Boolean Matching for Complex PLBs in LUT-based FPGAs with Application to Architecture Evaluation

Jason Cong and Yean-Yow Hwang

Department of Computer Science
University of California, Los Angeles
{cong, yeanyow}@cs.ucla.edu

Abstract

In this paper, we developed Boolean matching techniques for complex programmable logic blocks (PLBs) in LUT-based FPGAs. A complex PLB can not only be used as a *K*-input LUT, but also can implement some wide functions of more than *K* variables. We apply previous and develop new functional decomposition methods to match wide functions to PLBs. We can determine exactly whether a given wide function can be implemented with a XC4000 CLB or other three PLB architectures (including the XC5200 CLB). We evaluate functional capabilities of the four PLB architectures on implementing wide functions in MCNC benchmarks. Experiments show that the XC4000 CLB can be used to implement up to 98% of 6-cuts and 88% of 7-cuts in MCNC benchmarks, while two of the other three PLB architectures have a smaller cost in terms of logic capability per silicon area. Our results are useful for designing future logic unit architectures in LUT based FPGAs.

1. Introduction

The field programmable gate array (FPGA) is a new technology as an alternative to ASIC designs in recent years. An FPGA chip consists of programmable logic elements and interconnections. A *K*-input lookup-table (K-LUT) is a *K*-input one-output logic element composed of 2^K SRAM cells. The K-LUT can implement any function of up to *K* variables. In general, the FPGA logic utilization may increase if smaller LUTs are used. But circuit performance may degrade due to the increase of the circuit depth. In order to increase the utilization without penalty in performance, most FPGA architectures provide complex Programmable Logic Block (PLB) consisting of multiple LUTs to offer greater flexibility.

In this paper, we consider four PLB structures shown in Figure 1: (a) the XC4000 CLB, and other three PLBs

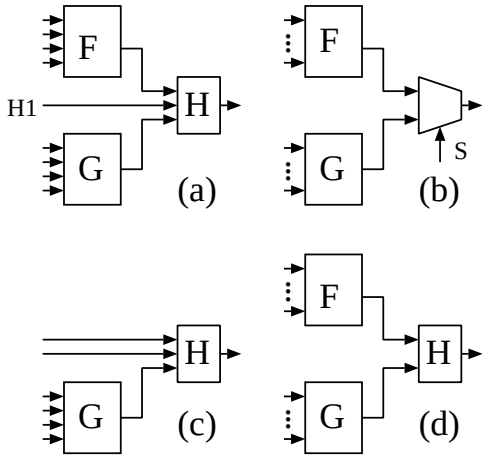

Figure 1 Four PLB architectures. (a) XC4000 CLB, (b) PLB1, (c) PLB2, (d) PLB3.

named (b) PLB1, (c) PLB2, and (d) PLB3. The XC4000 CLB, PLB1, and PLB3 contain two LUTs (F and G) at the input stage and one logic unit (a MUX or an LUT) at the output stage, while PLB2 has one LUT at the input stage and one LUT at the output stage. Obviously, these PLBs have different functional capabilities. The XC4000 CLB can be used to implement any function of up to five inputs, or any two independent functions of four inputs, or some function of up to nine inputs (which we call *wide* functions). PLB1 is the XC5200 CLB when LUTs F and G are 4-LUTs, and can implement any function of up to 5 inputs and some wide functions of 9 inputs. To save silicon area, LUT F or LUT G or both could be 3-LUTs. However, the functional capability is also reduced. For example, the implementation of every 5-input function is no longer guaranteed. PLB2 and PLB3 are simplified from the XC4000 CLB. A smaller silicon area is consumed by PLB2 or PLB3 comparing to the XC4000 CLB, but their functional capabilities are also reduced. In this paper, we shall develop Boolean matching methods for these PLBs and evaluate their functional capabilities.

Using PLBs for direct wide function implementation could be more cost-effective than using PLBs to cover *K*-

input functions obtained from the decomposition of wide functions. However, the problem of matching wide functions to (LUT based) PLBs has not been understood well before. As a result, most existing technology mapping algorithms (see [4] for a comprehensive survey) focus on K-LUT covering for area or delay minimization, or LUT to PLB packing for area minimization. Recently, Sasao and Butler [12] studied the bi-decomposition $f(X)=h(g_1(X_1),g_2(X_2))$ of logic functions. They gave necessary and sufficient conditions for non-disjoint bi-decompositions and considered the case when h is an AND, OR, or EXOR function, but they did not relate their results to Boolean matching for LUT based PLBs. Cong and Hwang [5] characterized two classes of wide functions for the XC4000 CLB based on the partially-dependent decomposition of logic functions and applied their results to LUT based FPGA technology mapping. Good results were reported (13% decrease on circuit depth). But their results did not provide a complete characterization of all wide functions that can be implemented with the XC4000 CLB. In this paper, we completely characterize wide functions for the XC4000 CLB, PLB1, PLB2, and PLB3, and evaluate their functional capabilities. Our evaluation, however, should be considered as only one aspect of a full PLB architecture evaluation, which should include other factors such as the LUT packing capability of PLBs and the routing area needed in PLBs, etc.

The remainder of this paper is organized as follows. Section 2 formulates the Boolean matching for PLB problem. Section 3 presents the functional decomposition approaches for matching wide functions to PLBs. Experimental results are reported in Section 4. Section 5 concludes the paper.

