

Performance-Driven Technology Mapping for Heterogeneous FPGAs

Jingsheng Jason Cong, *Senior Member, IEEE* and Songjie Xu, *Member, IEEE*

Abstract—In order to maximize performance and device utilization, the recent generation of field programmable gate arrays (FPGAs) take advantage of speed and density benefits resulting from *heterogeneous* FPGAs, which can be classified into heterogeneous FPGAs *without* bounded resources or heterogeneous FPGAs *with* bounded resources. In this paper, we study the technology mapping problem for heterogeneous FPGAs *with* or *without* bounded resources under the objective of delay optimization. We present the *first* polynomial-time delay optimal technology mapping algorithm, named HeteroMap, for heterogeneous FPGAs *without* bounded resources. Taking different delays of heterogeneous lookup tables (LUTs) into consideration, the HeteroMap algorithm 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. We then study the technology mapping problem for delay minimization for heterogeneous FPGAs *with* bounded resources. We show that this problem is NP-hard for general networks, in contrast to the delay minimization mapping problem for heterogeneous FPGAs *without* bounded resources, but can be solved optimally in pseudopolynomial time for trees. We then present two heuristic algorithms to solve this problem for general networks. We have successfully applied these algorithms on MCNC benchmarks on commercial FPGAs. Encouraging results on delay and area reduction are reported.

Index Terms—Boundary resources, circuit synthesis, design automation, field programmable gate arrays, heterogeneous LUTs, NP-hard, technology mapping, very large scale integration.

I. INTRODUCTION

THE short design cycle and low nonrecurring engineering (NRE) cost have made the field programmable gate arrays (FPGAs) an important technology in very large scale integration (VLSI) designs. In a traditional lookup table (LUT)-based FPGA device, the basic programmable logic block (PLB) is a K -input LUT (K -LUT) which can implement any Boolean function of up to K variables. In order to maximize performance and device utilization, recent generations of FPGAs take advantage of speed and density benefits resulting from *heterogeneous* FPGAs, which provide either an array of homogeneous 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 [20], Lucent ORCA2C series FPGAs [16], and the recently announced

Manuscript received June 1, 1999; revised March 4, 2000. This work was supported in part by Altera Corp., Lucent Technologies, and Xilinx Inc. under the California MICRO Program and by the National Science Foundation (NSF) under Young Investigator Award MIP9357582. This paper was recommended by Associate Editor M. Sarrazadeh.

J. J. Cong is with the Computer Science Department, University of California at Los Angeles, CA 90095 USA (e-mail: cong@cs.ucla.edu).

S. Xu is with Aplus Design Technologies, Inc., Los Angeles, CA 90024 USA. Publisher Item Identifier S 0278-0070(00)10285-4.

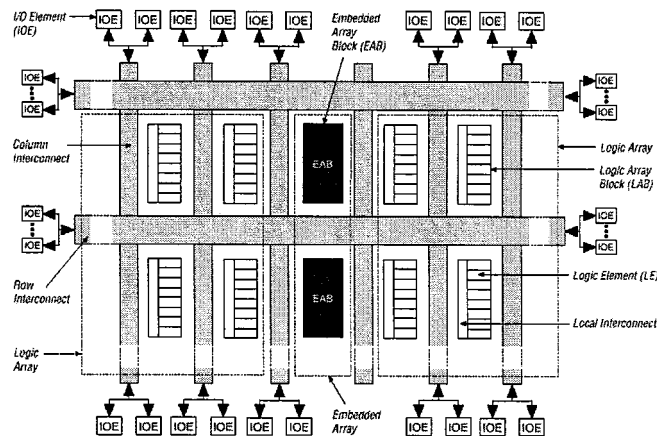


Fig. 1. Altera FLEX10K device block diagram (Courtesy of Altera Corp.).

Vantis VF1 FPGAs [1]¹ can be configured to have heterogeneous LUTs. These heterogeneous FPGAs do not have limitations on the availability of LUTs of specific configurable sizes (as long as within the chip capacity) due to their PLB configuration flexibility. On the other hand, Altera FLEX10K devices [2] (see Fig. 1) 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 limitations on one or several types of LUTs. For example, in one FLEX10K device chip, there are 3–12 EMBs² according to the device size, and each EMB can be used as an 11-LUT. Therefore, heterogeneous FPGAs can be classified as heterogeneous FPGAs *without* bounded resources or heterogeneous FPGAs *with* bounded resources.

In a heterogeneous FPGA, larger LUTs can cover more gates, but usually have longer delays. Given a heterogeneous FPGA *with* or *without* 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. For example, assuming each 4-LUT has delay of 1.0 and each 5-LUT has delay of 1.5, the circuit depicted in Fig. 2(a) can be mapped into five homogeneous 4-LUTs, with the total mapping delay of 3.0 [see Fig. 2(b)]. In Fig. 2(c), the same circuit can be mapped into two 4-LUTs and one 5-LUT, with a total mapping delay of 2.5. In general, given a heterogeneous FPGA with LUTs of different sizes, we want to find an optimal mapping solution with the minimum delay or area. In this paper,

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

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

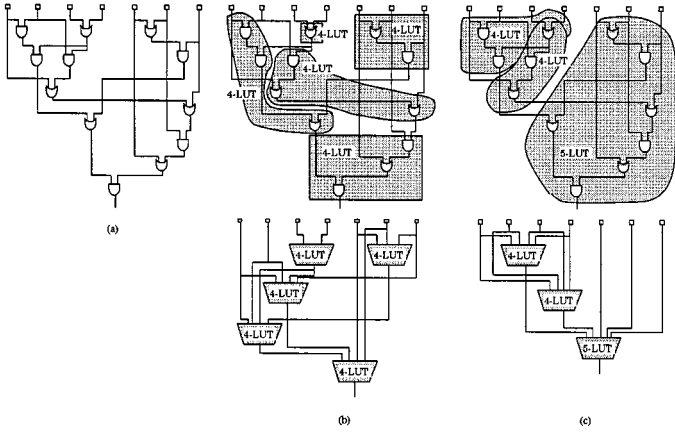


Fig. 2. (a) The original circuit. (b) Mapping to uniform 4-LUTs by the Flowmap algorithm. (c) Mapping to 4-LUTs and 5-LUTs by the HeteroMap algorithm.

we study the technology mapping problem for heterogeneous FPGAs *with* or *without* bounded resources under the objective of delay optimization.

In the past few years, extensive studies have been done on technology mapping for LUT-based FPGAs. A survey of these results can be found in [6]. However, most of these algorithms are not able to deal with the delay optimization problem for heterogeneous FPGAs, as they assume the identical capacity and delay for every LUT. In [14], a heuristic approach for technology mapping into heterogeneous LUT-based FPGAs was presented for area minimization, and their architecture assumes a mixture of only two types of independent LUTs with a fixed ratio in each target PLB. The recent work in [15] shows that the same area minimization mapping problem for a tree network 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 [9] and [19], it was shown that uncommitted EMBs in heterogeneous FPGAs can be efficiently used to implement logic for area minimization. The algorithm in [9] can further guarantee that the circuit delay will not increase while using available EMBs for area reduction.

In this paper, we present the *first* polynomial-time delay optimal technology mapping algorithm, named HeteroMap, for heterogeneous FPGAs *without* bounded resources. Taking different delays of heterogeneous LUTs into consideration, the HeteroMap algorithm computes the minimum mapping delay of a circuit based on a series of minimum-height K -feasible cut³ computations at each node in the circuit. For a heterogeneous FPGA consisting of K_1 -LUTs, K_2 -LUTs, \dots , 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. HeteroMap also effectively minimizes the area of the mapping solution by maximizing the volume of each cut and by the post-mapping packing operations. We then study the technology mapping problem for delay minimization for heterogeneous FPGAs *with* bounded resources, where the HeteroMap algorithm cannot be applied directly, since HeteroMap

cannot constrain the use of certain types of LUTs. We show that this problem is NP-hard for general networks, in contrast to the delay minimization mapping problem for heterogeneous FPGAs *without* bounded resources, but can be solved optimally in pseudopolynomial time for trees. We then present two heuristic algorithms, named BinaryHM and CN-HM, to solve this problem for general networks. HeteroMap, BinaryHM, and CN-HM produce favorable results on MCNC benchmarks on commercial heterogeneous FPGAs.

The remainder of this paper is organized as follows. Section II presents the problem formulation and preliminaries. Section III describes our delay optimal technology mapping algorithm for heterogeneous FPGAs *without* bounded resources. Section IV studies the delay minimization technology mapping problem for heterogeneous FPGAs *with* bounded resources. Experimental results and comparative study are presented in Section V. Section VI concludes the paper. Preliminary results in this paper were presented as extended abstracts in [10] and [11].

