

Technology Mapping for FPGAs with Nonuniform Pin Delays and Fast Interconnections

Jason Cong, Yean-Yow Hwang and Songjie Xu
 Department of Computer Science
 University of California, Los Angeles, CA 90095
 {cong, yeanyow, sxu}@cs.ucla.edu

Abstract

In this paper we study the technology mapping problem for FPGAs with nonuniform pin delays and fast interconnections. We develop the PinMap algorithm to compute the delay optimal mapping solution for FPGAs with nonuniform pin delays in polynomial time based on the efficient cut enumeration. Compared with FlowMap [5] without considering the nonuniform pin delays, PinMap is able to reduce the circuit delay by 15% without any area penalty. For mapping with fast interconnections, we present two algorithms, an iterative refinement based algorithm, named ChainMap, and a Boolean matching based algorithm, named HeteroBM, which combines the Boolean matching techniques proposed in [2] and [3] and the heterogeneous technology mapping mechanism presented in [1]. It is shown that both ChainMap and HeteroBM are able to significantly reduce the circuit delay by making efficient use of the FPGA fast interconnections resources.

1 Introduction

The LUT-based FPGA can be designed in such a way that the K input pins, p_1, p_2, \dots, p_K of each K -input LUT (K -LUT) have different logic access delays, which we call *FPGAs with nonuniform pin delays* (see Figure 1(a)). For example, in Figure 1(b), a K -LUT is implemented by two $(K-1)$ -LUTs with the outputs being selected by one MUX, which creates one fast pin p_1 that goes through only one MUX to the output o . Most of the LUT-based FPGAs have nonuniform pin delays for the LUT.

Another important means for FPGA performance enhancement is the use of fast interconnections, such as the cascade chains in Garp [9] and the local feedback connection in Vantis VF1 series FPGAs [10], which is to trade routing programmability with speed. Cascade chain is one type of fast interconnections widely used in the commercial FPGAs, which is the hard-wire connection between two adjacent LUTs as a high-speed alternative to the slower programmable interconnection. Figure 1(c) shows the two cascaded K -LUTs, A and B , each with an optional cascade input $cascade_{in}$ and an optional cascade output $cascade_{out}$ in addition to the normal input/output connected from/to the general programmable interconnection. This type of hard-wire connection is very fast compared with the normal programmable interconnection. Slightly different from the cascade chain fast interconnection, the fast local feedback connections in VF1 series FPGAs [10] route CBB¹ outputs to any CBB inputs within the same

¹CBB, *configurable building block*, is the basic logic block in VF1 series FPGAs. Each CBB consists of two 3-LUTs wired to one MUX,

Copyright ©1999 by the Association for Computing Machinery, Inc. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept., ACM Inc., fax +1 (212) 869-0481, or (permissions@acm.org).

VGB². This type of local feedback fast interconnection keeps the interconnection delay very short for better performance.

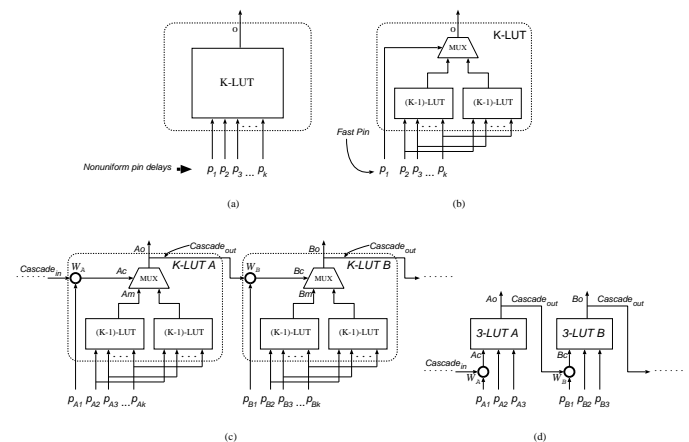


Figure 1: (a) A K -LUT with non-uniform pin delays; (b) A K -LUT with one fast pin; (c) The block diagram of FPGA with fast interconnections; (d) 3-LUT FPGA with fast interconnections.

In the past few years, intensive studies have been done on technology mapping for LUT-based FPGAs. The previous LUT-based FPGA mapping algorithms can be roughly divided into four classes, area minimization, delay minimization, simultaneous area and delay minimization, and routability optimization. A survey of the FPGA mapping algorithms up to the year of 1996 is available in [6]. However, most of these algorithms for delay optimization assume uniform LUT pin delays and constant interconnection delays. Therefore, these algorithms are not able to exploit the important performance enhancement features like nonuniform pin delays or fast interconnections for performance optimization. The only exception is the TEMPT algorithm proposed in [4], which is aimed at exploring FPGA architectures with hard-wire connections. TEMPT packs an LUT mapped circuit netlist into a network of hard-wired logic blocks (HLB), in which each HLB consists of several basic blocks hard-wired connected in an arbitrary tree topology and optimizes either speed or area. The LUTs connected using fast interconnections can be treated as one type of HLB, and the TEMPT algorithm can be applied to optimize the delay or area for this logic block structure. However, TEMPT is a tree-based post-mapping packing algorithm, which greatly restricts the mapping solution space.