2. Problem Formulation

Given a PLB architecture A that can be used to implement any function of up to K_A variables, a function $f(X)$ is regarded as a *wide function* with respect to A if $|X| > K_A$. For most LUT based PLBs available today, the number K_A is not difficult to obtain. For example, $K_A=5$ for both the XC4000 and XC5200 CLBs while $K_A=6$ for the ORCA PFUs. We study the problem of implementing wide functions with the four PLBs (XC4000 CLB, PLB1, PLB2, and PLB3) in this paper.

Boolean Matching for PLB Given a wide function $f(X)$ with respect to a PLB architecture A, determine if $f(X)$ can be implemented with A.

In general, the Boolean matching problem is difficult when input negation and/or permutation, output inversion, bridging of inputs, and assigning some inputs to 0/1 values are all considered. However, for LUT-based PLBs, the input negation, input permutation at each LUT, output inversion, and assigning some inputs to 0/1 values need not be considered. Only the assignment of inputs to different LUTs in the PLB and the bridging of inputs (among LUTs in the PLB) are relevant factors. In the following section, we solve the Boolean matching for PLB problem by decomposing functions with consideration of both factors.

3. Boolean Matching Techniques for PLBs

In this section, we first introduce terminologies and review classical functional decomposition results given in [1, 7], and then present complete characterizations of wide functions that can be implemented with the XC4000 CLB, PLB1 (incl. the XC5200 CLB), PLB2, and PLB3 architectures.

3.1. Functional Decomposition

Let $X=\{x_1,x_2,\ldots,x_n\}$ be a set of Boolean variables and $f(X)=f(x_1,x_2,\cdots,x_n)$ be a Boolean function. Given $B=\{x_1,x_2,\ldots,x_b\}\subseteq X$, let $f(B,X-B)$ also represent $f(X)$. The support of $f(X)$ is denoted as $sup(f)=X$. Let $f_{\bar{x}_i}=f(X)|_{x=0}$ and $f_{x_i}=f(X)|_{x=1}$ represent the cofactors of $f(X)$ with respect to x_i. Cofactors of $f(X)$ with respect to multiple variables are defined in a similar way. For example, $f_{x_1\bar{x}_2}=f(X)|_{x_1=1,x_2=0}$. The *cofactor set* of $f(X)$ with respect to B, denoted $cs_B(f)$, is the set of all *distinct* cofactors of $f(X)$ with respect to the variables in B. It is clear that $cs_B(f)$ represents the set of distinct columns in the decomposition chart [1] or the set of compatible classes [13].

Given a function $f(X)$ and a bound set B, the *disjoint decomposition* of $f(X)$ under the bound set B is $f(X)=g(y_1(B),y_2(B),\cdots,y_t(B),x_{b+1},\cdots,x_n)$, where $t<b$. When $t=1$, it is a *simple* disjoint decomposition. Let $Y=\{y_1,y_2,\ldots,y_t\}$ and $Y(B)=\{y_1(B),y_2(B),...,y_t(B)\}$. Functions $y_1(B)$ to $y_t(B)$ are called the *encoding functions* of the decomposition. The condition for the existence of a disjoint decomposition of $f(X)$ is given in the next theorem.

Theorem 1 [1, 7] $f(X)$ has a disjoint decomposition $g(Y(B),X-B)$ under the bound set B if and only if $|cs_B(f)|\le 2^t$ where $|Y(B)|=t$.

To compute a disjoint decomposition of $f(X)$ under a bound set B, we employ the *reduced ordered binary decision diagram* (OBDD) representation of functions [2] and use efficient functional decomposition approaches in [3, 9] based on cut computation in OBDDs.

3.2. Boolean Matching for XC4000 CLBs

We decompose wide functions to match the XC4000 CLB in three possible configurations A, B, and C, as shown in Figure 2, The bridging of inputs to LUTs F and G is represented by dotted lines in the configurations. In Configuration A, the H1 line (Figure 1(a)) is not used (i.e., connecting to 0/1 values). LUT inputs are not bridged in the configuration A.1 while they are bridged in the configuration A.2. In Configuration B, the H1 line is used but not bridged. The configuration B.1 is a special case of the configuration B.2 under the condition that LUT F has only one input and none of the inputs are bridged. In Configuration C, the H1 line is used as well as bridged. The H1 line is bridged with inputs to both LUTs F and G in the configuration C.1 while it is bridged with one LUT G input in the configuration C.2. It is easy to see that Configurations A, B, and C exhaust all possible ways of using the H1 line and the bridging of LUT inputs. As a result, we can determine exactly the feasibility of XC4000 CLBs for wide functions by matching the functions to these configurations.

