K_1-LUT netlist N' of the minimum mapping delay, then in the post-mapping procedure, CN-HM intends to minimize the circuit delay by using no more than r K_2-LUTs. Similar to BinaryHM, we first determine a range of the minimum delay, denoted $D_{MM}(N')$, of N', part of which will be covered by no more than r K_2-LUTs. Let $D_{FM}(N')$ be the current mapping delay of N' with each node as a K_1-LUT. Let $D_{HM}(N')$ be the minimum mapping delay of N' if there is no constraint on the number of K_2-LUTs used on N'. $D_{HM}(N')$ is obtained by performing a labeling procedure, similar to that in HeteroMap [6], on N', except that for each node v in N', the possible K_1-LUT implementation on v is v itself since CN-HM is a post-mapping approach. Clearly, $D_{FM}(N')$ is the upper bound of $D_{MM}(N')$. $D_{HM}(N')$ is the lower bound of $D_{MM}(N')$, as $D_{HM}(N')$ is obtained by using as many K_2-LUTs as *necessary* to minimize the delay for each node in N'. Therefore, we have

$$D_{HM}(N') \leq D_{MM}(N') \leq D_{FM}(N') \quad (5)$$

With the range of the minimum delay defined by Eqn. 5, CN-HM performs binary search over this range to get the circuit

		FlowMap		HeteroMap			BinaryHM			CN-HM			CN-HM+EP	
Circuits	emb_a	Dly_m	4-LUT	Dly_m	4-LUT	emb_u	Dly_m	4-LUT	emb_u	Dly_m	4-LUT	emb_u	4-LUT	emb_u
alu2	3	11	192	4	3	3	4	3	3	4	3	3	3	3
apex4	8	7	1239	4	34	17	6	575	8	6	577	8	577	8
C5315	6	10	593	9	569	4	9	569	4	10	593	0	553	6
C7552	8	9	666	8	624	10	8	668	5	9	666	0	606	8
9sym	3	6	141	4	0	1	4	0	1	4	0	1	0	1
9symml	3	6	97	4	0	1	4	0	1	4	0	1	0	1
b12	3	6	225	4	50	2	4	50	2	4	49	2	29	3
clip	3	5	190	4	85	2	4	85	2	4	85	2	85	2
des	11	6	1579	5	719	64	6	1579	0	6	1579	0	1573	1
ex5p	8	7	1302	4	59	35	7	1302	0	7	1302	0	1283	4
rd73	3	5	150	4	33	2	4	33	2	4	33	2	0	1
rd84	3	6	294	4	3	3	4	3	3	4	3	3	3	3
sao2	3	5	90	4	64	1	4	64	1	4	64	1	64	1
TOTAL	65	89	6758	62	2243	145	68	4931	32	70	4954	23	4776	42
Ar_Mean	5	6.85	519.85	4.77	172.54	11.15	5.23	379.31	2.46	5.38	381.08	1.77	367.38	3.23
Ar_Ratio	N/A	1	1	-30%	-67%	1	-24%	-27%	-78%	-21%	-27%	-84%	-29%	-71%
Ge_Mean	4.39	6.62	324.83	4.57	0	4.15	4.99	0	0	5.08	0	0	0	2.39
Ge_Ratio	N/A	1	1	-31%	N/A	1	-25%	N/A	N/A	-23%	N/A	N/A	N/A	-42%

Table 1: Mapping Comparison among FlowMap, HeteroMap, BinaryHM and CN-HM on FLEX10K device family.

delay target D_{tgt} at each time. CN-HM then identifies all the *critical* nodes in N' whose delays have to be reduced in order to achieve the overall circuit delay target D_{tgt}. These *critical* nodes altogether form a *critical* graph G_c with these critical nodes as the vertices and their interconnections in N' (critical paths) as edges.