To fully utilize these important architecture features for high performance FPGA circuit designs, we study in this paper the technology mapping problem for FPGAs with nonuniform pin delays and fast interconnections for performance optimization.

realizing a 4-LUT or two 3-LUT implementation.

²Each *variable-grain block* (VGB) consists of 4 CBBs.

We propose the PinMap algorithm to compute the delay optimal mapping solution for FPGAs with K -LUT of nonuniform pin delays in polynomial time. For mapping with fast interconnections, we present two efficient algorithms, an iterative refinement based algorithm, named ChainMap, and a Boolean matching based algorithm, named HeteroBM, which combines the Boolean matching techniques proposed in [2] and [3] and the heterogeneous technology mapping mechanism presented in [1]. With Boolean matching, HeteroBM is able to explore the fast interconnection implementation in a larger solution space than ChainMap. On the other hand, the mapping solution generated by HeteroBM has fast interconnection chains with length of only one, while ChainMap is able to identify the long fast interconnection chains along the critical paths for delay minimization. The experiments show that both ChainMap and HeteroBM are able to significantly reduce the circuit delay by making efficient use of the FPGA fast interconnection resources.

The remainder of this paper is organized as follows. Section 2 presents the problem formulation and preliminaries. Section 3 presents the PinMap algorithm for delay-optimal technology mapping for FPGAs with nonuniform pin delays. Section 4 proposes ChainMap and HeteroBM for technology mapping for FPGAs with fast interconnections under the objective of delay minimization. Experimental results and comparative study are presented in Section 5. Section 6 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 gate³, 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. 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. 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 every node v in the network. Given a subgraph H of the Boolean network, let $input(H)$ denote the set of *distinct* nodes outside H which supply inputs to the gates in H .

For a node v in the network, a *cone at v* , denoted C_v , is a subgraph consisting of v and its predecessors⁴ such that any path connecting a node in C_v and v lies entirely in C_v . The cone C_v is K -feasible if $|input(C_v)| \leq K$. Let N_v represent the largest cone at v . A cut of v is a partition (X_v, \bar{X}_v) of N_v such that \bar{X}_v is a cone at v . The *cutset* of the cut is $input(\bar{X}_v)$. A cut is K -feasible if its cutset contains at most K nodes. One way to obtain cuts for a node v is to combine cuts of fanins of v . If a complete set of cuts has been computed for each fanin of v , we can combine cuts from each fanin to obtain a complete set of cuts for v . This technique is called cut enumeration. By pruning cuts that are not K -feasible, we can obtain the complete set of K -feasible cuts for v . Such set of cuts can be computed for every node in N dynamically in a topological order from PIs to POs.

For any FPGA with K -LUTs depicted in Figure 1, the *pin delay* d_i ($i = 1, 2, \dots, K$) is defined to be the delay from the input pin p_i to the output o of the K -LUT. The *interconnection delay* is defined to be the delay from the output A_o of the LUT A to the input of the LUT B . There are two types of FPGA *interconnection* resources in general, the *fast interconnection* such as cascade chain and local feedback connection, with delay of d_c , and the *general routing interconnection* with delay of d_R , which

may be further categorized into *local interconnection*, *semi-global interconnection* and *global interconnection*.

Given a specific FPGA architecture and a circuit design, the pin delays d_i of each LUT are fixed. Once the technology mapping for FPGAs with fast interconnections is accomplished, the delay of the interconnection assigned to be the fast interconnection is also determined, which is d_c . The general routing interconnection delays d_R in the circuit, however, are not determined until the LUT placement and interconnection routing are completed. One way to approximate the general routing interconnection delay is to assume a reasonable uniform delay for all the interconnections to be routed. With this approximation, the general routing interconnection delay can be considered as an overhead associated with each LUT input pin that is not connected to the fast interconnection.

In general, for FPGAs with nonuniform pin delays, we assume that every pin may be associated with a different pin delay. For FPGAs with fast interconnections, each LUT has one optional fast interconnection input $cascade_{in}$ and one optional fast interconnection output $cascade_{out}$.

For experimentation and architecture evaluation, we also divide the LUT-based FPGAs into two categories, FPGAs with *Type-1 LUTs* and FPGAs with *Type-2 LUTs*. In FPGAs with *Type-1 LUTs*, the LUT is implemented using two $(K - 1)$ -LUTs with their outputs selected using an MUX (as in most of the commercial FPGAs), where the pin delays d_i are nonuniform, *i.e.*, the delay of the MUX selection pin (e.g. p_1 in Figure 1(b)) is smaller than the delay of the $(K - 1)$ -LUT input pins. This type of LUT implementation is abbreviated as LUT_{mux} . In FPGAs with *Type-2 LUTs*, the LUT is implemented as a single-output memory block of 2^K SRAM bits, where $d_1 = d_2 = \dots = d_K$. This type of LUT implementation is abbreviated as LUT_{mem} .

Based on the FPGA architectures with nonuniform pin delays and fast interconnections and the delay model described above, we study in this paper the technology mapping problems for FPGAs with nonuniform pin delays and fast interconnections, respectively.

Note that although the formulation of these two problems are based on homogeneous FPGAs with K -LUTs, the algorithms we shall present in the subsequent sections can be applied to heterogeneous FPGAs with LUTs of different sizes as well. Section 5.4 shows the application of our algorithms on Vantis VF1 series FPGAs which provide four types of LUTs with different sizes.