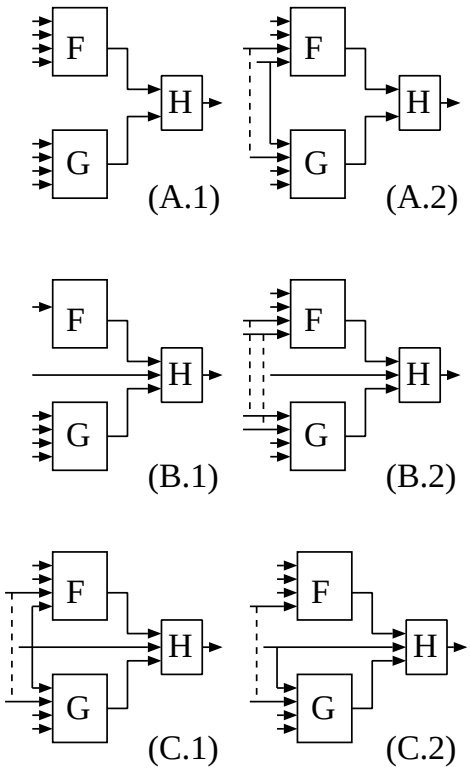

Figure 2 Three XC4000 CLB configurations for wide functions. The H1 line is not used, used but not bridged, and used as well as bridged in Configurations A, B, and C, respectively.

3.2.1. XC4000 CLB in Configuration A

Let (X_1,X_2) be a partition of X. A *disjoint bi-decomposition* of $f(X)$ is to represent $f(X)$ as $f(X)=g(y_1(X_1),y_2(X_2))$. Let (X_1,X_2,X_3) be a partition of X. A *non-disjoint bi-decomposition* of $f(X)$ is to represent $f(X)$ as $f(X)=g(y_1(X_1,X_3),y_2(X_2,X_3))$. A function $f(X)$ can be implemented with a XC4000 CLB in Configuration A if $f(X)$ has a bi-decomposition such that $|sup(y_1)|\le 4$ and $|sup(y_2)|\le 4$. The inputs in X_3 are the bridged inputs in the implementation. In general, X_3 may contain multiple inputs. The following two theorems give necessary and sufficient conditions for the existence of bi-decompositions of functions.

Theorem 2 [1] $f(X)$ has a disjoint bi-decomposition $g(y_1(X_1),y_2(X_2))$ if and only if $f(X)$ has simple disjoint decompositions under the bound set of X_1 and the bound set X_2.

Theorem 3 [12] Let $X_i=X-\{x_i\}$ and (X_1,X_2) be a partition of X_i. $f(X)$ has a non-disjoint bi-decomposition $g(y_1(X_1,x_i),y_2(X_2,x_i))$ if and only if $f_{\bar{x}_i}$ and f_x have disjoint bi-decompositions $f_{\bar{x}_i}(X_i)=g_0(y_{01}(X_1),y_{02}(X_2))$ and $f_{x_i}(X_i)=g_1(y_{11}(X_1),y_{12}(X_2))$, respectively, such that $g_0(y_1,y_2)=g_1(y_1,y_2)$.

To determine if a function $f(X)$ can be implemented with a XC4000 CLB in the configuration A.1, we try all possible partitions of X into X_1 and X_2 such that $|X_1|\le 4$ and $|X_2|\le 4$, and test if $f(X)$ satisfies Theorem 2. For the configuration A.2, we partition X_i into X_1 and X_2 with respect to each $x_i\in X$ exhaustively, and test if $f(X)$ satisfies Theorem 3. Both tests can be reduced to the simple disjoint decompositions of logic functions, which can be performed according to Theorem 1 using the OBDD representation of $f(X)$ efficiently.

Note that the disjoint bi-decomposition of each cofactor $f_{\bar{x}_i}(X_i)$ and $f_{x_i}(X_i)$ is not unique in the non-disjoint bi-decomposition of $f(X)$. In order to determine if $g_0=g_1$ for some bi-decompositions of $f_{\bar{x}_i}$ and f_{x_i}, after we obtain the bi-decomposition $f_{\bar{x}_i}=g_0(y_{01}(X_1),y_{02}(X_2))$, we invert $y_{01}(X_1)$, or $y_{02}(X_2)$, or both of them, and compute the g_0 functions with inverted inputs accordingly. In total, we obtain four equivalent bi-decompositions of $f_{\bar{x}_i}$. Then we compare each resulting function g_0 with g_1 for a match. Since $|sup(g)|=2$, we can easily exhaust all four possible bi-decompositions of $f_{\bar{x}_i}$. As a result, our approach is exact.

3.2.2. XC4000 CLB in Configuration B

A 6-input function $f(X)$ can be implemented with a XC4000 CLB in the configuration B.1 if $f(X)$ has a simple

disjoint decomposition $f(X)=g(y_1(X_1), x_i, x_j)$ where $x_i, x_j \notin X_1$. Clearly, the bound set X_1 contains four inputs. To decompose $f(X)$, we enumerate 4-input subsets of X as the bound set and compute a simple disjoint decomposition of $f(X)$.

Before we proceed to the configuration B.2, we introduce the *partially-dependent decomposition*. In general, each encoding function y_i in a decomposition $f(X)=g(Y(B),X-B)$ satisfies $sup(y_i)=B$. An encoding function is *partially-dependent* on B if $sup(y_i)\subset B$. The functional decomposition that produces partially-dependent encoding functions is called a *partially-dependent decomposition*. An extreme case is the *non-disjoint decomposition* where some encoding function y_i has $|sup(y_i)|=1$. A few approaches [10, 8, 5] have been proposed in the past to compute partially-dependent decompositions. In particular, a necessary and sufficient condition for the existence of partially-dependent encoding functions was given in [5]. To obtain a maximum number of partially-dependent encoding functions in a decomposition, the set of *all* partially-dependent encoding functions is computed first, and *assignable* encoding functions are chosen from the set to construct such a decomposition [10]. Note that partially-dependent encoding functions may or may not share their inputs. Such a procedure has been implemented based on the existence condition in [5].