Observation 1 *N' has mapping delay of no more than D_{tgt} by using no more than r K_2-LUTs only if G_c has the minimum cut of size no more than r.*

CN-HM checks whether a delay target D_{tgt} is possible to be obtained or not by the necessary condition depicted in Observation 1 before any further operation is performed. If it is possible, CN-HM will perform HeteroMap labeling procedure on the *critical* nodes in N'. Then, similar to BinaryHM, CN-HM performs a binary search on the delay ratio of K_1-LUT *vs.* K_2-LUT with the range from the original delay ratio of d to D_{tgt}, and applies HeteroMap with the given delay ratio to check whether no more than r K_2-LUTs can be used in N' such that the delay of N' is bounded by D_{tgt}. For a mapped netlist N' with n' nodes and m' edges, the delay target range $(D_{FM}(N') - D_{HM}(N'))$ is at most n'. If the granularity of the binary search over the circuit delay target D_{tgt} is set to be g_1 and the granularity of the binary search over the delay ratio of K_1-LUT *vs.* K_2-LUT is set to be g_2, the complexity of the CN-HM algorithm will be $O(\log \frac{n'}{g_1} \cdot \log \frac{n'}{g_2} \cdot (K_1 + K_2) \cdot n' \cdot m' \cdot \log n')$. For the experimental results reported in Section 6, g_1 is set to be 1 and g_2 is set to be 0.1.

In summary, BinaryHM operates on the original unmapped circuit and performs binary search on the delay ratio of K_1-LUT *vs.* K_2-LUT such that HeteroMap can eventually obtain the mapping solution with feasible number of K_2-LUTs used and the circuit mapping delay minimized. Instead, CN-HM takes the mapped circuit with each node as a K_1-LUT and performs binary search on the minimum delay of the circuit to get a delay target at each time. For each delay target, CN-HM identifies *critical* nodes and again performs binary search on the delay ratio of K_1-LUT *vs.* K_2-LUT such that HeteroMap can check whether the delay target is feasible or not.

6 Experimental Results

We have implemented the BinaryHM algorithm and the CN-HM algorithm on SUN Ultra SPARC workstation. We tested BinaryHM and CN-HM on MCNC benchmarks on Altera FLEX10K device family [1], which can be taken as the heterogeneous FPGAs with 4-LUTs and a limited number of 11-LUTs. The technology mapping comparison results are summarized in Table 1, where the BinaryHM and CN-HM algorithms are compared with FlowMap [2] which only uses 4-LUTs, and HeteroMap [6] which uses both 4-LUTs and 11-LUTs and the number of 11-LUTs used by HeteroMap could extend the resource limitations. In FLEX10K devices, the delay ratio between 4-LUT and 11-LUT is 1 : 4. For BinaryHM and CN-HM, the number of 11-LUTs (EMBs) available ("emb_a") is determined by the smallest FLEX10K device into which this circuit can be fitted. The experimental results show that compared with FlowMap using only 4-LUTs, on average, both BinaryHM and CN-HM can reduce more than 20% of the circuit mapping delays ("Dly_m") and 27% of the 4-LUT area ("4-LUT") by making efficient use of the *available* heterogeneous LUTs. Moreover, in order to meet the resource constraints, BinaryHM and CN-HM consume much fewer 11-LUTs ("emb_u") than HeteroMap does. Although for some circuits, not all the available 11-LUTs are used for delay minimization in BinaryHM and CN-HM, they can be used later on by EMB_Pack, an algorithm proposed in [5], to further minimize the circuit area while maintaining the circuit delay. As an example, we also showed in the last two columns of Table 1 the circuit area and the number of EMBs used eventually by CN-HM followed by EMB_Pack ("CN-HM+EP") as the postprocessing.

The CPU time comparison is summarized in Table 2. From Table 1 and Table 2 we can see that both BinaryHM and CN-HM are fairly efficient, using less than 20 minutes of CPU time for all 13 benchmarks ranging from 200 gates to 3,000 gates (four of them have over 2,000 gates).