3 Technology Mapping for LUTs with Nonuniform Pin Delays

Similar to the FlowMap algorithm [5], our mapping algorithm, called PinMap, has two phases: the *labeling* phase and the *mapping* phase. In the labeling phase, nodes are labeled with their minimum delays in a topological order from primary inputs to primary outputs. In the mapping phase, a delay-optimal mapping solution is generated according to the node labels and the cuts computed in the labeling phase. Different from FlowMap, our approach does not employ max-flow computation to find the cut that results in the minimum delays. Instead, the set of all K -feasible cuts is pre-computed for every node and is used to obtain appropriate cuts for arbitrary LUT pin delays.

Given a network N , the minimum delay for every node in N is computed from primary inputs to primary outputs in a topological order. Every primary input v_{pi} has a delay $t(v_{pi}) = 0$, hence is given a label of 0. The topological ordering of nodes assures that when a node v is visited, all of its predecessors have been labeled with their minimum delay. The minimum delay at node v , denoted $t(v)$, is obtained by computing the delay resulted from each cut in N_v serving as the fanins to the LUT at v , and selecting the cut that results in the minimum delay. Node v is then

³In the rest of the paper, *gate* and *node* are used interchangeably for Boolean networks.

⁴ u is a predecessor of v if there is a directed path from u to v .

labeled by $t(v)$ and the corresponding cut is saved for mapping solution generation. The minimum delay of N is the maximum label among all primary outputs.

We now present a greedy approach to connect nodes in the cutset $\{u_1, u_2, \dots, u_l\}$ ($l \leq K$) to the pins of LUT at v so that the LUT output has minimum delay. Let u_1, u_2, \dots, u_l satisfy $t(u_1) \geq t(u_2) \geq \dots \geq t(u_l)$. Without loss of generality, we assume that the pin delays d_1, d_2, \dots, d_K of LUT pins p_1, p_2, \dots, p_K also satisfy $d_1 \leq d_2 \leq \dots \leq d_K$. A greedy fanin-to-pin assignment is to connect node u_i to pin p_i for $1 \leq i \leq l$. We prove that the greedy fanin-to-pin assignment results in the minimum delay at the LUT output.

Theorem 1 *The greedy fanin-to-pin assignment results in the minimum delay at the LUT output.*

The proof can be found in [8].

Based on Theorem 1, we can compute the delay resulted from each cut in the complete set of K -feasible cuts of node v , and select the cut that results in the minimum delay. Since nodes are visited in a topological order, it is easy to see that the minimum delay at each node is obtained in the labeling phase.

After labeling nodes with the minimum delays, we traverse nodes in reverse topological order from primary outputs to primary inputs to generate a delay-optimal mapping solution in a way similar to that in [5].

The complexity of the PinMap algorithm is linear to the network size and the number of all K -feasible cuts in the network. Although in theory, the number of all K -feasible cuts grows exponentially with the value of K , each node has a manageable size of K -feasible cuts in practical circuits. When K is large, however, PinMap efficiency may degrade. A set of efficient cut pruning techniques have been proposed recently in [7], which may be applied in PinMap.

4 Technology Mapping for FPGAs with Fast Interconnections

In this section, we shall present two algorithms for technology mapping with fast interconnections: (1) ChainMap, an iterative refinement based algorithm, and (2) HeteroBM, an algorithm that combines the heterogeneous mapping mechanism with boolean matching, to explore the fast interconnection implementation. Both ChainMap and HeteroBM are capable of making use of the fast interconnection resource during technology mapping. In addition, ChainMap can also be applied in the post-mapping processing on the mapped LUT circuits.

4.1 Iterative Refinement Based Mapping Approach

We present a technology mapping heuristic, called ChainMap, which exploits fast interconnections (e.g., cascade chains or fast local feedback) in FPGAs for delay minimization. For the ease of explanation, we focus on cascade chains in this subsection. From Figure 1(c), we see that each LUT has only one cascade chain input and one cascade chain output. Since cascade chains are limited resources, we shall use the chains on critical paths for network delay reduction. The ChainMap algorithm is outlined in Table 1. Initially, the minimum delay of every node for all mapping solutions without using cascade chains is computed (e.g., using PinMap with uniform pin delay) and the variable *status* for each node is reset. Then, ChainMap iterates the steps 4 to 8 to reduce the delay of critical primary outputs until no more reduction can be obtained. At the beginning of each iteration, a set of critical primary outputs are identified according to the

current circuit delay. The delay for each critical primary output is reduced (by using cascade chains) with the procedure *reduce_node_delay()* (outlined in Table 2). The delay at the output of an LUT L can be reduced if one of the following two conditions holds: (1) the delay at all critical fanins of L can be reduced, or (2) L has only one critical fanin which can be connected to L using a cascade chain. In technology mapping, the same conditions hold for any node v and any cut S of v that results in the minimum delay at v . If delay reduction succeeds for every critical primary output, then the minimum delay for every node will be recomputed and the variable *status* for each node will be reset for the next iteration of delay reduction (steps 4 to 8) for new critical primary outputs. When no more delay reduction is obtained, ChainMap exits (step 10). Note that ChainMap does not spend cascade chains on non-critical paths. This is an important consideration for the ease of LUT placement since cascaded LUTs must be placed together on FPGAs. ChainMap can be applied to LUT networks in post-mapping stage for delay reduction with little modification.

```

procedure ChainMap( $N$ )
1  compute initial minimum delay for every node in  $N$ 
2  reset status of every node to unprocessed
3  repeat
4    for each critical primary output  $v$  do
5      status( $v$ ) = reduce_node_delay( $v$ )
6    if status( $v$ ) = succeed for every  $v$ 
7      update minimum delay for every node in  $N$ 
8      reset status of every node to unprocessed
9    else
10   exit
11 until true