We now match functions to the configuration B.2. Let $X_i=X-\{x_i\}$. Recall that $cs_{X_i}(f)$ represents the set of distinct cofactors of $f(X)$ with respect to variables in X_i. We consider two cases (i) $|cs_{X_i}(f)|=3, 4$, or (ii) $|cs_{X_i}(f)|=2$. (It is impossible that $|cs_{X_i}(f)|>4$.) In case (i), the decomposition of $f(X)$ under the bound set X_i requires exact two encoding functions (according to Theorem 1). As a result, $f(X)$ can be implemented with a XC4000 CLB in the configuration B.2 if $f(X)=g(y_1(X_1), y_2(X_2), x_i)$ such that $|X_1|, |X_2|\le 4$. Since in general $sup(y_i)=X_i$ and $|X_i|=|X|-1>4$, the decomposition is a partially-dependent decomposition. We can test the existence of such decompositions using the procedure developed in [5]. By selecting x_i enumeratively from X, we can conclude the existence of matching to configuration B.2 under case (ii).

In case (ii), however, the decomposition of $f(X)$ under the bound set X_i will be $f(X)=g(y_1(X_i), x_i)$ where $|X_i|>4$, which can not be matched directly to any considered configurations. However, we can use one more encoding function to obtain the condition that both encoding functions are partially-dependent. This is an important step in characterizing wide functions for XC4000 CLBs exactly. In particular, let $X_1, X_2 \subset X_i$ and $X_1 \cup X_2 = X_i$. A function $f(X)=g(y_1(X_i),x_i)$ can be represented as $f(X)=g'(z_1(X_1),z_2(X_2),x_i)$ if the following condition holds.

Theorem 4 Let $X_i=X-\{x_i\}$, $X_1, X_2 \subset X_i$ and assume $f(X)=g(y_1(X_i),x_i)$ under the bound set X_i. Then $f(X)$ can be represented as $f(X)=g'(z_1(X_1),z_2(X_2),x_i)$ if and only if $y_1(X_i)$ has a bi-decomposition $y_1(X_i)=h(z_1(X_1),z_2(X_2))$.

Proof See Appendix.

Based on Theorem 4, $f(X)=g(y_1(X_i),x_i)$ can be implemented with a XC4000 CLB in the configuration B.2 if $y_1(X_i)$ has a bi-decomposition $h(z_1(X_1),z_2(X_2))$ such that $|X_1|, |X_2|\le 4$ (Configuration A). Note that in both cases (i) and (ii), $X_1 \cap X_2 \neq \varnothing$ is possible. As a result, we have considered the bridging of inputs to LUTs F and G.

3.2.3. XC4000 CLB in Configuration C

Let $X_i=X-\{x_i\}$, $X_1, X_2 \subset X_i$, and $X_1 \cup X_2 = X_i$. ($X_1 \cap X_2 \neq \varnothing$ is possible). A function $f(X)$ can be implemented with a XC4000 CLB in the configuration C.1 if $f(X)=g(y_1(X_1,x_i),y_2(X_2,x_i),x_i)$ where $|X_1|, |X_2|\le 3$. The following theorem can be used to test the existence of matching to the configuration C.1.

Theorem 5 Let $X_i=X-\{x_i\}$, $X_1, X_2 \subset X_i$, and $X_1 \cup X_2 = X_i$. $f(X)$ has a decomposition $f(X)=g(y_1(X_1,x_i),y_2(X_2,x_i),x_i)$ if and only if $f_{\overline{x}_i}$ and f_{x_i} have bi-decompositions $f_{\overline{x}_i}(X_i)=g_0(y_{01}(X_1),y_{02}(X_2))$ and $f_{x_i}(X_i)=g_1(y_{11}(X_1),y_{12}(X_2))$, respectively.

Proof See Appendix.

A function $f(X)$ can be implemented with a XC4000 CLB in the configuration C.2 if $f(X)$ can be represented as $f(X)=g(y_1(X_1,x_i),y_2(X_2),x_i)$ such that $|X_1|<3$ and $|X_2|\le 4$. The following theorem can be used to test the existence of matching to the configuration C.1.

Theorem 6 Let $X_i=X-\{x_i\}$, $X_1, X_2 \subset X_i$, and $X_1 \cup X_2 = X_i$. $f(X)$ has a decomposition $f(X)=g(y_1(X_1,x_i),y_2(X_2),x_i)$ if and only if $f_{\overline{x}_i}$ and f_{x_i} have bi-decompositions $f_{\overline{x}_i}(X_i)=g_0(y_{01}(X_1),y_{02}(X_2))$ and $f_{x_i}(X_i)=g_1(y_{11}(X_1),y_{12}(X_2))$, respectively, such that $y_{02}(X_2)=y_{12}(X_2)$.

Proof See Appendix.