II. PROBLEM FORMULATION AND PRELIMINARIES

A Boolean network N can be represented as a directed acyclic graph (DAG) where each node represents a logic gate,⁴ 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 $\text{input}(v)$ to denote the set of fanins of gate v . Given a subgraph H of the Boolean network, let $\text{input}(H)$ denote 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,⁵ such that $|\text{input}(C_v)| \leq K$ and any path connecting a node in C_v and v lies entirely in C_v . The *level* of a node v is the length of the longest path from any PI node to v . The level of a PI node is zero. The *depth* of a network is the largest node level in the network. A Boolean network is K -bounded if $|\text{input}(v)| \leq K$ for each node v in the network.

We assume that a general LUT-based *heterogeneous* FPGA consists of c types of LUTs of K_1 -LUT, K_2 -LUT, \dots , and K_c -LUT ($K_1 < K_2 < \dots < K_c$), with delays d_1, d_2, \dots , and d_c ($d_1 < d_2 < \dots < d_c$, and d_1, d_2, \dots, d_c may not be integer), and with resource bounds B_1, B_2, \dots , and B_c , one for each type of LUTs, respectively. Without loss of generality, d_1 is scaled to one in remaining discussions.

For heterogeneous FPGAs *without* bounded resources, all of B_i s are ∞ . 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,⁶ which is called *heterogeneous*

⁴In the rest of the paper, *gate* and *node* are used interchangeably for Boolean networks.

⁵ u is a predecessor of v if there is a directed path from u to v .

⁶The 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 Section VI for more discussions.

³A minimum-height K -feasible cut is a K -feasible cut with the minimum height. The concepts of K -feasible cut and the height of a cut are to be defined formally in Section II.

LUT-delay model. The unit-delay model used in [4] for homogeneous FPGAs is a special case of the heterogeneous LUT-delay model.

Given these definitions, the technology mapping problem for heterogeneous FPGAs can be formulated as follows: Given a K_1 -bounded Boolean network N and the heterogeneous FPGA with or without 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. The corresponding technology mapping problem for delay minimization for heterogeneous FPGAs with/without bounded resources can be abbreviated as problem **DM-HF-UR** and problem **DM-HF-BR**, respectively.

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 secondary objective is to reduce the area used in the technology mapping solution. Fig. 2(b) shows an example of mapping a combinational circuit into uniform 4-LUTs, which is done by FlowMap [4], while Fig. 2(c) illustrates how the circuit is to be mapped into a heterogeneous FPGA with both 4-LUTs and 5-LUTs, which can be achieved by the HeteroMap algorithm presented in Section III.

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 $C(X, \bar{X})$, 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 $n(X, \bar{X})$, 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., $n(X, \bar{X}) \leq K$. Assuming that each edge (u, v) has a nonnegative 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 $l(v)$ ⁷ associated with each node v , the *height* of a cut (X, \bar{X}) , denoted $h(X, \bar{X})$, is defined to be the maximum label of the nodes in $C(X, \bar{X})$. The *volume* of a cut (X, \bar{X}) , denoted $\text{vol}(X, \bar{X})$, is the number of nodes in \bar{X} , i.e., $\text{vol}(X, \bar{X}) = |\bar{X}|$. Assuming that each edge has the capacity of one, Fig. 3 shows a cut (X, \bar{X}) in a network with given node labels, where $n(X, \bar{X}) = 3$, $e(X, \bar{X}) = 9$, $h(X, \bar{X}) = 2.5$, and $\text{vol}(X, \bar{X}) = 9$. The darkened nodes are in the node cut set.

III. DELAY OPTIMAL TECHNOLOGY MAPPING FOR HETEROGENEOUS FPGAs WITHOUT BOUNDED RESOURCES

In this section, we present a delay optimal technology mapping algorithm, named HeteroMap, for heterogeneous FPGAs

⁷ $l(v)$ will be used later on to denote the minimum mapping delay at node v . The computation procedure for $l(v)$ will be described in Section III. Note that $l(v)$ may not be an integer.

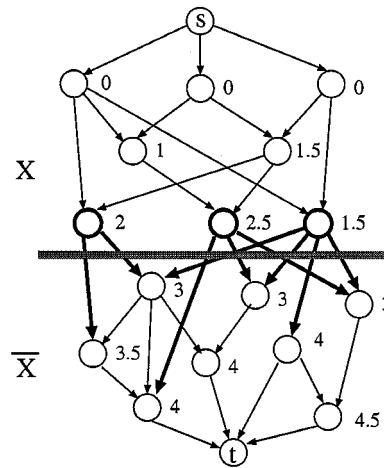


Fig. 3. A 3-feasible cut of node cut-size 3, edge cut-size 9, height 2.5, and volume 9.

without bounded resources under the heterogeneous LUT-delay model. It is applicable to any K -bounded ($K \leq K_1$) Boolean network. Given a general Boolean network as input, if it is not K -bounded ($K \leq K_1$), we first transform it into a two-input simple gate network by using any of the decomposition algorithms, such as `tech_decomp` in SIS [17], DMIG [3], and DOGMA [12]. The optimality of our algorithm holds not only for two-input simple gate networks, but for any K -bounded ($K \leq K_1$) general Boolean network as well.

The HeteroMap algorithm has two phases. In the first phase (Section III-A), according to the topological order from PI to PO, HeteroMap uses the dynamic programming technique to compute the label for each node, which is the delay of the node if implemented by a LUT in a delay-optimal mapping solution. In the second phase (Section III-B), according to the reverse topological order starting from POs, HeteroMap generates the heterogeneous LUT mapping solution based on the node labels and cuts computed in the first phase. Our algorithm also minimizes the circuit area by maximizing the volume of each cut and by the post-mapping packing operations (Section III-C). The details are discussed in the following subsections.

A. 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 . The *label* of t , denoted $l(t)$, is the delay of t in the *optimal* heterogeneous LUT mapping solution of N_t . The first phase of our algorithm computes the labels for all the nodes in N , according to the topological order starting from the PIs. The topological order guarantees that every node is processed after all of its predecessors have been processed. For each PI node u , we assign $l(u) = 0$. Suppose that t is the current node being processed. Then, for each node $v \neq t$ in N_t , the label $l(v)$ must have 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 . Fig. 4(a) shows a Boolean network in which gate t is to be labeled. Fig. 4(b) shows the construction of the network N_t . Assuming that $LUT(t)$ is the K -LUT that implements node t in an optimal mapping solution of N_t under the heterogeneous

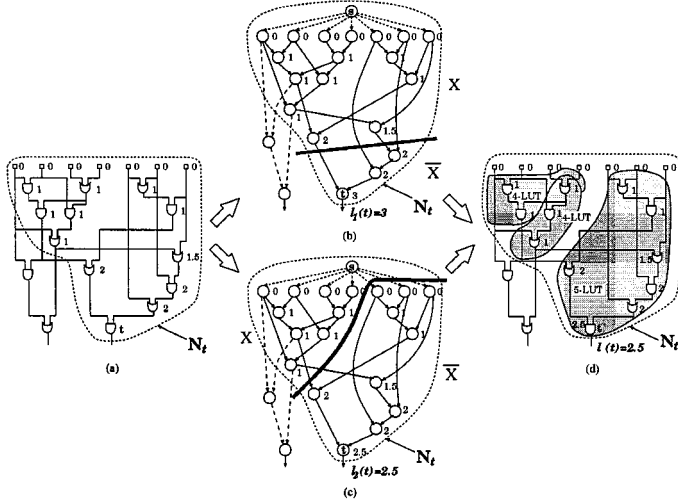


Fig. 4. (a) The Boolean network in which gate t is to be labeled. (b) The construction of N_t and the computation of the minimum-height 4-feasible cut in N_t such that $l_1(t) = 3$. (c) The computation of the minimum-height 5-feasible cut in N_t such that $l_2(t) = 2.5$. (d) Determining $l(t) = \min\{l_1(t), l_2(t)\} = 2.5$.

LUT-delay model, K must be K_1, K_2, \dots , or K_c . Suppose that we can compute $l_i(t)$ for each $i = 1, 2, \dots$, and c , which is the minimum delay of node t in the optimal mapping solution of N_t if t is implemented by a K_i -LUT. Then, label $l(t)$ is the minimum one among all $l_i(t)$ ($i = 1, 2, \dots, c$). That is

$$l(t) = \min_{1 \leq i \leq c} l_i(t). \quad (1)$$

In order to compute $l_i(t)$ for each $i = 1, 2, \dots$, and c , suppose $\text{LUT}_i(t)$ be the K_i -LUT that implements node t . Let \bar{X}_i denote the set of nodes in $\text{LUT}_i(t)$ and X_i denote the remaining nodes in N_t . Then, (X_i, \bar{X}_i) forms a K_i -feasible cut between s and t in N_t because the number of inputs of $\text{LUT}_i(t)$ is no more than K_i . Moreover, $l_i(t) = h(X_i, \bar{X}_i) + d_i$. Therefore, in order to minimize $l_i(t)$, we want to find a minimum-height K_i -feasible cut (X_i, \bar{X}_i) in N_t . Equation (1) can be rewritten as

$$l(t) = \min_{1 \leq i \leq c} \min_{(X_i, \bar{X}_i) \text{ is } K_i\text{-feasible}} (h(X_i, \bar{X}_i) + d_i) \quad (2)$$

Based on the above discussion, we have the following:

Lemma 1: The label $l(t)$ computed by (1) gives the minimum delay of any mapping solution of N_t under the heterogeneous LUT-delay model.