```

Table 1: The ChainMap algorithm.

```

procedure reduce_node_delay( $v$ )
1  if  $v$  is a primary input return fail.
2  if status( $v$ ) = fail return fail.
3  if status( $v$ ) = succeed return succeed.

4  for each cut  $S$  that results in minimum delay at  $v$  do
5    for each critical fanin  $u \in S$  do
6      status( $u$ ) = reduce_node_delay( $u$ )
7    if status( $u$ ) = succeed for every critical  $u \in S$ 
8      update  $t(v)$  according  $t(u)$  and the routing delay
9      return succeed
10   if  $S$  contains more than one critical fanins
11     try next cut of  $v$ 
12   if the critical fanin  $u \in S$  chains to other node already
13     try next cut of  $v$ 
14   connect  $u$  to  $v$  using a chain
15   update  $t(v)$  according to  $t(u)$  and the chain delay
16   return succeed

17 return fail

```

Table 2: An algorithm that reduces delay using cascade chains.

Every iteration in ChainMap has a complexity linear to the network size and the number of enumerated K -feasible cuts. When K is not large, each iteration is very efficient. The number of iterations in ChainMap is bounded by the optimal mapping depth for the input network. Overall, ChainMap is very efficient for small LUT size K . When K becomes large, the cut enumeration for all nodes dominates the ChainMap computation time.

4.2 HeteroBM—Boolean Matching Based Technology Mapping

In [2] and [3], the Boolean matching techniques are proposed for complex programmable logic blocks (PLBs) in LUT-based FP-

GAs. With the fast interconnection depicted in Figure 1(c), two cascaded K -LUTs can be taken as a logic block with totally $(2K - 1)$ inputs, which means that theoretically the two cascaded K -LUTs are able to implement logic functions of up to $(2K - 1)$ inputs. With the Boolean matching techniques proposed in [2] and [3], it can be determined whether one I -input wide function ($K < I \leq 2K - 1$) can be implemented by this logic block or not.

Table 3 shows the wide function implementation capability of the two cascaded K -LUTs with $K = 3, 4, 5$ and 6 , based on the Boolean matching procedure finally applied in the HeteroBM algorithm⁵.

I -input Wide Function	Logic Block (K -LUT $\xrightarrow{\text{cascade}}$ K -LUT)			
	$K = 3$	$K = 4$	$K = 5$	$K = 6$
$I = 4$	100%			
$I = 5$	83%			
$I = 6$		100%		
$I = 7$		95%	92%	
$I = 8$		62%	77%	92%
$I = 9$			67%	83%
$I = 10$			61%	77%
				57%

Table 3: The wide function implementation capability of the hard-wire connected K -LUTs with different K s.

As the two cascaded K -LUTs are able to implement some of the wide functions of I inputs ($K < I \leq 2K - 1$), it would be natural to combine the Boolean matching technique with the technology mapping that considers heterogeneous LUTs.

In this work, we use HeteroMap [1] that computes delay-optimal mapping solutions in polynomial time for heterogeneous FPGAs with LUTs of different sizes.

Combining the Boolean matching technique and the heterogeneous technology mapping engine, we propose the HeteroBM algorithm that explores the fast interconnection implementation during LUT technology mapping. Given a K -bounded Boolean network N as the input and the K -LUT FPGAs with fast interconnections, HeteroBM computes the delay optimized mapping solution for FPGAs with fast interconnections in two steps, labeling and mapping.

In HeteroBM, we assume that there are two types of LUTs, K_1 -LUTs and K_2 -LUTs, with $K_1 = K$ and $K < K_2 \leq 2 \times K - 1$. According to the delay model introduced in Figure 1, each pin of a K -LUT implementation has delay of d_{R_o} , which is the pin delay from the general routing interconnection. Each K_2 -LUT implementation computed by HeteroBM consumes two K -LUTs, in which, $p_{A_1}, p_{A_2}, \dots, p_{A_K}$ have the delay of d_{AB} (the delay from p_{A_i} to B_o), and $p_{B_2}, p_{B_3}, \dots, p_{B_K}$ have the delay of d_{R_o} (see Figure 1(c)). The delay derivation for d_{R_o} and d_{AB} is based on the delay model described in Section 2 and is addressed in detail in Table 4.