To implement $f(X)$ with a XC4000 CLB in Configuration C, we select $x_i \in X$ and a subset $X_1 \subset X_i$ where $|X_1|\le 3$ enumeratively, and compute bi-decompositions of cofactors $f_{\overline{x}_i}$ and f_{x_i} to satisfy Theorem 5 and Theorem 6 using the basic decomposition procedures.

Note that the bi-decomposition is not unique with respect to each pair of X_1 and X_2. For the configuration C.2, after we obtain a bi-decomposition $g_0(y_{01}(X_1), y_{02}(X_2))$ of $f_{\bar{x}_i}$, we compute the second one by inverting the encoding function $y_{02}(X_2)$. Then we compare if $y_{02} = y_{12}$ according to Theorem 6.

3.3. Boolean Matching for PLB1

For all functions implemented with the PLB1 architecture, the select input x_i can always stand by itself. This is because for any function $f(X) = \bar{x}_i g_1(X_1, x_i) + x_i g_2(X_2)$, we always have $f(X) = \bar{x}_i h(X_1) + x_i g_2(X_2)$ where $h(X_1) = g_1(X_1, 0)$. Let K_F and K_G represent the input sizes of LUTs F and G in PLB1, respectively. The next theorem gives a necessary and sufficient condition for implementing $f(X)$ with a PLB1.

Theorem 7 A function $f(X)$ can be implemented with a PLB1 if and only if either $|sup(f_{\bar{x}_i})| \le K_F$ and $|sup(f_{x_i})| \le K_G$ or $|sup(f_{\bar{x}_i})| \le K_G$ and $|sup(f_{x_i})| \le K_F$ holds for some $x_i \in X$.

Proof See Appendix.

3.4. Boolean Matching for PLB2

Obviously, PLB2 can not implement functions of more than six inputs, nor guarantee an implementation of every 5-input function. When there is no bridging of inputs, a function $f(X)$ can be implemented with a PLB2 if $f(X)$ can be represented as (i) $f(X) = g(y_1(X_1), x_i)$ or (ii) $f(X) = g(y_1(X_1), x_i, x_j)$, both are simple disjoint decompositions with the bound set $|X_1|$. For 5-input functions, $|X_1| = 4$ for case (i) and 3 for case (ii), and $|X_1| = 4$ for 6-input functions. When the bridging of inputs is considered, $f(X)$ must be represented as $f(X) = g(y_1(X_1), x_i, x_j)$ where $x_i \in X_1$. This is in fact a non-disjoint decomposition of $f(X)$ with an encoding function $y_2(X_1) = x_i$. We don't consider the case where both $x_i, xsubj \in X_1$ since it makes a 4-input function that can be implemented with LUT G alone. By enumerating the 3-input and 4-input subsets of X, we can match functions to PLB2 exactly with the methods introduced in the previous subsections.

3.5. Boolean Matching for PLB3

It should be clear that the same methods used for matching functions to XC4000 CLBs in Configuration C can be used for PLB3, except that LUTs F and G may have different input sizes.

Circuits	5-cuts	6-cuts	7-cuts
5xp1	285	422	734
9sym	651	1256	2333
9symml	653	1156	2195
C499	3694	9716	27599
C880	1815	3969	9079
alu2	2696	6666	18231
alu4	5889	14841	40332
apex6	1960	2691	4370
apex7	444	743	1342
count	133	94	87
des	28875	65245	157028
duke2	2115	4606	10106
misex1	155	266	400
rd84	1181	2351	5307
rot	2283	4857	10362
vg2	201	410	850
z4ml	44	36	36
total	53074	119325	290391

Table 1 Numbers of 5-cuts, 6-cuts, and 7-cuts in MCNC benchmarks.

4. Experimental Results

We implemented our Boolean matching methods in C language and incorporated it into the RASP logic synthesis system for FPGAs [6]. We evaluated PLB functional capability by counting the number of 5-cuts, 6-cuts, and 7-cuts that PLB can implement in MCNC benchmarks. Our approach is as follows. All benchmark circuits are decomposed into 2-input networks. Let v be a node in the decomposed network N and $input(v)$ represents the set of fanins of v. Given a subgraph H of a network, let $input(H)$ denote the set of distinct nodes outside H which supply inputs to nodes in H. A *cone* C_v rooted at v is a subnetwork consisting of v and its predecessors such that if node $u \in C_v$, every path from u to v resides entirely in C_v. A cone C_v is a K-input cone if $|input(C_v)| = K$. Every K-input cone can be covered by a K-LUT. Let N_v represent the largest cone rooted at v. A *cut* in N_v is a partition $(X_v, \bar{X}_v)$ of N_v such that $\bar{X}_v$ is a cone rooted at v and X_v is $N_v - \bar{X}_v$. This cut is a *K-cut* if $\bar{X}_v$ is a K-input cone. Let $u_1, u_2 \in input(v)$. A cut in N_{u_1} and a cut in N_{u_2} can be combined to form a cut in N_v. As a result, starting from primary inputs toward primary outputs in a topological order, we can enumerate all K-cuts for each node in N [11]. Table 1 shows the number of 5-cuts, 6-cuts, and 7-cuts in each MCNC benchmarks. To evaluate the functional capability of a PLB, we enumerated the cuts, computed their functions, matched the functions to the PLB, and computed the percentage of functions that can be matched to the PLB.