In Fig. 4, we assume that there are two types of LUTs, 4-LUTs and 5-LUTs, with delays of 1.0 and 1.5, in a heterogeneous FPGA device. Fig. 4(b) shows a minimum-height 4-feasible cut, which leads to $l_1(t) = 3$. Fig. 4(c) shows a minimum-height 5-feasible cut, which leads to $l_2(t) = 2.5$. Then, $l(t) = \min\{3, 2.5\} = 2.5$, and the LUT that implements t in the optimal mapping solution is a 5-LUT as shown in Fig. 4(d).

Lemma 2 (Monotone Property): Let $l(t)$ be the label of node t and $l(t')$ be the label of a predecessor t' of t , then $l(t) \geq l(t')$.

Proof: Suppose that the type of LUT that node t uses in the optimal mapping solution of N_t is a K_i -LUT. Then the cut (X, \bar{X}) formed by this K_i -LUT in N_t is a minimum-height K_i -feasible cut in N_t , and $l(t) = h(X, \bar{X}) + d_i$. Since node

t' is a predecessor of node t , (X, \bar{X}) also determines a K_i -feasible cut (X', \bar{X}') in $N_{t'}$ with $h(X', \bar{X}') \leq h(X, \bar{X})$, where $X' = X \cap V(N_{t'})$ and $\bar{X}' = \bar{X} \cap V(N_{t'})$. Moreover, according to (2), $l(t') \leq h(X', \bar{X}') + d_i$. Therefore, $l(t) = h(X, \bar{X}) + d_i \geq h(X', \bar{X}') + d_i \geq l(t')$. ■

According to (1) and (2), the key problem in computing $l(t)$ for node t is to compute the minimum-height K_i -feasible cut (X_i, \bar{X}_i) in N_t to get $l_i(t) = h(X_i, \bar{X}_i) + d_i$, where $i = 1, 2, \dots$, and c . The computation of a minimum-height K -feasible cut in the case of the heterogeneous LUT-delay model is more complicated than that under the unit-delay model, although $l(t)$ is monotone. First of all, we need to determine the range of the minimum height of a K_i -feasible cut in N_t , denoted $H_t(i)$.

Lemma 3: Let t be a non-PI node in network N , then

$$\max\{l(u) : u \in \text{input}(t)\} - d_i \leq H_t(i) \leq \max\{l(u) : u \in \text{input}(t)\}. \quad (3)$$

Proof: Since $(N_t - \{t\}, \{t\})$ is a K -feasible cut in N_t , $l_i(t) \leq \max\{l(u) : u \in \text{input}(t)\} + d_i$. According to Lemma 2, $l_i(t) \geq \max\{l(u) : u \in \text{input}(t)\} - d_i$. Therefore, we can derive the range of $H_t(i)$ to be $\max\{l(u) : u \in \text{input}(t)\} - d_i \leq H_t(i) \leq \max\{l(u) : u \in \text{input}(t)\}$. ■

Based on (3), similar to the approach in [7] for homogeneous LUT mapping with the arbitrary fixed net delay model, we construct a sorted height array, denoted $H_{\text{tlist}}(i)$,⁸ which includes all the distinct labels $l(u)$ ($u \in N_t$) that are in the range of (3). The size of the height array is at most $|N_t|$. We can perform a binary search over this sorted height array to determine the minimum height $H_t(i)$ of which a K_i -feasible cut is available, and hence $l_i(t)$.

Whether N_t has a K -feasible cut of height H or not can be tested efficiently using the following method [4]. We first apply a network transformation on N_t that collapses all the nodes in N_t with $\text{label} > H$, together with t , into the new sink t' . Denote the resulted network as N'_t . It is proved in [4] that N_t has a K -feasible cut of height H , if and only if N'_t has a K -feasible cut. To test whether there is a K -feasible cut in N'_t , we can compute the min-cut in N'_t and check whether its cut size is no more than K . In order to compute the min-cut in N'_t , we apply the approach proposed in [4], which converts the node cut problem on N'_t into a standard edge cut computation problem on a flow network transformed from N'_t . The edge cut problem can be solved by the augmenting path algorithm for max-flow min-cut computation. Assuming that the number of edges in N_t is m , we can determine in $O(K_i \cdot m)$ time whether N_t has a K_i -feasible cut of height H , and find one if such a cut exists. Since we can find the K_i -feasible cut of *minimum height* in N_t by performing the binary search on the sorted height array $H_{\text{tlist}}(i)$ whose size is bounded by $|N_t|$, we have the following.

Lemma 4: A minimum-height K_i -feasible cut in N_t under heterogeneous LUT-delay model can be found in $O(K_i \cdot m \cdot \log |N_t|)$ time where m is the number of edges in N_t .

For each $i = 1, 2, \dots, c$, we compute $l_i(t)$, and the label of node t can be determined by (1) in $O(\sum_{i=1}^c K_i \cdot m \cdot \log |N_t|)$

⁸Our implementation can extract the sorted height array $H_{\text{tlist}}(i)$ in $O(|N_t|)$ time based on merge sort.

```

Algorithm HeteroMap
/*phase 1: label network*/
1  for each PI node  $v$  do
2     $l(v) = 0$ ;
3  for each non-PI node  $t$  in  $N$  in topological order do
4     $l(t) = \infty$ ;
5    construct the network  $N_t$ ;
6     $p = \max\{l(u) : u \in \text{input}(t)\}$ ;
7    for  $i = 1$  to  $c$  do
8      get the height array  $H_{t_{i:st}}(i)$  which includes
          all the distinct labels  $l(u)$ 
          ( $u \in N_t$  and  $p - d_i \leq l(u) \leq p$ );
9      compute the minimum height ( $H_t(i)$ )
           $K_i$ -feasible cut ( $X_i(t), \bar{X}_i(t)$ ) in  $N_t$ 
          by binary search on the height
          array  $H_{t_{i:st}}(i)$ ;
10      $l_i(t) = H_t(i) + d_i$ ;
11     if  $l(t) > l_i(t)$  then
12        $l(t) = l_i(t)$ ;  $\bar{X}(t) = \bar{X}_i(t)$ ;
13     endif
14   endfor
15   endwhile
/*phase 2: generate heterogeneous mapping solution*/
16    $L =$  list of all the PO nodes;
17   while  $L$  contains non-PI nodes do
18     take a non-PI node  $v$  from  $L$ ;
19     generate a LUT  $v'$  to implement the function of  $v$ 
          such that  $\text{input}(v') = \text{input}(\bar{X}(v))$ ;
20      $L = (L - v) \cup \text{input}(v')$ ;
21   endwhile
end-algorithm;

```

Fig. 5. Pseudocode of the HeteroMap algorithm.

time, where m is the number of edges in N_t . Applying the label computation for each node in N according to the topological order in the labeling phase, we have:

Theorem 1: Given a network N with n nodes and m edges, the labeling process can be done in $O(\sum_{i=1}^c K_i \cdot n \cdot m \cdot \log n)$ time.

B. Mapping Phase

The second phase of the HeteroMap algorithm generates a delay optimal mapping solution based on the cuts computed in the first phase. This phase is similar to that in [4], except that the LUTs generated in our mapping solution can be heterogeneous. We maintain a list L of nodes that have to be implemented by LUTs. Initially, L contains those nodes that have fanouts to the PO nodes. Then, we repeatedly remove a node u from L , create the LUT based on $\text{LUT}(u)$ computed in the labeling phase, and add into L all the nodes in $\text{input}(\text{LUT}(u))$ whose LUTs have not yet been created. The mapping phase ends when L only contains PI nodes. The entire second phase takes linear time. It is not hard to see that the delay from any PI to the output of $\text{LUT}(u)$ is no more than $l(u)$ in the resulting mapping solution. Therefore, the mapping solution is optimal. The nodes that are never added into L do not need to be implemented since they are completely covered by the LUTs implementing other nodes.

The HeteroMap algorithm is summarized in Fig. 5.

C. Area Minimization in the HeteroMap Algorithm

The secondary objective of the HeteroMap algorithm is area optimization, which is considered by maximizing the volume of each cut during the mapping process and by the post-mapping heterogeneous predecessor packing operations. In general, the minimum-height K_i -feasible cut (X_i, \bar{X}_i) computed in the labeling phase is not unique. Intuitively, the larger $\text{vol}(X_i(t), \bar{X}_i(t)) = |\bar{X}_i(t)|$, the more nodes we can pack into the K_i -LUT t' , which leads to fewer LUTs in total.