According to the topological order from PI to PO, in the labeling procedure, HeteroBM labels each node v in N with the delay of v in the mapping solution based on the heterogeneous technology mapping engine described in [1]. For each node v , HeteroBM computes the minimum delay of this node based on the minimum height K_1 -feasible cut and K_2 -feasible cut, and each K_2 -feasible cut has to go through the feasibility checking by Boolean matching to see whether it can be implemented by two cascaded K -LUTs or not. After the labeling procedure, the delay of each node v in the mapping solution and the K -LUT implementation or two cascaded K -LUT implementation for each node v are obtained, which will be used in the subsequent mapping phase to derive the final mapping solution in the reverse topological order from PO to PI. If each Boolean matching procedure for one K_2 -feasible cut takes $O(B)$ time, the complexity of the HeteroBM algorithm is $O((K_1 + K_2) \cdot B \cdot n \cdot m \cdot \log n)$.

⁵The choice of the Boolean matching procedure (configuration) is based on the complexity of corresponding algorithms and their effectiveness.

5 Experimental Results and Comparative Study

5.1 Experimentation Environment

To effectively carry out the experimentation, a set of optimized benchmarks are selected. For details, please refer to [8].

According to the delay models introduced in Sections 1 and 2, a set of realistic delay values have been derived to guide the experimentation and architecture evaluation. Table 4 describes the various delay values in ns and the methods of derivation, which we shall refer to in the experimentation reported in the subsequent sections. Note that as 3-LUTs do not have fast pins (see Figure 1(d)), its d_{co} is not calculated according to these derivation rules described in Table 4(c). In general, the values of all the parameters are derived directly from the related commercial FPGA architectures.

d_{mux} (MUX delay)	d_c (fast interconnection delay)	d_R (general routing interconnection delay)
0.7	0.2	4.1

(a) Basic delay values.

K -LUT	d_{K-LUT}		FPGAs with F.P.		FPGAs with F.I.		
	$d_{K-LUT_{mem}}$	$d_{K-LUT_{mux}}$	d_1	$d_2 = d_3 \dots = d_K$	d_{co}	d_{R_o}	d_{AB}
$K = 3$	2.0	N/A	6.1	6.1	2.2	6.1	8.3
$K = 4$	2.1	2.7	4.8	6.8	0.9	6.8	7.7
$K = 5$	2.2	2.8	4.8	6.9	0.9	6.9	7.8
$K = 6$	2.3	2.9	4.8	7.0	0.9	7.0	7.9

(b) Delay values for FPGAs with F.P. and F.I..

Delay	Description
$d_{K-LUT_{mem}}$	the delay of K -LUT based on LUT_{mem} implementation
$d_{K-LUT_{mux}}$	the delay of K -LUT based on LUT_{mux} implementation
d_1	the delay of the fast pin input
$d_2 = d_3 \dots = d_K$	the delay of the slow pin input
d_{co}	the pin delay from the fast interconnection
d_{R_o}	the pin delay from the general routing interconnection
d_{AB}	the delay from p_{A_i} to B_o

Delay	The method of derivation
$d_{K-LUT_{mem}}$	derived from reality
$d_{K-LUT_{mux}}$	$d_{(K-1)-LUT_{mem}} + d_{mux}$
d_1	$d_R + d_{mux}$
$d_2 = d_3 \dots = d_K$	$d_R + d_{K-LUT_{mux}}$
d_{co}	$d_c + d_{mux}$
d_{R_o}	$d_R + d_{K-LUT_{mux}}$
d_{AB}	$d_{R_o} + d_{co}$

(c) Delay description and derivation.

Table 4: Delay values for FPGAs with fast pins (F.P.) and fast interconnections (F.I.).

5.2 Technology Mapping for FPGAs with Fast Pins

In this section, we compare experimental results from three technology mapping approaches that use or don't use fast pins. We choose $K = 4$ in these experiments. In the first approach, we apply depth-optimal mapping algorithm FlowMap to obtain the first set of experimental data. Assuming no fast pin used, the delay of the mapping solution is the pin delay multiplied by the depth of the mapping solution. We then perform PinMap as post-processing for the FlowMap mapping solutions to exploit LUT fast pins, and obtain the second set (FM_p) of experimental data. At last, we perform our PinMap technology mapping algorithm to obtain the third set of experimental data. The results from the three approaches are reported in Table 5.

Comparing to FlowMap followed by PinMap as a post-processing step, our PinMap mapping algorithm obtains 7% smaller circuit delay on average (aRatio1). If LUT fast pins are not exploited by PinMap in post-processing, FlowMap mapping solutions could have 11% larger circuit delay in the worst case (aRatio1). The area of FlowMap and PinMap mapping solutions

are about the same. The runtime of PinMap is two times longer than the FlowMap runtime. PinMap spends most of its runtime in cut enumeration, since PinMap is very fast as a post-processing procedure where cut enumeration is not needed. FlowMap and PinMap obtain optimal circuit delay for LUTs of types LUT_{mux} and LUT_{mem} , respectively. Comparing Delay1 of PinMap to Delay2 of FlowMap (aRatio2), LUT_{mux} results in 9% smaller circuit delay than LUT_{mem} . However, if PinMap is not available, LUT_{mux} result in 8% larger delay than LUT_{mem} , which shows that with delay optimal technology mapping for FPGAs with fast pins, it is beneficial to develop the LUTs with fast pins for performance optimization.

