Cut Ranking and Pruning: Enabling A General And Efficient FPGA Mapping Solution

Article · June 1999

DOI: 10.1145/296399.296425 · Source: CiteSeer

3 authors, including:

Jason Cong
University of California, Los Angeles
696 PUBLICATIONS **21,409** CITATIONS
SEE PROFILE

Chang Wu
University of California, Los Angeles
15 PUBLICATIONS **339** CITATIONS
SEE PROFILE

Cut Ranking and Pruning: Enabling A General And Efficient FPGA Mapping Solution

Jason Cong and *Chang Wu*
Department of Computer Science
University of California, Los Angeles, CA 90095
{cong,changwu}@cs.ucla.edu

Yuzheng Ding
Bell Laboratories
Lucent Technologies, Murray Hill, NJ 07974
yuzheng.ding@lucent.com

Abstract

Cut enumeration is a common approach used in a number of FPGA synthesis and mapping algorithms for consideration of various possible LUT implementations at each node in a circuit. Such an approach is very general and flexible, but often suffers high computational complexity and poor scalability. In this paper, we develop several efficient and effective techniques on cut enumeration, ranking and pruning. These techniques lead to much better runtime and scalability of the cut-enumeration based algorithms; they can also be used to compute a tight lower-bound on the size of an area-minimum mapping solution. For area-oriented FPGA mapping, experimental results show that the new techniques lead to over 160X speed-up over the original optimal duplication-free mapping algorithm, achieve mapping solutions with 5-21% smaller area for heterogeneous FPGAs compared to those by Chortle-crf [6], MIS-pga-new [9], and TOS-TUM [4], yet with over 100X speed-up over MIS-pga-new [9] and TOS-TUM [4].

1 Introduction

A K-lookup-table (K-LUT) is a basic functional block of lookup-table based FPGAs, which can implement any Boolean function with no more than K inputs. Homogeneous FPGAs have LUTs with the same number of inputs, while heterogeneous FPGAs consist LUTs with different input sizes. For example, Xilinx XC4K [15] has both 4- and 5-LUTs and ORCA2C [12] has both 5- and 6-LUTs (and even 4-LUTs with shared inputs). Vantis [13] recently announced a new type of FPGAs with 3-, 4-, 5- and 6-LUTs.

One generic type of FPGA mapping algorithms are based on cut enumeration. Since each K-LUT corresponds to a K-cut (to be precisely defined later), these algorithms construct a mapping solution by enumerating and evaluating all (or a subset of) K-cuts for every node and choosing the best K-cuts for a subset of nodes to form LUTs to cover the original circuit. For example, the algorithms in [9] enumerate a few cuts for each node and solve the cut selection problem as a binate covering problem. The DFmap algorithm [2] enumerates all the K-cuts within the maximum fanout-free cone (MFFC) of each node and select one with the minimum area to form an area-minimum duplication-free mapping solution. For delay minimization of sequential circuits, the work in [10] enumerates all the K-cuts and select one with the minimum label for each node to compute an optimal mapping solution under retiming. In general, cut-enumeration based approaches have the advantage that they can handle different optimization objectives and can be easily extended for heterogeneous FPGAs or FPGAs with some special structures. As a comparison, other existing approaches, especially those based on max-flow computation [1] may not be easily extended to handle special features in FPGA architecture, such as non-uniform pin-to-pin delays of an LUT.

However, the most serious problem in cut-enumeration based approaches is high time complexity. For a circuit with n nodes, the number of K-cuts (or different K-LUT formations) for a node can be as large as $O(n^K)$. Furthermore, evaluating and selecting the best K-cuts (or K-LUTs) to cover the entire circuit is difficult. [9] formulates the cut selection problem as a binate-covering problem which is NP-hard. DFmap algorithm [2] can find optimal duplication-free solution in polynomial time. However, it cannot consider duplication of nodes to reduce the area.

In this paper, we develop several efficient and effective techniques on cut enumeration, ranking and pruning. These techniques lead to much better runtime and scalability of the cut-enumeration based algorithms. They can also be used to compute a tight lower-bound on the size of an area-minimum mapping solution (finding such a solution is a NP-hard problem). We have applied these techniques to several applications, including (1) a faster area-optimal duplication-free mapping algorithm; (2) area-minimal mapping for general mapping with possible node duplication; and (3) extensions of (1) and (2) to heterogeneous FPGA mapping. Experimental results show significant improvements over existing approaches.