Therefore, our algorithm maximizes the volume of each cut during the minimum-height K_i -feasible cut computation. According to the description in Section III-A, the minimum-height K_i -feasible cut computation is reduced to a series of min-cut [13] computations. Therefore, HeteroMap uses the approach proposed in [4], which finds the maximum volume min-cut with the same complexity as that of computing a min-cut. Note that, however, the resulting minimum-height K_i -feasible cut may not be the *maximum volume minimum height K_i -feasible cut* since only the *maximum volume minimum height min-cut* is computed during each cut computation. In order to further reduce the area, HeteroMap extends the *predecessor packing* operation, which was first introduced in [3] for homogeneous mapping solutions, to minimize the number of K -LUTs in the heterogeneous mapping solution, where one LUT capacity can be different from another. If a K_a -LUT u ($a = 1, 2, \dots$, or c) has a fanin K_b -LUT v ($b = 1, 2, \dots$, or c), v has only one fanout, and $|\text{input}(u) \cup \text{input}(v)| \leq K_a$, then v can be merged into u such that the LUT implementing v can be saved. This operation is carried out in the mapping solution of HeteroMap in the reverse topological order from PO to PI, and ends when no such packing operations can be applied.

IV. DELAY-ORIENTED TECHNOLOGY MAPPING FOR HETEROGENEOUS FPGAs WITH BOUNDED RESOURCES

In this section, we shall study the delay-oriented technology mapping problem for *heterogeneous* FPGAs with *bounded* resources, where some of B_i s have finite values as defined in Section II. Section IV-A analyzes the computational complexity of this problem. To solve the problem, Section IV-B presents a pseudopolynomial time optimal algorithm for trees, while Section IV-C presents two heuristic algorithms for general networks.

A. Complexity of Problem DM-HF-BR

We shall investigate in this subsection the computational complexity of the **DM-HF-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 -LUTs ($K_1 < K_2$, r is a variable), with delay ratio of $1 : d$ ($d > 1$). We first define the decision version of the **DM-HF-BR** problem.

Problem: Delay-bounded heterogeneous LUT mapping with bounded resources (**DB-HF-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 first show that the **DB-HF-BR** problem is NP-complete for $K_1 \geq 5$. The proof of the NP-completeness for the **DB-HF-BR** problem is based on the polynomial time transformation from the 3-Satisfiability (**3SAT**) problem to the **DB-HF-BR** problem. First, we define the **3SAT** problem, which is a well-known NP-complete problem.

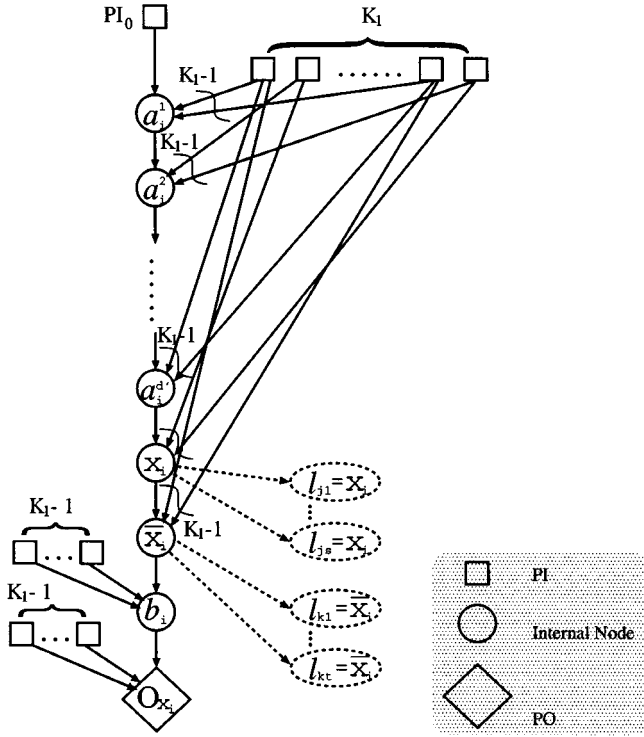


Fig. 6. Transformation of a **3SAT** instance to a **DB-HF-BR** instance: the construction of $N(x_i)$ for variable x_i .

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

Instance: A set of Boolean variables $X = \{x_1, x_2, \dots, x_n\}$ and a collection of m clauses $C = \{C_1, C_2, \dots, C_m\}$, 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 $C_j = 1$ ($j = 1, 2, \dots, m$)?

We shall construct a polynomial time transformation that transforms each instance of **3SAT** to an instance of **DB-HF-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 using the K_2 -LUT implementation on a node in the corresponding network N of the **DB-HF-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, \dots, x_r and m clauses C_1, C_2, \dots, 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 $x_i, \bar{x}_i, a_i^1, a_i^2, \dots, a_i^{d'}$, 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 Fig. 6, where $PI_0, a_i^1, a_i^2, \dots, a_i^{d'}, x_i, \bar{x}_i, b_i$ and O_{x_i} form a direct chain. Every internal node in $N(x_i)$ and PO O_{x_i} has K_1 fanins. Let $S = \{a_i^1, a_i^2, \dots, a_i^{d'}, x_i, \bar{x}_i\}$. Each node in S has one of its fanins coming from a node $s \in \{PI_0\} \cup S$ and the other $(K_1 - 1)$ fanins

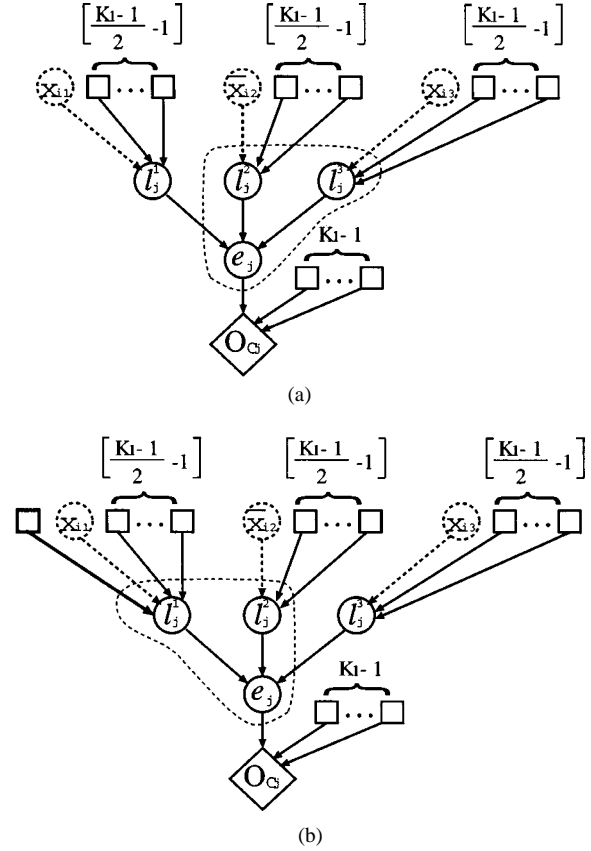


Fig. 7. Transformation of a **3SAT** instance to a **DB-HF-BR** instance: the construction of $N(C_j)$ for variable C_j and the possible packing operation on e_j : (a) When K_1 is odd and $K_1 \geq 5$, or K_1 is even and $K_1 \geq 8$; (b) When $K_1 = 6$.

alternately coming from the first $(K_1 - 1)$ PIs of the K_1 PIs and the last $(K_1 - 1)$ PIs of the same set of K_1 PIs. Thus, $\forall s_1, s_2 \in S$ ($s_1 \in \text{input}(s_2)$), $|\text{input}(s_1)| = K_1$, $|\text{input}(s_2)| = K_1$, and $|\text{input}(s_1) \cup (\text{input}(s_2) - \{s_1\})| = K_1 + 1 > K_1$. Therefore, none of two nodes in S can be packed into the same K_1 -LUT. 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 three 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 ;
- $(3 \cdot \lfloor (K_1 - 1)/2 - 1 \rfloor + K_1 - 1)$ PI nodes (K_1 is odd and $K_1 \geq 5$, or K_1 is even and $K_1 \geq 8$) or $(3 \cdot \lfloor (K_1 - 1)/2 - 1 \rfloor + K_1)$ PI nodes ($K_1 = 6$);
- One PO node, denoted as O_{C_j} .

The nodes are connected as shown in Fig. 7, 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 $\lfloor (K_1 - 1)/2 - 1 \rfloor$ 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 Fig. 7(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 Fig. 7(b). The reason that $N(C_j)$ is constructed in this way is to make sure

that it is impossible to pack all the three literal nodes in $N(C_j)$ to e_j using a K_1 -LUT while it is always possible to pack two of the three literal nodes in $N(C_j)$ to e_j to form a K_1 -LUT (to be shown later).

After connecting subnetworks $N(c_j)$, $j = 1, 2, \dots, m$, with the subnetworks $N(x_i)$ ($i = 1, 2, \dots, r$), the network 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 holds for $K_1 \geq 5$.