Circuits	FlowMap				FM _p		PinMap		
	#LUT	Delay1 (ns)	Delay2 (ns)	CPU (s)	Delay1 (ns)	CPU (s)	#LUT	Delay1 (ns)	CPU (s)
C2670	531	68.0	63.0	4.9	66.0	0.7	538	64.0	14.5
C3540	1045	102.0	94.5	12.1	92.0	1.3	1102	82.0	23.8
C5315	759	68.0	63.0	6.4	62.0	1.0	756	56.8	19.9
C6288	894	170.0	157.5	42.3	154.0	1.1	1063	152.0	38.1
C7552	660	68.0	63.0	8.3	62.0	0.9	664	56.0	23.5
alu4	1319	47.6	44.1	3.7	47.6	1.7	1240	43.6	21.2
apex1	667	47.6	44.1	2.4	43.6	0.8	705	39.6	10.8
apex3	836	40.8	37.8	2.8	40.8	1.0	802	38.8	13.9
apex4	1239	40.8	37.8	4.6	40.8	1.4	1280	40.8	23.2
cps	715	34.0	31.5	2.1	34.0	0.8	717	32.0	10.6
dal	516	40.8	37.8	2.7	36.8	0.6	555	34.8	11.4
des	1349	40.8	37.8	8.3	38.8	1.8	1406	36.8	32.6
ex5p	1057	47.6	44.1	5.8	45.6	1.3	1111	39.6	21.1
i10	1346	102.0	94.5	15.9	92.0	1.7	1401	83.2	37.8
i8	551	40.8	37.8	2.2	36.8	0.7	521	32.8	7.9
k2	511	47.6	44.1	2.3	45.6	0.6	546	41.6	9.3
mm30a	857	258.4	239.4	24.9	200.4	1.2	755	191.6	26.8
pair	720	47.6	44.1	3.4	43.6	0.9	704	39.6	15.3
s5378	633	68.0	63.0	2.8	64.0	0.9	646	57.2	13.4
aMean	852.9	72.7	67.3	8.3	65.6	1.1	868.9	61.2	19.7
aRatio1	1.00	1.11			1.00		1.02	0.93	
aRatio2		1.08	1.00					0.91	

Table 5: Comparison of mapping results for FPGAs with fast pins.

5.3 Technology Mapping for FPGAs with Fast Interconnections

With the delay model described in Figure 1 and the delay values set up in Section 5.1, Table 6 shows the mapping comparison results of FlowMap, FlowMap followed by ChainMap as postprocessing (FM_p), ChainMap, HeteroBM, and HeteroBM followed by ChainMap as postprocessing (HBM_p) on FPGAs with 4-LUTs. It is shown that compared with FlowMap mapping solution without using fast interconnections, ChainMap as the postprocessing can reduce 21% circuit delay, while ChainMap during technology mapping is able to reduce 27% of the circuit delay very efficiently. Combining Boolean matching with heterogeneous technology mapping, HeteroBM can obtain 30% reduction of the circuit delay over FlowMap. The mapping solution generated by HeteroBM has fast interconnection chains with length of one, while ChainMap is able to identify the long fast interconnection chains for delay minimization. HBM_p represents the mapping solution generated by either HeteroBM or HeteroBM followed by ChainMap as postprocessing, whichever is better in delay. Compared with HeteroBM, HeteroBM followed by ChainMap cannot further reduce the circuit delay on 17 circuits out of all the 19 benchmarks, but can significantly reduce the use of fast interconnection resources. HeteroMap followed by ChainMap as postprocessing achieves 37% delay reduction over FlowMap, as the HeteroBM mapping solution provides a good starting point for ChainMap.

Table 7 shows the fast interconnection using in the mapping solutions produced by FlowMap followed by ChainMap as postprocessing (FM_p), ChainMap, HeteroBM, and HeteroBM followed by ChainMap as postprocessing (HBM_p). Note that for some circuits, HBM_p uses much less fast interconnection chains than HeteroBM as ChainMap consider the delay minimization

Circuits	FlowMap		FM _p		ChainMap		HeteroBM		HBM _p	
	Delay (ns)	4-LUT	Delay (ns)	4-LUT	Delay (ns)	4-LUT	Delay (ns)	4-LUT	Delay (ns)	4-LUT
C267	68.0	531	68.0	523	62.1	523	44.4	333	44.4	333
C354	102.0	1045	95.2	83.4	83.4	1156	66.6	1327	66.6	1327
C5315	68.0	759	56.2	56.2	56.2	734	44.4	698	44.4	698
C6288	170.0	894	128.7	127.8	127.8	1070	134.6	1057	116.9	116.9
C7552	68.0	660	56.2	50.3	50.3	663	44.4	839	44.4	839
alu4	47.6	1319	47.6	40.8	40.8	1226	36.7	1728	36.7	1728
apex1	47.6	667	41.7	35.8	35.8	680	35.8	913	35.8	913
apex3	40.8	836	40.8	34.9	34.9	788	35.8	1008	35.8	1008
apex4	40.8	1239	40.8	40.8	40.8	1224	35.8	1411	35.8	1411
cps	34.0	715	34.0	28.1	28.1	708	29.0	971	29.0	971
dal	40.8	516	34.9	34.9	34.9	559	29.0	702	29.0	702
des	40.8	1349	40.8	34.9	34.9	1352	29.0	1731	29.0	1731
ex5p	47.6	1057	47.6	40.8	40.8	1118	34.9	1183	34.9	1183
i1	102.0	1346	84.3	79.3	79.3	1380	66.6	1744	66.6	1744
i8	40.8	551	34.9	29.0	29.0	490	29.0	606	29.0	606
k2	47.6	511	41.7	41.7	41.7	520	36.7	684	36.7	684
mm30a	258.4	857	93.2	86.4	86.4	669	152.2	590	81.4	81.4
pair	47.6	720	41.7	41.7	41.7	655	30.8	823	30.8	823
s5378	68.0	633	62.1	62.1	62.1	612	44.4	728	44.4	728
aMean	72.65	852.89	57.39	53.21	53.21	848.79	50.53	1004.47	45.87	1004.47
aRatio	1	1	0.79	0.73	0.73	1	0.70	1.18	0.63	0.63