The remainder of the paper is organized as follows. Section 2 presents a brief problem formulation and basic definitions. Section 3 discusses our cut enumeration, ranking and pruning techniques. Section 4 presents applications of our techniques for area-oriented mappings. Our experimental results are reported in Section 5 and discussions are in Section 6. Proofs of the theorems and some other details are omitted due to page limit.

2 Problem Formulation and Definitions

In this work, we consider the FPGA technology mapping problem for combinational circuits.[1] A general circuit is modeled as a directed acyclic graph in which nodes represent logic elements, and edges represent interconnects. A node

[1] For sequential circuits, we can cut at the flipflops to get combinational subcircuits and map each one separately.

without incoming edge is a *primary input* (PI) and a node without outgoing edge is a *primary output* (PO).

A *homogeneous* FPGA consists of only one type of LUTs with the same input size, while a *heterogeneous* FPGA consists of LUTs with different input sizes. LUTs of different input sizes may have different areas. We use K to denote the input size of an LUT under discussion. The area-oriented LUT-based FPGA mapping problem is to *cover* a gate-level network with LUTs such that the *total area* of those LUTs is minimum for either homogeneous or heterogeneous FPGAs. This problem is NP-hard even for homogeneous FPGAs [5].

In this paper, we use C_v to denote a *cone* of node v, FFC_v a *fanout-free cone* of v, $MFFC_v$ a *maximum fanout-free cone* of v and F_v the *fanin cone* of v. We refer readers to [3] for the detailed definitions. A cone is *K-feasible* if its input size is K or smaller.

A *cut* is a partitioning $(X,\ \overline{X})$ of a cone C_v such that $\overline{X}$ is a cone of v. The *node cut-set* of the cut, denoted $V(X, \overline{X})$ consists of the inputs of cone $\overline{X}$. A cut is K-feasible if $\overline{X}$ is a K-feasible cone. We may simply refer such a partition $(X,\ \overline{X})$ as a *cut for node v*, when discussing a fixed C_v.

3 Cut Generation, Ranking and Pruning

The procedure of selecting a set of LUTs of proper sizes to cover the gate-level network has three components:

1. *Cut generation*: For each node being considered, generate all or a subset of the K-feasible cuts, via enumeration or direct computation.
2. *Cut ranking*: Compare the cuts according to the given optimization objectives when two or more cuts are generated for a node.
3. *Node selection*: Choose the nodes (and their best cuts) for implementation using LUTs.

Each of these three steps may involve costly computations, due to the potentially large number of cuts and the NP-hardness of the area minimization problem. Existing algorithms generally alleviate this by heuristically restricting the search scope, which often result in sub-optimality.

The set of general techniques that will be presented here can significantly speed up the cut generation and selection processes for area minimization in FPGA mapping. These techniques are general in the sense that they are applicable to any cut enumeration process, whether through the entire solution space or a subset. They are also scalable since the amount of computation per node can be made independent of the size of the entire network and only associated with the size of its input and the LUT size. Applications and experimental results to be presented in subsequent sections will show that they are also very effective.

3.1 Cut Generation

We first consider the enumeration of all K-feasible cuts of a cone for a given node, then discuss how to share such computation among nodes in a network for better efficiency.

3.1.1 Cut Enumeration Scheme

Given a node v and a cone C_v of v, we associate a Boolean variable to each node $w \in C_v \cup input(C_v)$, also denoted as w. A cut $(X_v,\ \overline{X}_v)$ of C_v can be represented using a product term (or a *p-term* in short) of the variables associated with the nodes in the node cut-set of $V(X_v,\ \overline{X}_v)$. In the rest of this section we will often refer to a cut by the p-term associated with it. A set of cuts can be represented by a sum-of-product expression using the corresponding p-terms. In this way we can represent the set of all K-feasible cuts of C_v by a unate Boolean function which we call the *generating function* of the cuts.