We define packing a_1^1, a_2^2, \dots , and $a_i^{d'}$ into x_i to form a K_2 -LUT as $\text{pack}_{K_2}(x_i)$, and packing $a_1^1, a_2^2, \dots, a_i^{d'}$ and x_i into \bar{x}_i to form a K_2 -LUT as $\text{pack}_{K_2}(\bar{x}_i)$. 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 $(K_1 - 1)/2$ fanins. If all the three literal nodes are packed into e_j , the total number of fanins for e_j is $3 \cdot (K_1 - 1)/2$, which is larger than K_1 when $K_1 \geq 5$. If K_2 is an even number, each of the literal nodes has $(K_1/2) - 1$ fanins. If all the three literal nodes are packed into e_j , the total number of fanins for e_j is $(3 \cdot K_1/2) - 3$, 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 Fig. 7(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 $\text{pack}_{K_1}(e_j)$ (see Fig. 7).

By linking the variable assignment of $x_i = 1$ for the **3SAT** instance \mathcal{F} with $\text{pack}_{K_2}(x_i)$, and linking the assignment of $x_i = 0$ with $\text{pack}_{K_2}(\bar{x}_i)$, we can show the following.

Theorem 2: \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 the Appendix to this paper.

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

Proof: **DB-HF-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-HF-BR** instance takes polynomial time. Finally, according to Theorem 2, the **3SAT** question has an YES answer if and only if the **DB-HF-BR** question has an YES answer. Therefore, the NP-completeness of **3SAT** implies that **DB-HF-BR** is NP-complete. ■

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

Note that the construction of N does not apply when $K_1 \leq 4$, therefore, the complexity of the problem is still open for $K_1 \leq 4$.

B. Delay Optimal Mapping for Trees

Although the **DM-HF-BR** problem is NP-hard for general DAGs, we shall show in this section that it can be solved optimally in pseudopolynomial 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, \dots, 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 W in T induces an W_i -cut of T_{v_i} , with $\sum_{i=1}^f W_i = W$, and vice versa. Let $C_T(W)$ denote the set of cuts of size W in T , and define $C_T(1) = t$. It was shown in [5] that

$$C_T(W) = \bigcup_{\sum_{i=1}^f W_i = W} \left(C_{T_{v_1}}(W_1) \times C_{T_{v_2}}(W_2) \times \dots \times C_{T_{v_f}}(W_f) \right) \quad (4)$$

where the union is on all combinations of W_i such that $\sum_{i=1}^f W_i = W$.

Therefore, all the cuts of size W in a tree can be generated based on the recursive equation (4), and the number of cuts generated according to (4) is bounded by a constant depending only on W , which is the $(W - 1)$ th *Catalan number* [13], denoted c_{W-1} , where $c_W = 1/(W + 1) \binom{2W}{W}$. The total number of W -feasible cuts in a tree is thus bounded by $\sum_{i=0}^{W-1} c_i$.

Let $l_v(p)$ be the minimum delay of node v with p K_2 -LUTs used in the mapping solution of T_v , we want to compute $l_v(p)$ for each $p = 0, 1, \dots, r$, for each node v in T from leaves to root t in topological order using dynamic programming. 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, \dots, 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 formula:

$$l_{1cv}(p) = \sum_{j=1}^s \min_{p_j=p} \left\{ \max_{1 \leq j \leq s} l_{v_j}(p_j) + 1 \right\}. \quad (5)$$

Through checking all the K_1 -feasible cuts in T_v , we can compute $l_{1cv}(p)$ for each $p = 0, 1, \dots, r$, 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, \dots, 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 formula:

$$l_{2cv}(p) = \sum_{j=1}^s \min_{p_j=p-1} \left\{ \max_{1 \leq j \leq s} l_{v_j}(p_j) + d \right\}. \quad (6)$$

Through checking all the K_2 -feasible cuts in T_v , we can compute $l_{2cv}(p)$ for each $p = 0, 1, \dots, r$, 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, $l_v(p) = \min\{l_{1cv}(p), l_{2cv}(p)\}$ for each $p = 0, 1, \dots, r$. 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.

Based on the above description, for every node v in T and every K_1 -feasible cut c in T_v , it takes $O(r \cdot r^{K_1})$ time to compute all $l_{1_{cv}}(p)$ ($p = 0, 1, \dots, r$) as in order to compute each $l_{1_{cv}}(p)$, maximally $O(p^{K_1})$ time is needed to enumerate all the p K_2 -LUTs' distributions among the trees rooted at the cut nodes v_1, v_2, \dots, v_s ($s \leq K_1$) of this K_1 -feasible cut c ; for every K_2 -feasible cut c in T_v , it takes $O(r \cdot r^{K_2})$ time to compute all $l_{2_{cv}}(p)$ ($p = 0, 1, \dots, r$). Since the total number of K_2 -feasible cuts in the tree rooted at every node v is bounded by $\sum_{i=0}^{K_2-1} c_i$, where c_i is the i th Catalan number, 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 $l_v(p_1, p_2, \dots, p_q)$ ($0 \leq p_1 \leq r_1, 0 \leq p_2 \leq r_2, \dots, 0 \leq p_q \leq r_q$), 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-HF-BR** problem can be solved optimally in pseudopolynomial time for trees. This result, however, is mainly of theoretical interest as explained in Section IV-C.

C. Delay-Oriented Mapping for DAGs

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-HF-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. Therefore, we do not intend to use the optimal tree mapping algorithm for general DAGs. Instead, we develop two efficient heuristics, BinaryHM (Section IV-C1) and CN-HM (Section IV-C2) to solve this problem. Both use the HeteroMap algorithm presented in Section III to compute a delay-optimal heterogeneous LUT mapping solution without resource constraints under a given heterogeneous LUT delay ratio.

1) *The BinaryHM Algorithm:* We 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 $1 : d$ ($d > 1$). BinaryHM intends to use a limited number of K_2 -LUTs on those nodes that can help the overall delay minimization in a most significant way. For a Boolean network N , let $D_{MM}(N)$ denote the minimum mapping delay of N . We can determine the lower and upper bound of $D_{MM}(N)$ as follows. Let $D_{FM}(N)$ be the minimum mapping delay obtained by FlowMap using only K_1 -LUTs. Clearly, $D_{FM}(N)$ is an upper bound of $D_{MM}(N)$, i.e., $D_{MM}(N) \leq D_{FM}(N)$, since the FlowMap solution does not use any K_2 -LUTs. Let $D_{HM}(N)$ be the minimum mapping delay obtained from HeteroMap using K_1 -LUTs and K_2 -LUTs with delay ratio of $1 : d$. Since $D_{HM}(N)$ is obtained without any resource constraint, it is the lower bound of $D_{MM}(N)$, i.e., $D_{HM}(N) \leq D_{MM}(N)$. Thus

$$D_{HM}(N) \leq D_{MM}(N) \leq D_{FM}(N). \quad (7)$$

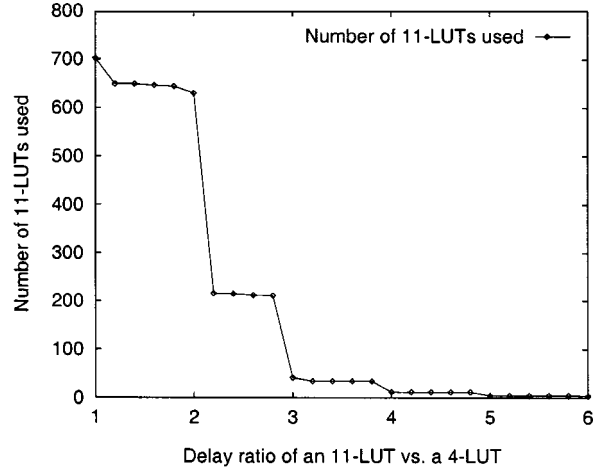


Fig. 8. The number of 11-LUTs used by HeteroMap with respect to different delay ratios between an 11-LUT and a 4-LUT.

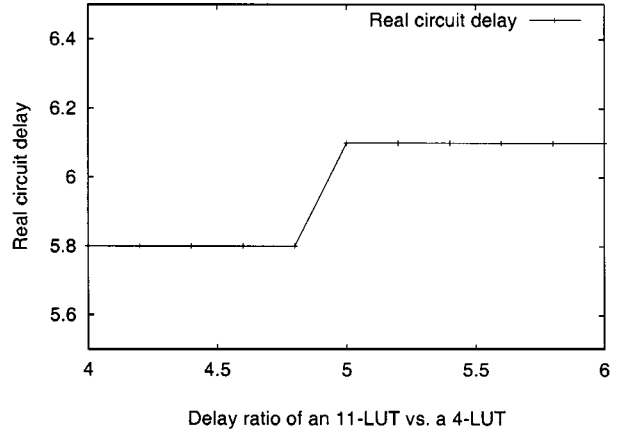


Fig. 9. The real circuit delay obtained by HeteroMap with respect to different delay ratios between an 11-LUT and a 4-LUT.