In the first experiment, we matched 6-cuts and 7-cuts to the XC4000 CLB configurations A.1, A.2, C.1, and C.2,

respectively. Boolean matching for these configurations are all based on the bi-decomposition of functions. Note that for 6-cuts and 7-cuts, if $F_{A.1}$, $F_{A.2}$, and $F_{C.1}$ represent the set of cuts that can be implemented in configurations A.1, A.2, and C.1, respectively, we have $F_{A.1} \subset F_{A.2} \subset F_{C.1}$. In Table 2 and 3, we see the percentages increase from columns A.1 to A.2 to C.1. More than 90% of 6-cuts and more than 60% of 7-cuts can be implemented with XC4000 CLBs in the configuration C.1 on an average.

In Table 4, we matched 6-cuts and 7-cuts to XC4000 CLBs in configurations B.1 and B.2, respectively. Note that every 6-cut that can be implemented in the configuration B.1 can be implemented in the configuration B.2. More than 80% of 6-cuts can be implemented with XC4000 CLBs in the configuration B.1, and almost all of them can be implemented in the configuration B.2. For 7-cuts, 88% of them can be implemented with XC4000 CLBs in the configuration B.2.

	6-cuts			
Circuits	A.1	A.2	C.1	C.2
5xp1	25%	85%	86%	25%
9sym	51%	93%	93%	51%
9symml	55%	92%	92%	54%
C880	44%	87%	88%	45%
apex6	56%	95%	95%	50%
apex7	40%	92%	93%	40%
count	30%	59%	71%	29%
misex1	77%	89%	89%	77%
rd84	61%	91%	90%	57%
vg2	55%	97%	97%	40%
z4ml	14%	36%	36%	14%
average	51%	90%	91%	49%

Table 2 Functional capability of the XC4000 CLB in configurations A.1, A.2, and C.1, and C.2 for 6-cuts in MCNC benchmarks.

	7-cuts			
Circuits	A.1	A.2	C.1	C.2
5xp1	13%	50%	51%	13%
9sym	39%	69%	70%	38%
9symml	40%	72%	72%	40%
C880	26%	57%	58%	27%
apex6	37%	65%	66%	34%
apex7	19%	55%	56%	19%
count	14%	26%	39%	13%
misex1	69%	87%	87%	69%
rd84	45%	65%	65%	43%
vg2	33%	63%	65%	26%
z4ml	0%	0%	0%	0%
average	34%	62%	63%	33%

Table 3 Functional capability of the XC4000 CLB in configurations A.1, A.2, and C.1, and C.2 for 7-cuts in MCNC benchmarks.

In Table 5, we matched 5-cuts and 6-cuts to PLB1 with a variety of input sizes K_F and K_G. We consider the cases when (K_F,K_G) is (3,4), (4,4), or (4,5), respectively. Note that the PLB1 architecture is able to implement any 5-cut in the second (Xilinx XC5200 CLB) and third (K_F,K_G) combinations. It is interesting to see that by reducing the K_F from 4 to 3, the PLB1 architecture only loses marginally in its functional capability (100% to 98% for 5-cuts and 8% to 5% for 6-cuts). However, the SRAM area in LUTs are reduced by 25%. On the other hand, if we make the LUT G a 5-LUT, the PLB1 functional capability for 6-cuts increases substantially (from 8% to 98%), and is approaching the capability of the ORCA PFU for implementing 6-cuts.

In Table 6, we test the functional capability of the PLB2 architecture, which is simplified from the XC4000 CLB. Although PLB2 consumes the smallest SRAM area in LUTs among the four PLBs evaluated in this paper, the implementation of 5-cuts is not guaranteed. However, the results in Table 6 show that 96% of the 5-cuts can still be implemented with PLB2 by applying simple disjoint decomposition (SD) and 2% more can be implemented if non-disjoint decomposition (ND) is applied. PLB2 is capable of 85% of 6-cuts (SD).

In Table 7, we test the functional capability of the PLB3 architecture, which is also simplified from the XC4000 CLB. We consider two LUT input combinations $(K_F,K_G)=(3,4)$ and $(4,4)$. PLB3 in the second case is similar to the XC4000 CLB except that the H LUT has two inputs. The implementation of 5-cuts is not guaranteed by

	6-cuts		7-cuts
Circuits	B.1	B.2	B.2
5xp1	64%	96%	72%
9sym	78%	98%	90%
9symml	79%	98%	89%
C880	81%	99%	87%
apex6	87%	100%	93%
apex7	79%	97%	87%
count	84%	100%	71%
misex1	79%	95%	90%
rd84	81%	95%	87%
vg2	84%	99%	87%
z4ml	67%	97%	31%
average	81%	98%	88%

Table 4 Functional capability of the XC4000 CLB in configurations B.1 and B.2 for 6-cuts and 7-cuts in MCNC benchmarks.