Circuits	FlowMap CPU (s)	HeteroMap CPU (s)	BinaryHM CPU (s)	CN-HM CPU (s)
alu2	1.20	52.80	159.30	20.10
apex4	10.70	13.70	18.50	12.80
C5315	5.40	15.50	602.70	6.70
C7552	10.70	39.80	161.20	13.60
9sym	0.30	0.30	0.40	0.30
9symml	0.20	0.30	0.30	0.20
b12	0.70	0.70	0.70	0.70
clip	0.50	0.70	0.60	0.60
des	11.50	15.30	12.80	12.60
ex5p	15.70	18.10	28.40	20.40
rd73	0.40	0.60	0.60	0.40
rd84	1.10	1.60	1.60	1.10
sao2	0.20	0.30	0.20	0.20
TOTAL	58.60	159.70	987.30	89.70
Ar_Mean	4.51	12.28	75.95	6.90
Ar_Ratio	1	2.7	16.9	1.5
Ge_Mean	1.53	3.00	4.93	2.08
Ge_Ratio	1	2.0	3.2	1.4

Table 2: CPU Comparison among FlowMap, HeteroMap, BinaryHM and CN-HM on FLEX10K device family.

In order to show the effectiveness of our algorithms towards the final circuit layout delay, we use FPGA development system MAX+PLUSII 8.1 [1] to perform layout on all the mapping solutions from FlowMap, BinaryHM and CN-HM, and sumarize the results in Table 3. The experimental results show that both BinaryHM and CN-HM can reduce 10% of the circuit layout delays (Dly_l) over FlowMap.

7 Conclusions and Future Work

In this paper, we showed that the delay minimization technology mapping problem for *heterogeneous* FPGAs with *bounded* resources is NP-Hard for general networks, but can be solved optimally in pseudo-polynomial time for trees.

Circuits	FlowMap Dly_l (ns)	BinaryHM Dly_l (ns)	CN-HM Dly_l (ns)
alu2	56.80	30.40	30.40
apex4	54.00	43.40	44.10
C5315	53.90	65.70	53.90
C7552	70.80	79.80	70.80
9sym	33.40	28.90	28.90
9symml	30.20	28.90	28.90
b12	37.00	30.70	29.90
clip	32.20	22.50	30.10
des	70.30	70.30	70.30
ex5p	57.00	57.00	57.00
rd73	32.70	29.60	29.60
rd84	37.40	30.10	30.10
sao2	27.50	30.20	30.10
TOTAL	593.20	547.50	534.10
Ar_Mean	45.63	42.12	41.08
Ar_Ratio	1	-8%	-10%
Ge_Mean	43.33	38.66	38.54
Ge_Ratio	1	-11%	-11%

Table 3: Layout Comparison among FlowMap, BinaryHM and CN-HM on FLEX10K device family.

We also presented two heuristic algorithms, named BinaryHM and CN-HM, for delay minimization in *heterogeneous* FPGA designs with *bounded* resources. Both BinaryHM and CN-HM produce favorable results on MCNC benchmarks on Altera FLEX10K device family.

Notice in Table 1 that our algorithems are unable to use the available EMB resources for delay minimization on circuit *des* and *ex5p*. One of the reasons is that these two circuits are large in size (compared to the other benchmarks we use) but small in depth, which makes our algorithms very difficult to minimize their delays. We plan to generalize the algorithms to handle EMB implementations with multiple-output configurations, which is expected to be more effective in terms of delay reduction using EMBs for circuits like *des* and *ex5p*.

On the other hand, we believe that in order to obtain high density and high performance, heterogeneous FPGAs with/without resource constraints are the future trend of the FPGA architecture development. We shall continue working on technology mapping for heterogeneous FPGAs and extend our algorithms to handle the general delay model in heterogeneous FPGA designs, which will take both interconnection delays and the LUT delays into consideration.

We also expect to use our mapping algorithms to evaluate different types of heterogeneous FPGA architectures to achieve better performance and area utilization.

8 Acknowledgments

We would like to give our sincere thanks to Yean-Yow Hwang for the helpful discussions. This work is partially supported by Altera Corp. under California MICRO Program and National Science Foundation Young Investigator Award MIP-9357582.

References

[1] Altera, *"Programmable Logic Devices Data Book"*, Altera Corp., San Jose, CA, 1996.

[2] J. Cong, Y. Ding, *"FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs"*, IEEE Transactions on Computer-Aided Design, Feb. 1994, Vol. 13, No. 1, pp. 1-12.

[3] J. Cong, Y. Ding, *"On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping"*, IEEE Trans. on VLSI Systems, June 1994, Vol. 2, No. 2, pp. 137-148.