Table 6: Comparison of mapping results for FPGAs with fast interconnections.

on critical paths only based on the recursive refinement, while HeteroBM will minimize the delay on every node in the circuit netlist, which ends up using lots of fast interconnections on non-critical paths. However, the data reported in column Common_{hc}, which is the number of chains used by both HeteroBM and HBM_p, shows that 99% of the chains used by HBM_p are originally the chains generated by HeteroBM. Compared with the average number of edges (2731.74) in the FlowMap mapping solution, all the algorithms we developed use very few fast interconnections while achieving the great delay reduction.

Circuits	FM _p	ChainMap	HeteroBM	HBM _p	Common _{hc}
C2670	0	28	100	65	65
C3540	7	170	517	517	517
C5315	15	25	227	57	56
C6288	10	16	250	98	91
C7552	5	44	244	244	244
alu4	0	127	665	577	577
apex1	1	114	266	166	163
apex3	0	131	240	198	196
apex4	0	0	311	311	311
cps	0	138	330	188	187
dal	17	39	232	223	223
des	0	128	663	254	251
ex5p	0	167	411	411	411
i10	58	108	587	587	587
i8	33	130	158	100	98
k2	11	71	203	130	124
mm30a	35	44	130	75	58
pair	22	17	284	125	116
s5378	4	13	201	40	37
aMean	11.47	79.47	316.79	229.79	226.95
aRatio	1	6.93	27.61	20.03	19.78

Table 7: Comparison on the usage of fast interconnections.

5.4 Application of the Algorithms on Vantis VF1 FPGA Architecture

In this section, we shall combine the PinMap algorithm with the heterogeneous technology mapping engine to provide the mapping solution for Vantis VF1 series FPGAs, which consist of heterogeneous LUTs with nonuniform pin delays. The CBB and VGB in VF1 series FPGAs use 3-LUTs as basic logic blocks to be connected together to implement 3-LUTs, 4-LUTs, 5-LUTs, and 6-LUTs. Each CBB implements two 3-LUTs or one 4-LUT. Two CBBs together implement a 5-LUT. Each VGB contains 4 CBBs and can implement a 6-LUT. Each LUT size is associated with a specific LUT delay.

According to the timing parameters specified in [10] and the delay derivation method set up in Table 4(c), Table 8 describes the delay information used by our mapping algorithm on VF1 series FPGAs.

In addition to the nonuniform pin delays, the local feedback

d_{mux} (MUX delay)	d_c (local feedback connection delay)	d_R (general routing interconnection delay)
0.6	0.6	4.1

(a) Basic delay values.

K-LUT	Pin delays				
	P6	P5	P4	P2	P1
K = 3			0.6	1.7	1.7
K = 4			0.6	2.4	2.4
K = 5		0.6	1.2	2.8	2.8
K = 6	0.6	1.2	1.8	4.3	4.3

(b) Delay values for FPGAs with F.P. and F.I..

Table 8: Delay information for VF1 series FPGAs.

connection in the VGB can be used to build logic blocks to implement some wide functions based on Boolean matching. For instance, two 3-LUTs can be connected to implement some 4-input and 5-input functions. The delay of the local feedback connection is shown in Table 8(a), which is slower than the hard-wire connection. With the fast pins widely available for each type of LUTs and our delay analysis, the wide function implementation may not create much gain on the overall circuit delay during our technology mapping procedure. Therefore, we intend not to use Boolean matching on VF1 series FPGAs.

We compare experimental results from three technology mapping approaches for Vantis VF1 FPGAs. For such a variety of LUT configurations with different characteristic delays, the HeteroMap algorithm [1] can compute delay-optimal mapping solutions under uniform pin-delay assumption. Our first approach employs HeteroMap to produce experimental data. Assuming no fast pin used, the LUT delay in HeteroMap mapping solution is the largest pin delay for each size of the LUT. We then perform PinMap as a post-processing step for the HeteroMap mapping solutions to exploit nonuniform pin delays, and obtain the second set (HM_p) of experimental data. At last, we perform our PinMap technology mapping algorithm to obtain the third set of experimental data. The results from the three approaches are reported in Table 9.

Comparing to HeteroMap followed by PinMap as a post-processing step, our PinMap mapping algorithm obtains 13% smaller circuit delay on average (aRatio) for Vantis VF1 FPGAs. If nonuniform LUT pin delays are not exploited by PinMap in post-processing, HeteroMap mapping solutions could have 14% larger circuit delay in the worst case (aRatio). PinMap mapping solutions have 60% larger area comparing to HeteroMap mapping solutions. This is due to extensive uses of 6-LUTs in PinMap for its small pin delays. PinMap is two times slower than HeteroMap, but is very fast as a post-processing procedure. Considering performance, area, and computation efficiency together, HeteroMap followed by PinMap for post-processing is the best one among three compared approaches for VF1 FPGAs.