Circuits	(3,4)		(4,4)	(4,5)
	5-cuts	6-cuts	6-cuts	6-cuts
5xp1	98%	38%	44%	99%
9sym	100%	13%	19%	100%
9symml	100%	10%	15%	100%
C499	92%	4%	4%	95%
C880	98%	4%	5%	99%
alu2	100%	7%	9%	99%
alu4	98%	6%	8%	98%
apex6	100%	8%	9%	100%
apex7	99%	8%	10%	99%
count	100%	16%	16%	100%
des	99%	4%	8%	99%
duke2	100%	2%	4%	100%
misex1	100%	12%	21%	100%
rd84	98%	4%	9%	100%
rot	99%	5%	7%	98%
vg2	100%	6%	8%	100%
z4ml	91%	0%	0%	97%
average	98%	5%	8%	98%

Table 5 Functional capability of the PLB1 architecture for 5-cuts and 6-cuts in MCNC benchmarks when $(K_F, K_G) =$ (3,4), (4,4), and (4,5).

Circuits	5-cuts		6-cuts
	SD	ND	SD
5xp1	87%	13%	64%
9sym	96%	3%	78%
9symml	93%	6%	79%
C499	97%	0%	88%
C880	98%	1%	81%
alu2	96%	3%	83%
alu4	96%	2%	81%
apex6	97%	3%	87%
apex7	94%	5%	79%
count	100%	0%	84%
des	95%	2%	87%
duke2	97%	0%	91%
misex1	96%	4%	79%
rd84	95%	1%	81%
rot	95%	3%	84%
vg2	98%	3%	84%
z4ml	100%	0%	67%
average	96%	2%	85%

Table 6 Functional capability of the PLB2 architecture for 5-cuts and 6-cuts in MCNC benchmarks through simple disjoint decomposition (SD) and non-disjoint decomposition (ND) of functions.

PLB3. From Table 7, we see most of the 5-cuts and 1/3 of the 6-cuts can be implemented with PLB3 when $K_F = 3$. For the case of $K_F = K_G = 4$, very high percentage of the 5-cuts and 6-cuts can be implemented.

The experimental results are summarized in Table 8. We evaluate the functional capability for each PLB through

Circuits	(3,4)		(4,4)	
	5-cuts	6-cuts	5-cuts	6-cuts
5xp1	85%	14%	97%	85%
9sym	94%	31%	100%	93%
9symml	92%	33%	99%	92%
C499	89%	38%	97%	85%
C880	89%	24%	97%	87%
alu2	91%	29%	98%	88%
alu4	84%	28%	96%	-
apex6	95%	35%	100%	95%
apex7	94%	19%	99%	92%
count	68%	14%	100%	59%
duke2	97%	53%	98%	96%
misex1	90%	52%	100%	89%
rd84	88%	32%	96%	91%
rot	92%	26%	96%	89%
vg2	99%	33%	100%	97%
z4ml	39%	0%	86%	36%
average	89%	32%	97%	89%

Table 7 Functional capability of the PLB3 architecture for 5-cuts and 6-cuts in MCNC benchmarks when $(K_F, K_G) =$ (3,4) and (4,4).

dividing the number of cuts implemented by the number of SRAM bits in the PLB. Among all four PLB architectures with a variety of input sizes for LUTs F and G, we see PLB2 obtains the largest number of 5-cuts and 6-cuts implemented for each LUT SRAM bit, and the second place belongs to PLB1 with $(K_F, K_G) = (3,4)$ (a large number of 5-cuts per bit) and PLB3 with $(K_F, K_G) = (4,4)$ (a large number of 6-cuts per bit). However, the XC4000 CLB provides good functional capability for 7-cuts.

5. Conclusions

In this paper, we present Boolean matching techniques for implementing wide functions with four complex programmable logic blocks: the XC4000 CLB and three other architectures PLB1, PLB2, and PLB3. Our

PLBs	Config. (K_F, K_G)	#bits	#cuts per SRAM bit		
			5-cut	6-cut	7-cut
XC4K	B.2	40	1327	2923	6389
PLB1	(3,4)	24	2167	249	-
	(4,4)	32	1659	298	-
	(4,5)	48	1106	2436	-
PLB2		24	2167	4226	-
PLB3	(3,4)	28	1687	1364	-
	(4,4)	36	1430	2950	-

Table 8 Functional capability per each SRAM bit of the four PLB architectures for 5-cuts, 6-cuts, and 7-cuts in MCNC benchmarks.

techniques are based on functional decomposition and are able to completely characterize wide functions for the four types of PLBs. We evaluate PLBs in term of their capability for wide functions. Experimental results show that the XC4000 CLB can implement 98% of 6-cuts and 88% of 7-cuts on an average, while PLB2 provides the largest number of 5-cuts and 6-cuts implemented per each LUT SRAM bit among the four PLB architectures. Our techniques and results are useful for developing future logic units as well as new technology mapping algorithms in LUT based FPGAs.

Acknowledgement

We would like to thank Prof. Marek-Sadowska for pointing the reference [12] and Prof. Sasao for providing an early version of his work. This work is partially supported by NSF Young Investigator (NYI) Award MIP-9357582, and grants from Xilinx, Quickturn, and Lucent Technologies under the California MICRO programs.

References

[1] Ashenhurst, R. L., ''The Decomposition of Switching Functions,'' *Proc. Int'l Symp. on Theory of Switching Functions*, 1959.

[2] Bryant, R. E., ''Graph-based Algorithms for Boolean Function Manipulation,'' *IEEE Trans. on Computers*, Vol. **C-35**, pp. 677-691, Aug. 1986.

[3] Chang, S.-C. and M. Marek-Sadowska, ''Technology Mapping via Transformations of Function Graphs,'' *Proc. IEEE Int'l Conf. on Computer Design*, pp. 159-162, Oct. 1992.