Given a delay ratio d , HeteroMap uses as many K_2 -LUTs as necessary to achieve the minimum delay for every node in N . However, if we increase the delay ratio of K_1 -LUT versus K_2 -LUT from the original delay ratio, d , all the way to $D_{FM}(N)$, we expect to see that HeteroMap will generate mapping solutions using fewer and fewer K_2 -LUTs as 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. When the delay ratio reaches $D_{FM}(N)$, HeteroMap will produce exactly the same mapping solution as FlowMap, as using any K_2 -LUT will not lead to better mapping solution. For example, with $K_1 = 4$ and $K_2 = 11$, Fig. 8 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. Assuming that the original delay ratio of an 11-LUT versus a 4-LUT is four, by using this delay ratio, the mapping delay $D_{HM}(N)$ achieved by HeteroMap is the lower bound of $D_{MM}(N)$. Fig. 9 shows how the overall circuit delay obtained by HeteroMap (the circuit delay is computed by using the real delay ratio) changes when we increase the delay ratio which results in the decrease of the number of 11-LUTs used in the mapping solution.

Therefore, by doing a binary search on the delay ratio of K_1 -LUT versus K_2 -LUT within the range $[d, D_{FM}(N)]$, BinaryHM will eventually find a mapping solution which uses no more than r K_2 -LUTs and has the minimum mapping delay [in the range specified by (7)].

For a netlist N with n nodes and m edges, the length of the delay ratio range $(D_{FM}(N) - d)$ is at most n . Thus, if the granularity of the binary search over the delay ratio of K_1 -LUT versus K_2 -LUT is selected to be g , BinaryHM will go through the HeteroMap algorithm for $\log(n/g)$ times. Therefore, the complexity of the BinaryHM algorithm is $O(\log(n/g) \cdot (K_1 + K_2) \cdot n \cdot m \cdot \log n)$. For the experimental results reported in Section V-B, the value of g is set to be 0.1.

2) *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 further minimizes the circuit delay using no more than r K_2 -LUTs by focusing on the most delay-critical nodes in N' for each circuit delay target. Let $D_{MM}(N')$ be the minimum mapping delay of N' . Similar to BinaryHM, we first determine the range of $D_{MM}(N')$. Let $D(N')$ be the current delay of N' , in which each node is 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, 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, $D(N')$ 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 every node in N' . Therefore, we have

$$D_{HM}(N') \leq D_{MM}(N') \leq D(N'). \quad (8)$$

CN-HM performs a binary search over the interval $[D_{HM}(N'), D(N')]$. For each circuit delay target D_{tgt} during the binary search, CN-HM identifies all the *critical* nodes in N' , and some of these critical nodes' 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. The concept of critical graph (or network) has been used extensively in *speed-up*, a well-known timing optimization algorithm for combinational logic [18]. The critical graph represents the most timing critical portions of the logic. By considering the critical graph for each circuit delay target, CN-HM can avoid consuming the limited number of K_2 -LUTs on noncritical nodes during the circuit delay minimization.

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

CN-HM first checks whether a delay target D_{tgt} can or cannot be obtained according to the necessary condition depicted in Observation 1 before any further operation is performed. If the necessary condition is not met, CN-HM will locate a less aggressive delay target D_{tgt} through the binary

search. If the condition is met, CN-HM will perform a 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 versus 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 $(D(N') - D_{HM}(N'))$ 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 versus K_2 -LUT is set to be g_2 , the complexity of the CN-HM algorithm will be $O(\log(n'/g_1) \cdot \log(n'/g_2) \cdot (K_1 + K_2) \cdot n' \cdot m' \cdot \log n')$. For the experimental results reported in Section V-B, g_1 is set to be 1.0 and g_2 is set to be 0.1.

In summary, BinaryHM operates on the original unmapped circuit and performs a binary search on the delay ratio of K_1 -LUT versus K_2 -LUT such that HeteroMap can eventually obtain the mapping solution with a 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 a 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 a binary search on the delay ratio of K_1 -LUT versus K_2 -LUT, such that HeteroMap can check whether the delay target is feasible or not.

V. EXPERIMENTAL RESULTS AND COMPARATIVE STUDY

We have implemented HeteroMap, BinaryHM, and CN-HM in the C language on the SUN ULTRA SPARC Workstation and integrated them into the RASP system developed at the University of California at Los Angeles (UCLA) [8]. We present the experimental results of HeteroMap on heterogeneous FPGAs *without* bounded resources in Section V-A and the experimental results of BinaryHM and CN-HM on heterogeneous FPGAs under resource constraint in Section V-B.

A. Delay Optimal Technology Mapping for Heterogeneous FPGAs Without Bounded Resources

We first compare FlowMap [4] and HeteroMap on XC4000 series FPGAs, which can implement circuits with 4-LUTs and 5-LUTs with delays equal to 1.0 and 1.5,⁹ respectively. For FlowMap, we set $K = 5$ such that after mapping, the five-input nodes will be implemented in 5-LUTs, and the remaining nodes will be implemented in 4-LUTs. The mapping comparison results between FlowMap and HeteroMap are summarized in Table I. The " Dly_m " in columns 2 and 7 is the critical path delay in the mapped network under the heterogeneous LUT-delay model. The PLB number in columns 3 and 8 comes from packing every two 4-LUTs into one PLB and leaving all 5-LUTs as independent PLBs. "A-mean" and "G-mean" represent arithmetic mean and geometric mean, respectively. Table I shows

⁹The delay ratio of 4-LUT versus 5-LUT implementation in a XC4000 PLB structure is determined based on the timing characteristics specified in [20] and the best performance by HeteroMap in terms of the post-layout delay.

TABLE I
THE COMPARISON BETWEEN FLOWMAP AND HETEROMAP ON XC4000 SERIES FPGAS

Circuits	FlowMap					HeteroMap				
	Dly _m	PLB	PLB _{m4k}	CPU (s)	Dly _l (ns)	Dly _m	PLB	PLB _{m4k}	CPU (s)	Dly _l (ns)
alu2	13.50	117	87	1.80	75.43	10.50	114	98	7.70	70.94
apex4	8.50	1061	877	67.30	99.77	6.50	691	581	33.40	91.88
C5315	10.50	363	259	10.90	83.55	10.00	327	276	191.90	67.62
C7552	10.50	431	341	15.20	78.50	8.00	390	343	84.80	65.73
9sym	7.00	80	63	0.70	42.32	5.50	77	65	0.80	43.02
9symml	7.00	48	40	0.50	41.13	5.50	48	43	0.40	38.07
b12	6.50	126	91	1.30	43.95	6.00	122	102	1.40	48.98
clip	6.00	107	75	1.10	44.11	5.00	101	88	0.90	41.31
des	7.00	1033	778	92.70	*	6.00	823	706	36.00	*
ex5p	8.00	794	526	34.60	99.28	6.50	745	603	59.00	91.10
rd73	6.50	91	69	0.90	46.26	5.00	76	70	0.70	49.08
rd84	7.50	207	166	2.80	57.56	6.00	177	136	3.10	64.65
sao2	6.00	66	49	0.60	38.07	4.50	49	42	0.40	34.79
TOTAL	104.50	4524	3421	230.40	749.93	85	3740	3153	420.50	707.19
A-mean	8.04	348	263.15	17.72	62.49	6.54	287.69	242.54	32.35	58.93
A-%	1	1	1	1	1	-19%	-17%	-8%	+83%	-6%
G-mean	7.80	203.48	153.66	3.90	58.71	6.33	178.36	151.95	5.37	56.07
G-%	1	1	1	1	1	-19%	-12%	-1%	+38%	-4%

that HeteroMap reduces 19% of the mapping delays and 17% of the PLB numbers over FlowMap.

Match4K [8] is an intelligent post-mapping multistep matching heuristic for XC4000 device family. One of its features is to implement a 5-LUT using a partial XC4000 PLB, which can significantly reduce the PLB number in the mapping solution. To take advantage of the Match4K algorithm in XC4000 mapping, we also compare FlowMap followed by Match4k with HeteroMap followed by Match4K, and perform layout using Xilinx Alliance 1.4 FPGA development system on the mapping solutions. The comparison results are also summarized in Table I. The post-layout delays are reported by Xilinx Alliance 1.4 FPGA development system. After Match4K, HeteroMap can still reduce 6% of the post-layout delays (“Dly_l”) and 8% of the PLB numbers over FlowMap (“PLB_{m4k}”). Although the run time of HeteroMap is slightly higher than FlowMap, the overall compilation time is dominated by the layout design performed by the Alliance 1.4 system. Furthermore, if we impose the same set of timing constraints¹⁰ for both FlowMap and HeteroMap mapping solutions during the layout design, it will take Alliance 1.4 system three to four times longer CPU time to meet the timing constraints on the FlowMap mapping solutions than the ones generated by HeteroMap.