For a node $w \in C_v$, $F_w \cap C_v$ is a cone of w, which we call the *projection* of w in C_v and is denoted as $P_w^{C_v}$ or simply P_w when C_v is clearly understood in context. If $w \in input(v)$ but $w \notin V(X_v,\ \overline{X}_v)$, then $(X_v,\ \overline{X}_v)$ derives a cut $(X_w,\ \overline{X}_w)$ of P_w where $\overline{X}_w = \overline{X}_v \cap P_w$ and $V(X_w,\ \overline{X}_w) \subseteq V(X_v,\ \overline{X}_v)$. We call $(X_w,\ \overline{X}_w)$ a *subcut of v from w*. If we extend this notion of subcut to nodes $w \in input(v) \cap V(X_v,\ \overline{X}_v)$ by saying that w itself (as the Boolean variable) represents a subcut of v from node w, it is easily seen that a cut of v can be obtained by merging one subcut from each of its inputs together. By considering all possible combinations of all subcuts, we will be able to generate all cuts. This implies a recursive definition of the generating function. Specifically, let $f_c(K,\ v,\ w)$ for $w \in C_v$ be the generating function for all K-feasible cuts of $P_w^{C_v}$, and define $f_c(K,\ v,\ w) = 0$ for $w \in input(C_v)$. Then, we can show

Lemma 1

$$f_c(K,\ v,\ w) = \bigotimes_{u \in input(w)}^{K} [u + f_c(K,\ v,\ u)], \qquad (1)$$

where operator $+$ is Boolean OR, and $\otimes^K$ is Boolean AND but filtering out all resulting p-terms with more than K variables.

All K-feasible cuts of C_v are then enumerated by $f_c(K,\ v,\ v)$.

3.1.2 Efficient Computation via Subcut Sharing

The above formulation applies directly to any node v and any cone C_v of it in a network. Since $f_c(K,\ v,\ w)$ is a function related to $P_w^{C_v}$, generally the computation can only take place once C_v is given, and intermediate results can not be directly shared among cones, except for some particular types of cones.

Cuts of Fanin Cones. If we want to enumerate all K-feasible cuts of all nodes in a network, for every node w we will compute $f_c(K,\ w,\ w)$ for $C_w = F_w$. Note that, however, for any node v such that $w \in F_v$, $P_w^{F_v} = F_w$. That is, P_w remains the same for all successors of w. Therefore Eqn. (1) simplifies to

$$f_c(K,\ w,\ w) = \bigotimes_{u \in input(w)}^{K} [u + f_c(K,\ u,\ u)], \qquad (2)$$

and can be iteratively computed in a topological order in one pass for all nodes.

Cuts of MFFCs. Enumerating cuts of MFFCs is more complicated since in general $P_w^{MFFC_v} \neq MFFC_w$. To use the one-pass method, for each node w, instead of $f_c(K,\ w,\ w)$ we compute $f_c(K,\ v,\ w)$, such that $MFFC_v$ is the *largest* MFFC containing w. In this case $f_c(K,\ v,\ w)$ is the generating function of the K-feasible cuts of $P_w^{MFFC_v}$, and $f_c(K,\ w,\ w)$ can be easily derived from it. According to the nesting property of MFFCs [2], we can easily show

Lemma 2 *If* $w \in MFFC_u$ *and* $u \in MFFC_v$, *then* $P_w^{P_u^{MFFC_v}} = P_w^{MFFC_v}$.

3.2 Cut Ranking

Once a cut is generated it has to be evaluated or ranked. In the case of FPGA mapping for area minimization the evaluation is based on the effect of implementing this cut on the overall area of the mapping solution: a cut is more favorably ranked if, when used to implement this node, it results in smaller mapped area of the entire network (or a given portion of it).

Such ranking is generally not easy, because it effectively requires computing an optimal mapping of the network (or a portion of it) with one given LUT fixed.

3.2.1 Duplication-free Mapping

In the case of duplication-free mapping, it is shown in [2] that an optimal mapping can be partitioned into a set of MFFCs, and the minimum mapped area for each MFFC can be computed independently.