[4] Cong, J. and Y. Ding, ''Combinational Logic Synthesis for LUT Based Field Programmable Gate Arrays,'' *ACM Trans. on Design Automation of Electronic Systems*, Vol. **1, 2**, pp. 145-204, 1996.

[5] Cong, J. and Y.-Y. Hwang, ''Partially-Dependent Functional Decomposition with Applications in FPGA Synthesis and Mapping,'' *Proc. ACM 5th Int'l Symp. on FPGA*, pp. 35-42, Feb. 1997.

[6] Cong, J., J. Peck, and Y. Ding, ''RASP: A General Logic Synthesis System for SRAM-based FPGAs,'' *Proc. ACM 4th Int'l Symp. on FPGA*, Feb. 1996.

[7] Curtis, H. A., ''A Generalized Tree Circuit,'' *Journal of the ACM*, Vol. **8**(4) pp. 484-496, 1961.

[8] Huang, J.-D., J.-Y. Jou, and W.-Z. Shen, ''Compatible Class Encoding in Roth-Karp Decomposition for Two-Output LUT Architecture,'' *Proc. IEEE Int'l Conf. on Computer-Aided Design*, pp. 359-363, Nov. 1995.

[9] Lai, Y.-T., K.-R. R. Pan, and M. Pedram, ''FPGA Synthesis using Function Decomposition,'' *Proc. Int'l Conf. on Computer Design: VLSI in Computers*, pp. 30-35, Oct. 1994.

[10] Legl, C., B. Wurth, and K. Eckl, ''An Implicit Algorithm for Support Minimization during Functional Decomposition,'' *Proc. European Design and Test Conf.*, March 1996.

[11] Rudell, R., Private Communication 1996.

[12] Sasao, T. and J. T. Butler, ''On Bi-Decompositions of Logic Functions,'' *Proc. Int'l Workshop on Logic Synthesis*, 1997.

[13] Wurth, B., K. Eckl, and K. Antreich, ''Functional Multiple-Output Decomposition: Theory and an Implicit Algorithm,'' *Proc. ACM/IEEE Design Automation Conf.*, pp. 54--59, Jun. 1995.

Appendix: Proofs

[Theorem 4] (**if**) Because $y_1(X_i)$ has a bi-decomposition $y_1(X_i)=h(z_1(X_1),z_2(X_2))$, we have $f(X)=g(y_1(X_i),x_i)=g(h(z_1(X_1),z_2(X_2)),x_i)$ $=g'(z_1(X_1),z_2(X_2),x_i)$. (**only if**) Assume $f(X)$ can be represented as $f(X)=g'(z_1(X_1),z_2(X_2),x_i)$ as well as $f(X)=g(y_1(X_i),x_i)$. Consider $f_{x_i}=g(y_1(X_i),1)$. Because $g(y_1,1)=y_1$ or $-y_1$, we have $f_{x_i}=y_1(X_i)$ or $f_{x_i}=-y_1(X_i)$. Since $f_{x_i}=g'_{x_i}(z_1(X_1), z_2(X_2), 1)=h(z_1(X_1), z_2(X_2))$, we have $y_1(X_i)=h(z_1(X_1), z_2(X_2))$ or $y_1(X_i)=-h(z_1(X_1), z_2(X_2))$. In either case, $y_1(X_i)$ belongs to Category A. □

[Theorem 5] (**only if**) Because $f(X)$ has a decomposition $f(X)=g(y_1(X_1,x_i), y_2(X_2,x_i),x_i)$, we have $f_{\bar{x}_i}(X_i)=g_0(y_{01}(X_1),y_{02}(X_2))$ where $g_0=g_{\bar{x}_i}$, $y_{01}(X_1)=y_1(X_1,0)$, and $y_{02}(X_2)=y_2(X_2,0)$. Similarly, we have $f_{x_i}(X_i)=g_1(y_{11}(X_1),y_{12}(X_2))$ where $g_1=g_{x_i}$, $y_{11}(X_1)=y_1(X_1,1)$, and $y_{12}(X_2)=y_2(X_2,1)$. (**if**) Assume $f_{\bar{x}_i}(X_i)=g_0(y_{01}(X_1),y_{02}(X_2))$ and $f_{x_i}(X_i)=g_1(y_{11}(X_1),y_{12}(X_2))$. Define $y_1(X_1,x_i)=\bar{x}_i y_{01}(X_1)+x_i y_{02}(X_2)$, $y_2(X_2,x_i)=\bar{x}_i y_{11}(X_1)+x_i y_{12}(X_2)$, and $g=\bar{x}_i g_0+x_i g_1$. Then $f(X)=\bar{x}_i f_{\bar{x}_i}+x_i f_{x_i}$ can be represented as $f(X)=g(y_1(X_1,x_i), y_2(X_2,x_i),x_i)$. □

[Theorem 6] Since $x_i \notin X_2$, the proof is similar to that of Theorem 5 except that $y_{02}(X_2)=y_{12}(X_2)=y_2(X_2)$. □

[Theorem 7] Let the select line carry the input x_i or $\bar{x}_i$ and apply the Shannon expansion of $f(X)$ with respect to x_i. Then it is clear the theorem holds. □