6 Conclusions and Future Work

We studied in this paper the technology mapping problem for FPGAs with nonuniform pin delays and fast interconnections and presented the interesting results to show the effectiveness of the algorithms we developed and the architecture implication. We proposed the PinMap algorithm to take full advantage of the nonuniform pin delays associated with the LUT input pins. For mapping with fast interconnections, we presented two efficient algorithms, ChainMap and HeteroBM. It is shown that these algorithms are able to significantly reduce the circuit delay by making good use of the FPGA fast pins and fast interconnection resources.

In the future, we plan to consider the placement during technology mapping to accomplish the logic synthesis with layout consideration for better circuit performance.

Circuits	HeteroMap			HM _p		PinMap		
	#3LUT	Delay (ns)	CPU (s)	Delay (ns)	CPU (s)	#3LUT	Delay (ns)	CPU (s)
C2670	831	46.6	36.1	44.8	0.4	1408	41.4	110.6
C3540	3721	73.5	193.5	62.0	1.4	5549	53.3	255.8
C5315	1462	46.3	51.9	37.6	0.6	2153	33.3	257.5
C6288	2465	141.8	149.7	124.1	0.9	6103	99.1	663.6
C7552	1882	48.2	103.4	42.5	0.8	2801	35.8	511.4
alu4	3411	39.5	56.9	38.4	1.9	4857	32.4	91.8
apex1	1910	35.6	32.3	32.9	0.8	3571	29.5	69.3
apex3	1849	34.1	33.3	33.3	1.1	4318	29.6	84.9
apex4	3972	35.6	82.1	35.6	1.8	7156	30.6	162.2
cpe	2366	29.1	23.4	26.7	0.9	3118	24.2	46.1
dalu	1380	28.7	22.9	26.9	0.5	1998	24.7	127.6
des	3116	23.7	47.7	20.6	1.3	4057	19.6	248.5
ex5p	3205	33.4	81.5	33.0	1.6	5386	27.3	357.5
i10	5234	75.9	333.5	65.3	1.6	6993	56.4	458.2
i8	1354	28.7	18.5	26.8	0.7	2318	24.0	29.6
k2	1547	39.9	21.0	38.4	0.6	2825	31.4	46.8
mm30a	1652	149.0	198.7	109.9	0.7	2470	104.7	110.8
pair	2140	35.3	23.8	31.2	0.7	2786	25.8	48.6
s5378	1715	51.7	17.0	46.0	0.9	2481	39.9	49.9
aMean	2380	52.5	80.4	46.1	1.0	3808	40.2	196.4
aRatio	1.00	1.14		1.00		1.60	0.87	

Table 9: PinMap results for VF1 series FPGA.

7 Acknowledgments

We would like to give our sincere thanks to Chin-Chih Chang, Cheng-Kok Koh and Chang Wu for their helpful and inspiring discussions. This work is partially supported by Actel and Vantis under the California MICRO Program and National Science Foundation Young Investigator Award MIP9357582.

References

- [1] J. Cong and S. Xu, "Delay-Optimal Technology Mapping for FPGAs with Heterogeneous LUTs", Proc. 35th ACM/IEEE Design Automation Conf., San Francisco, CA, June, 1998, pp. 704-707.
- [2] J. Cong and Y.-Y. Hwang, "Partially-Dependent Functional Decomposition with Applications in FPGA Synthesis and Mapping", Proc. ACM 5th Int'l Symposium on FPGA, Feb. 1997, pp. 35-42.
- [3] J. Cong and Y.-Y. Hwang, "Boolean Matching for Complex PLBs in LUT-based FPGAs with Application to Architecture Evaluation", Proc. ACM 6th Int'l Symposium on FPGA, Feb. 1998, pp. 27-34.
- [4] K. Chung and J. Rose, "TEMPT: Technology Mapping for the Exploration of FPGA Architectures with Hard-Wired Connections", 29th ACM/IEEE Design Automation Conference, 1992, pp. 361-367.
- [5] J. Cong and 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.
- [6] J. Cong and 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.
- [7] J. Cong, Y. Ding, and C. Wu, "Cut Ranking and Pruning: Enabling A General And Efficient FPGA Mapping Solution" Proc. ACM 4th International Symposium on FPGA, Feb. 1999, pp. 29-35.
- [8] J. Cong, Y.-Y. Hwang, and S. Xu, "Technology Mapping for FPGAs with Nonuniform Pin Delays and Fast Interconnection", UCLA Computer Science Department Technical Report CSD-990018.
- [9] J. R. Hauser and J. Wawrzynek, "Garp: A MIPS Processor with a Reconfigurable Coprocessor", Proc. of IEEE Symposium on Field-Programmable Custom Computing Machines, 1997, pp24-33, <http://www.cs.berkeley.edu/projects/brass/documents/GarpArchitecture.html>.
- [10] Advanced Micro Devices, "VANTIS VF1 FPGA Data Sheet", Advanced Micro Devices, Inc., Sunnyvale, CA, 1998.