A heterogeneous FPGA could consist of three or more types of logic blocks. In order to show more clearly the benefits of exploring heterogeneity in technology mapping, we test FlowMap and HeteroMap on a hypothetical heterogeneous FPGA architecture which consists of three types of LUTs with input sizes four, five, and six, and the delays 1.0, 1.25, and 1.5, respectively. For FlowMap, we set $K = 6$ such that after mapping each six-input node is implemented by one 6-LUT, each five-input node by one 5-LUT, and all

remaining nodes by 4-LUTs. The results are shown in Table II. In order to compare the area in the mapping solution, let $r = (\text{area of } (K + 1)\text{-LUT}) / (\text{area of } K\text{-LUT})$. In Table II we set $r = 2$ based on the SRAM cell count, which assumes that the area of each 4-LUT, 5-LUT and 6-LUT is one, two, and four, respectively. Under this assumption, HeteroMap reduces 11% of the mapping delays and 20% of the area, when compared with FlowMap. In general, the area ratio of r may not be 2 due to the additional logic and interconnects in a PLB. To make a more accurate area analysis, we perform a sequence of tests with $1 \leq r \leq 3$ and plot the results in Fig. 10 to show the area comparison between FlowMap and HeteroMap. HeteroMap outperforms FlowMap on area when $r \geq 1.5$, and is always better than FlowMap in terms of the delay.

B. Delay-Oriented Technology Mapping for FPGAs with EMBs

We tested BinaryHM and CN-HM on MCNC benchmarks on Altera FLEX10K device family [2], 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 III, where the BinaryHM and CN-HM algorithms are compared with FlowMap [4] which only uses 4-LUTs, and HeteroMap which uses both 4-LUTs and 11-LUTs, and the number of 11-LUTs used by HeteroMap could exceed the resource bounds. 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, 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

¹⁰The timing constraint for each circuit is obtained by taking the minimum post-layout delay of FlowMap and HeteroMap mapping solutions with the layout design without timing constraints and multiplying it by 0.9.

TABLE II
COMPARISON BETWEEN FLOWMAP AND HETEROMAP ON A HETEROGENEOUS FPGA WITH THREE TYPES OF LOGIC BLOCKS

Circuits	FlowMap					HeteroMap				
	Dly _m	Area	4-LUT	5-LUT	6-LUT	Dly _m	Area	4-LUT	5-LUT	6-LUT
alu2	10.00	326	38	40	52	9.25	260	96	38	22
apex4	7.50	2994	200	363	517	6.25	2047	949	367	91
C5315	8.75	1007	161	61	181	8.50	807	393	67	70
C7552	8.50	1024	172	180	123	7.00	948	350	171	64
9sym	5.50	226	8	23	43	5.25	174	80	21	13
9symml	5.50	133	9	8	27	5.25	111	41	7	14
b12	5.50	340	24	38	60	5.25	334	102	34	41
clip	5.25	278	32	37	43	4.50	242	122	24	18
des	4.50	1615	197	5	352	4.00	1558	376	57	267
ex5p	7.25	2543	93	151	537	5.75	1925	619	201	226
rd73	5.75	222	36	15	39	5.00	193	91	11	20
rd84	7.00	494	34	64	83	5.75	405	147	55	37
sao2	4.50	161	19	27	22	4.00	140	56	24	9
TOTAL	85.50	11363	1023	1012	2079	75.75	9144	3422	1077	892
A-mean	6.58	874.08	78.69	77.85	159.92	5.83	703.38	263.23	82.85	68.62
A-%	1	1	1	1	1	-11%	-20%	+235%	+6%	-57%
G-mean	6.38	520.80	46.65	40.57	89.64	5.65	436.47	168.63	45.09	39.01
G-%	1	1	1	1	1	-12%	-16%	+262%	+11%	-57%

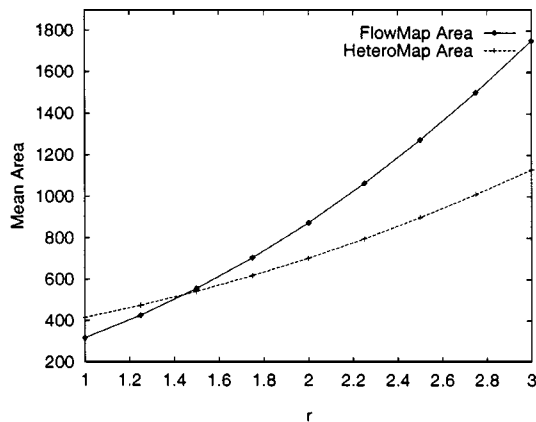


Fig. 10. Area comparison between FlowMap and HeteroMap with respect to the area ratio of $(K + 1)$ -LUT and K -LUT.

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 [9], to further minimize the circuit area while maintaining the circuit delay. As an example, we also showed in Table III the circuit area and the number of EMBs used eventually by CN-HM followed by EMB_Pack (“CN-HM + EP”) as the postprocessing. The number of 4-LUTs in the mapping solutions obtained by the Altera FPGA development system MAX + PLUS II 9.01 [2] is shown in the last column of Table III (“MAX”). From Table III, we clearly see that through the effective use of EMBs under the resource constraints both BinaryHM and CN-HM reduce the delay and area of the mapping solutions considerably over the state of the art of academic algorithms and commercial tools.

The CPU time comparison is summarized in Table IV. From Tables III and IV we can see that both BinaryHM and CN-HM are fairly efficient, using less than 40 min of CPU time for all

13 benchmarks ranging from 200 gates to 3000 gates (four of them have over 2000 gates).

In order to show the effectiveness of our algorithms on the final circuit layout delay, we use the FPGA development system MAX + PLUS II 9.01 [2] to perform layout on all the mapping solutions obtained from FlowMap, MAX + PLUS II itself, BinaryHM and CN-HM, and summarize the results in Table V. The experimental results show that both BinaryHM and CN-HM are able to reduce 12% and 6% of the circuit layout delays (“Dly_l”) over FlowMap and MAX + PLUS II, respectively.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we studied the technology mapping problem for heterogeneous FPGAs *with* or *without* bounded resources under the objective of delay optimization. We presented the *first* polynomial-time delay optimal technology mapping algorithm, named HeteroMap, for heterogeneous FPGAs *without* bounded resources. We further 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 pseudopolynomial time for trees. We then presented two heuristic algorithms, named BinaryHM and CN-HM, to solve this problem for general networks. HeteroMap, BinaryHM and CN-HM produced favorable results on MCNC benchmarks on commercial heterogeneous FPGAs.

From the experimental results of the HeteroMap algorithm for XC4000 series FPGAs, we can see that the post-layout delay is not reduced as dramatically as the mapping delay over the FlowMap algorithm. One of the main reasons is that the interconnection delay, in addition to the LUT delay, ought to be appropriately modeled and optimized during mapping in order to achieve the fully optimized FPGA design. This will be an important future direction in our research. Also notice in Table III that our algorithms are unable to use the available EMB resources for delay minimization on circuits *des* and *ex5p*. One

TABLE III
MAPPING COMPARISON AMONG FLOWMAP, HETERO MAP, BINARYHM, AND CN-HM ON FLEX10K DEVICE FAMILY

Circuits	emb _a	FlowMap		HeteroMap			BinaryHM			CN-HM			CN-HM+EP		MAX
		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	4-LUT
alu2	3	11	192	4	3	3	4	3	3	4	3	3	3	3	165
apex4	8	7	1239	4	34	17	6	576	8	6	577	8	577	8	1085
C5315	10	10	593	9	569	4	9	569	4	10	601	0	536	10	485
C7552	9	9	666	8	624	10	8	639	5	9	680	0	608	9	587
9sym	3	6	141	4	0	1	4	0	1	4	0	1	0	1	100
9symml	3	6	97	4	0	1	4	0	1	4	0	1	0	1	77
b12	3	6	225	4	50	2	4	50	2	4	49	2	29	3	67
clip	3	5	190	4	85	2	4	85	2	4	85	2	85	2	155
des	15	6	1579	5	719	64	6	1579	0	6	1579	0	1573	1	1745
ex5p	6	7	1302	4	59	35	7	1302	0	7	1302	0	1283	4	1305
rd73	3	5	150	4	33	2	4	33	2	4	33	2	0	1	51
rd84	3	6	294	4	3	3	4	3	3	4	3	3	3	3	269
sao2	3	5	90	4	64	1	4	64	1	4	64	1	64	1	105
TOTAL	72	89	6758	62	2243	145	68	4903	32	70	4976	23	4761	47	6196
A-mean	5.54	6.85	519.85	4.77	172.54	11.15	5.23	377.15	2.46	5.38	382.77	1.77	366.23	3.62	476.62
A-%	N/A	1	1	-30%	-67%	1	-24%	-28%	-78%	-21%	-26%	-84%	-30%	-68%	-8%
G-mean	4.61	6.62	324.83	4.57	0	4.15	4.99	0	0	5.08	0	0	0	2.51	247.92
G-%	N/A	1	1	-31%	N/A	1	-27%	N/A	N/A	-23%	N/A	N/A	N/A	-40%	-24%