[4] J. Cong, Y. Ding, *"Tutorial and Survey Paper — Combinational Logic Synthesis for LUT Based Field Programmable Gate Arrays"*, ACM Transactions on Design Automation of Electronic Systems, Vol. 1, No. 2, April 1996, pp. 145-204.

[5] J. Cong, S. Xu, *"Technology Mapping for FPGAs with Embedded Memory Blocks"*, Proc. ACM International Symposium on FPGA, Monterey, CA., Feb. 1998, pp. 179-188.

[6] J. Cong, S. Xu, *"Delay-Optimal Technology Mapping for FPGAs with Heterogeneous LUTs"*, Proc. 35th ACM/IEEE Design Automation Conf., San Fransisco, CA., June, 1998, pp. 704-707.

[7] T. Cormen, C. Leiserson, R. Rivest, *"Algorithm"*, MIT Press, Cambridge, MA, 1990

[8] J. He, J. Rose, *"Technology Mapping for Heterogeneous FPGAs"*, Proc. ACM International Symposium on FPGA, Feb. 1994.

[9] M. R. Korupolu, K. K. Lee, D. F. Wong, "Exact Tree-based FPGA Technology Mapping for Logic Blocks with Independent LUTs", Proc. 35th ACM/IEEE Design Automation Conf., June 1998, pp. 708-711.

[10] Lucent Technologies, *"ORCA OR2C-A/OR2T-A Series FPGAs Data Sheet"*, Lucent Technologies, Inc., Allentown, PA, 1996.

[11] Advanced Micro Devices, *"VANTIS VF1 FPGA Data Sheet"*, Advanced Micro Devices, Inc., Sunnyvale, CA, 1998.

[12] S. J. E. Wilton, *"SMAP: Heterogeneous Technology Mapping for Area Reduction in FPGAs with Embedded Memory Arrays"*, Proc. ACM International Symposium on FPGA, Monterey, CA, Feb. 1998, pp. 171-178.

[13] Xilinx, *"The Programmable Logic Data Book"*, Xilinx Inc., San Jose, CA, 1997.

Appendix: NP-Completeness Proof of the DB-HM-BR Problem

We start with a lemma about the transformation from a satisfiable **3SAT** instance to a delay-bounded **DB-HM-BR** instance.

Lemma 1 *If $\mathcal{F}$ is satisfiable, then N has a mapping solution of delay $d+3$ using K_1-LUTs and no more than r K_2-LUTs.*

Proof: Let $v(x_i)$ be the value of x_i in the truth assignment that satisfies $\mathcal{F}$. We compute the mapping solution M as follows. For $i = 1, 2, \ldots, r$, if $v(x_i) = 1$, we perform $pack\_K_2(x_i)$, otherwise, perform $pack\_K_2(\bar{x}_i)$. Clearly, in each subnetwork $N(x_i)$, exactly one of $pack\_K_2(x_i)$ and $pack\_K_2(\bar{X}_i)$ is performed, and exactly r K_2-LUTs are used in total. If $pack\_K_2(x_i)$ is performed, the delay of x_i is d and that of $\bar{x}_i$ is $d+1$. If $pack\_K_2(\bar{x}_i)$ is performed, the delay of x_i is $d'+1$ and that of $\bar{x}_i$ is d ($d < d'+1$). On the otherhand, because the assignment satisfies $\mathcal{F}$, each clause C_j contains at least one literal with value 1, corresponding to a literal node in subnework $N(C_j)$ whose variable node in some subnetwork $N(x_i)$ is packed with its predecessors and the delay for this variable node is d. Therefore, $pack\_K_1(e_j)$, $j = 1, 2, \ldots, m$, is performed to pack two of the three literal nodes, which may have delays larger than $d+1$ since their variable nodes' delays are larger than d, into e_j to reduce the delay of e_j to no more than $d+2$.

We now compute the delay of the mapping solution. There are three types of paths from the PI nodes to the PO nodes in N:

- Paths entirely inside $N(x_i)$. Since exactly one of $pack\_K_2(x_i)$ and $pack\_K_2(\bar{X}_i)$ is performed, the delay of $\bar{x}_i$ is either d or $d+1$, both of which are no more than $d+1$. Therefore, the total delay along the path is no more than $d+3$.