TABLE IV
CPU COMPARISON AMONG FLOWMAP, HETERO MAP, BINARYHM, AND CN-HM ON FLEX10K DEVICE FAMILY

Circuits	FlowMap CPU (s)	HeteroMap CPU (s)	BinaryHM CPU (s)	CN-HM CPU (s)
alu2	1.10	1.00	288.40	50.20
apex4	10.60	9.40	64.40	21.30
C5315	5.40	85.80	1267.50	6.80
C7552	10.80	42.50	562.90	15.10
9sym	0.30	0.20	0.90	0.40
9symml	0.20	0.20	0.60	0.30
b12	0.70	0.50	1.80	0.80
clip	0.40	0.40	1.20	0.50
des	11.70	9.50	60.90	12.90
ex5p	16.10	9.80	85.30	34.90
rd73	0.40	0.30	1.10	0.40
rd84	1.10	0.90	3.90	1.40
sao2	0.20	0.20	0.60	0.20
TOTAL	59.00	160.70	2339.50	145.20
A-mean	4.54	12.36	179.96	11.17
A-%	1	2.7	39.7	2.5
G-mean	1.50	1.74	12.62	2.61
G-%	1	1.2	8.4	1.8

TABLE V
LAYOUT COMPARISON AMONG FLOWMAP, MAX + PLUS II, BINARYHM, AND CN-HM ON FLEX10K DEVICE FAMILY

Circuits	FlowMap Dly _l (ns)	MAX+PLUS II Dly _l (ns)	BinaryHM Dly _l (ns)	CN-HM Dly _l (ns)
alu2	53.60	56.60	29.90	29.90
apex4	53.00	52.50	46.30	44.10
C5315	55.80	56.90	58.90	58.60
C7552	80.00	80.10	79.60	76.60
9sym	35.40	28.80	28.40	28.40
9symml	30.20	30.70	28.40	28.40
b12	37.00	26.90	29.20	30.00
clip	30.70	34.80	29.30	29.70
des	83.40	66.60	75.60	76.30
ex5p	54.60	56.90	57.00	52.30
rd73	32.20	25.70	29.10	29.10
rd84	38.40	35.30	28.50	28.50
sao2	29.30	25.90	29.60	29.90
TOTAL	613.60	577.70	549.80	541.80
A-mean	47.20	44.44	42.29	41.68
A-mean	1	-6%	-10%	-12%
G-mean	44.31	41.24	38.94	38.59
G-%	1	-7%	-12%	-13%

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 it difficult for our algorithms to minimize their delays. We plan to generalize the algorithms to utilize the multiple-output configurations of EMBs. This is expected to be more effective in terms of delay reduction using EMBs for “wide” circuits like *des* and *ex5p*.

We believe that in order to obtain high density and high performance, heterogeneous FPGAs with or 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 more general delay models 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.

APPENDIX

NP-COMPLETENESS PROOF OF THE DB-HF-BR PROBLEM

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

Lemma 5: 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 $i = 1, 2, \dots, r$, 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 other hand, because the assignment satisfies \mathcal{F} , each clause C_j contains at least one literal with value one, corresponding to a literal node in subnetwork $N(C_j)$ whose variable node in some subnetwork $N(x_i)$ is packed with its predecessors, and the delay for this variable node is d . Therefore, $pack_K_1(e_j)$, $j = 1, 2, \dots, m$, is performed to pack the literal nodes (at most two), which 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 to the above analysis, $pack_K_1(e_j)$, $j = 1, 2, \dots, m$, is performed to pack the literal nodes (at most two), 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 6: 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 ($i = 1, 2, \dots, r$).

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 , then 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 7: 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 $N(C_j)$, $j = 1, 2, \dots, m$, 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 6 and 7, if a mapping solution using K_1 -LUTs and r K_2 -LUTs satisfies the delay bound of $d + 3$, it will be similar to the one we constructed in the proof of Lemma 5.

Lemma 8: 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 7, there is at least one variable node, on which the $pack_K_2$ 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. ■

ACKNOWLEDGMENT

The authors would like thank Y.-Y. Hwang, J. C. Peck, Jr., and C. Wu for their discussions and the anonymous reviewers for their helpful comments.

REFERENCES

- [1] "VANTIS VF1 FPGA Data Sheet," Advanced Micro Devices, Inc., Sunnyvale, CA, 1998.
- [2] *Programmable Logic Devices Data Book*, Altera Corp., San Jose, CA, 1998.
- [3] K. C. Chen, J. Cong, Y. Ding, A. B. Kahng, and P. Trajmar, "DAG-map: Graph-based FPGA technology mapping for delay optimization," *IEEE Design Test Comput.*, pp. 7–20, Sept. 1992.
- [4] J. Cong and Y. Ding, "FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 1–12, Jan. 1994.
- [5] —, "On area/depth tradeoff in LUT-based FPGA technology mapping," *IEEE Trans. VLSI Syst.*, vol. 2, pp. 73–94, June 1994.
- [6] —, "Tutorial and survey paper—Combinational logic synthesis for LUT based field programmable gate arrays," *ACM Trans. Design Automat. Electron. Syst.*, vol. 1, no. 2, pp. 145–204, Apr. 1996.
- [7] J. Cong, Y. Ding, T. Gao, and K. Chen, "LUT-based FPGA technology mapping under arbitrary net-delay models," *Comput. Graphics*, vol. 18, no. 4, pp. 137–148, 1994.
- [8] J. Cong, J. Peck, and Y. Ding, "RASP: A general logic synthesis system for SRAM-based FPGAs," in *Proc. ACM 4th Int. Symp. FPGA*, Feb. 1996, pp. 137–143.
- [9] J. Cong and S. Xu, "Technology mapping for FPGAs with embedded memory blocks," in *Proc. ACM Int. Symp. FPGA*, Monterey, CA, Feb. 1998, pp. 179–188.
- [10] —, "Delay-optimal technology mapping for FPGAs with heterogeneous LUTs," in *Proc. 35th ACM/IEEE Design Automation Conf.*, San Francisco, CA, June 1998, pp. 704–707.

[11] —, "Delay-oriented technology mapping for heterogeneous FPGAs with bounded resources," in *Proc. IEEE Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 1998, pp. 40–45.

[12] J. Cong and Y.-Y. Hwang, "Structural gate decomposition for depth-optimal technology mapping in LUT-based FPGA design," in *33rd ACM/IEEE Design Automation Conf.*, June 1996, pp. 726–729.

[13] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1990.

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

[15] M. R. Korpupolu, K. K. Lee, and D. F. Wong, "Exact tree-based FPGA technology mapping for logic blocks with independent LUTs," in *Proc. 35th ACM/IEEE Design Automation Conf.*, San Francisco, CA, June 1998, pp. 708–711.

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

[17] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephen, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," Univ. California, Berkeley, CA, Tech. Rep. UCB/ERL M92/41, May 1992.

[18] K. J. Singh, A. R. Wang, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Timing optimization of combinational logic," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1988, pp. 282–285.

[19] S. J. E. Wilton, "SMAP: Heterogeneous technology mapping for area reduction in FPGAs with embedded memory arrays," in *Proc. ACM Int. Symp. FPGA*, Monterey, CA, Feb. 1998, pp. 171–178.

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



Songjie Xu (M'00) received the B.S. degree in computer science and engineering from Beijing Institute of Technology, Beijing, China, in 1993, the M.S. degree in computer engineering from Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 1996, and the Ph.D. degree in computer science from University of California, Los Angeles, in 2000.

Her research interests include logic synthesis and layout-driven synthesis. She worked with Altera Corporation, San Jose, CA, in the Summer of 1998. She is a founding member of Aplus Design Technologies, Inc., Los Angeles, CA, where she has been a Senior Engineer since 1999.

Ms. Xu received the Best Graduate Award from the Ministry of Mechanics and Electronics of P.R. China in 1993, Best B.S. Thesis Award in 1993 and Best M.S. Thesis Award in 1996. She received the Dimitris N. Chorafas Foundation Prize for Engineering and Technology from the University of California at Los Angeles in 1999.



Jingsheng Jason Cong (S'88-M'90-SM'96) received the B.S. degree in computer science from Peking University, Beijing, China, in 1985, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 1987 and 1990, respectively.

Currently, he is a Professor and Co-Director of the VLSI CAD Laboratory in the Computer Science Department of University of California, Los Angeles. He has been a Guest Professor of Peking University since 2000. His research interests include layout syn-

thesis and logic synthesis for high-performance low-power VLSI circuits, design and optimization of high-speed VLSI interconnects, FPGA synthesis and reconfigurable architectures. He has published over 120 research papers and led over 20 research projects supported by DARPA, NSF, and a number of industrial sponsors in these areas. He served as the General Chair of the 1993 ACM/SIGDA Physical Design Workshop, the Program Chair and General Chair of the 1997 and 1998 International Symposium on FPGAs, respectively, Program Co-Chair of the 1999 International Symposium on Low-Power Electronics and Designs, and on program committees of many major conferences, including DAC, ICCAD, and ISCAS.

Dr. Cong received the Best Graduate Award from the Peking University in 1985, and the Ross J. Martin Award for Excellence in Research from the University of Illinois at Urbana-Champaign in 1989. He received the NSF Young Investigator Award in 1993, the Northrop Outstanding Junior Faculty Research Award from the University of California, Los Angeles in 1993, the IEEE Trans. on CAD Best Paper Award in 1995 from IEEE CAS Society, and the ACM SIGDA Meritorious Service Award in 1998. He is an Associate Editor of IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS and ACM Transactions on Design Automation of Electronic Systems.