- Paths entirely inside$N(C_j)$. No such paths have delay more than $3 \leq d+3$.

- Paths from $N(x_i)$ to $N(C_j)$. According the above analysis, $pack\_K_1(e_j)$, $j = 1, 2, \ldots, m$, is performed to pack two of the three literal nodes, which have delays larger than $d+1$, into e_j to reduce the delay of e_j to no more than $d+2$. Therefore, the delay of O_{C_j} is no more than $d+3$.

In summary, our mapping solution M of N, constructed based on the truth assignment that satisfies $\mathcal{F}$, has maximum delay of $d+3$. ∎

In order to derive a truth assignment that satisfies $\mathcal{F}$ from a delay-bounded mapping solution of N using K_1-LUTs and no more than r K_2-LUTs, we first analyze the characteristic of such a mapping solution.

Lemma 2 *In a mapping solution of N with delay bound of $d+3$ using K_1-LUTs and no more than r K_2-LUTs, exactly one K_2-LUT is used on only one of x_i, $\bar{x}_i$, b_i and O_{x_i} for each i ($i = 1, 2, \ldots, r$).*

Proof: First, in order to maintain the delay bound of $d+3$ over each O_{x_i}, there must be at least one K_2-LUT used in each N_{x_i}. As the mapping solution uses no more than r K_2-LUTs, there is exactly one K_2-LUT used in each $N(x_i)$, and the only possibility to use K_2-LUT is on one of x_i, $\bar{x}_i$, b_i and O_{x_i}. ∎

An interesting observation is that if for some i, neither *pack_K*$_2$ (x_i) nor *pack_K*$_2(\bar{x}_i)$ is performed, which means that one K_2-LUT is used for b_i or O_{x_i} instead of x_i or $\bar{x}_i$, the delay of $\bar{x}_i$ will be $d'+2$. Therefore, no matter how the *pack_K*$_1(e_j)$ operation is performed on the corresponding clause subnetworks which have $\bar{x}_i$ as their variable node, the delay of the POs in those subnetworks will be no less than $d'+4$, which is larger than $d+3$. Therefore, if for some i, neither *pack_K*$_2(x_i)$ nor *pack_K*$_2(\bar{x}_i)$ is performed, $\bar{x}_i$ does not have any fanout, implying that $\bar{x}_i$ is not in any clause of $\mathcal{F}$.

Lemma 3 *In a mapping solution of N with delay bound of $d+3$ using K_1-LUTs and no more than r K_2-LUTs, for each $N(C_j)$, $j = 1, 2, \ldots, m$, at least one of the operations pack_K$_2(x_{j_1})$, pack_K$_2(x_{j_2})$ and pack_K$_2(x_{j_3})$ must be performed, where x_{j_1},x_{j_2} and x_{j_3} are the variable nodes for clause C_j.*

Proof: If for some clause C_j, none of the above operations are performed, all the variable nodes of this clause will have delays larger than d such that all the three literal nodes have delays larger than $d+1$. The delay of O_{C_j}, hence, will be larger than $d+3$, no matter how *pack_K*$_1(e_j)$ is performed. Therefore, at least one of these operations must be performed. ∎

According to Lemmas 2 and 3, if a mapping solution using K_1-LUTs and r K_2-LUTs satisfies delay bound of $d+3$, it will be similar to the one we constructed in the proof of Lemma 1.

Lemma 4 *If N has a mapping solution of delay $d+3$ using K_1-LUTs and no more than r K_2-LUTs, then $\mathcal{F}$ is satisfiable.*

Proof: The truth assignment for $\mathcal{F}$ is constructed as follows. For each variable x_i of $\mathcal{F}$, if *pack_K*$_2(x_i)$ is performed in the mapping solution, we assign $v(x_i) = 1$; otherwise we assign $v(x_i) = 0$. For any clause C_j, according to Lemma 3, at least one K_2-LUT packing operation is performed for some variable node, on which *pack_K*$_2$ operation is performed and whose delay is d, which implies that C_j is satisfied. Therefore, every clause of $\mathcal{F}$ is satisfied by the assignment. ∎