First, consider a *mapped* $MFFC_v$ in which each node is an LUT. Assume the area of a K-LUT is A_K (which does not have to be always different for different values of K). We define the *effective area* $\alpha()$ for each node and edge of the network as follows. First, for any node w outside of the MFFC we have $\alpha(w) = 0$. Then iteratively, for each edge $< u, w >$ we have $\alpha(< u, w >) = \alpha(u)/ \mid output(u) \mid$, and for each node w inside the MFFC we have $\alpha(w) = \sum_{u \in input(w)} \alpha(< u, w >) + A_{\mid input(w) \mid}$. Intuitively, the effective area of a node w is the area of $P_w^{MFFC_v}$, except that the portions that are shared with other projection cones are divided proportionally and distributed into each cone containing them. A shared portion is always a cone rooted at a multiple output node; in duplication-free mapping it will have been mapped independently, and once the outputs reconverge, the total area of that shared cone will be accounted for. Based on these observations, we can easily shown

Lemma 3 *The effective area of* v, *the root of* $MFFC_v$, *is the real area of* $MFFC_v$ *under the given mapping solution.*

This lemma is true for *any* mapping solution of $MFFC_v$, including any optimal solution. If the mapping solution is optimal, for any node w its projection cone $P_w^{MFFC_v}$ must also be optimally mapped, including its exclusive portion as well as the shared portion (which will be mapped independently in a duplication-free mapping). Therefore the effective area of w will be minimum with respect to all possible mapping solutions of $P_w^{MFFC_v}$. We call this minimum effective area the *effective cost* of w, denoted $\gamma(w)$. (The effective cost for nodes outside $MFFC_v$ is still 0.) Obviously, we have

Corollary 1 *The effective cost of* v, *the root of* $MFFC_v$, *is the minimum area of any mapping solution of* $MFFC_v$.

The importance of effective cost is that we can use it to efficiently compute an optimal duplication-free mapping of $MFFC_v$ including the evaluation of cuts in an MFFC. We do so by extending it to any cut $(X_w, \overline{X}_w)$ of $P_W^{MFFC_v}$. The effective cost of the cut is the sum of the effective cost of the nodes in $input(w) \cap V(X_w, \overline{X}_w)$, and the effective cost of the induced cuts of nodes that are in $input(w)$ but not in $V(X_w, \overline{X}_w)$. With the effective cost of a cut so defined, we can compute the effective cost of the nodes according to the following rule.

Theorem 1 *For any node* $w \in MFFC_v$,

$$\gamma(w) = \min_{\substack{(X_w, \overline{X}_w) \text{ is a} \\ K\text{-feasible cut of } MFFC_w}} [\gamma(X_w, \overline{X}_w) + A_{\mid V(X_w, \overline{X}_w) \mid}]. \quad (3)$$

Moreover, the cut that minimizes the $\gamma(w)$ *of this equation is the best* K*-feasible cut of* $MFFC_w$.

If we compute the effective cost of each node in $MFFC_v$ in a topological order, for any node $u \in P_w^{MFFC_v}$ the cost of u as well as the cost of each cut of u would have been computed when $\gamma(w)$ is to be computed. For a K-feasible cut the effective cost $\gamma(X_w, \overline{X}_w)$ can be evaluated in $O(K)$ time, superior to the $O(n)$ method used in [2].

3.2.2 General Mapping

We can extend the effective area computation to general mapping directly. For effective area under a given mapping solution, we still define $\alpha(v) = 0$ for any PI node v, $\alpha(< u, v >) = \alpha(u)/ \mid output(u) \mid$ for any edge $< u, v >$, and $\alpha(v) = \sum_{u \in input(v)} \alpha(< u, v >) + A_{\mid input(v) \mid}$. Similarly, we still define the *effective cost* $\gamma(v)$ of a node v to be the minimum effective area of v in any mapping solution.

Computing effective cost is more complicated. Note that logic sharing among fanin cones are handled by distributing the effective cost of the shared portion, which corresponds to an optimal mapping of that part, evenly to all fanouts. By doing so it implicitly assumes that the optimal mapping of the shared portion, obtained independently, will be consistently used in the optimal mapping of the cones that share it. While this is true for duplication-free mapping, it is not true for general mapping, as illustrated in Figure 1, where node w is duplicated in mapping of F_u but not in mapping of F_v. In this case, the accurate distribution of such cost becomes complicated. In fact, the difficulty in determining how to optimally map a multiple-output node is the reason that FPGA mapping for area minimization is NP-hard [5].

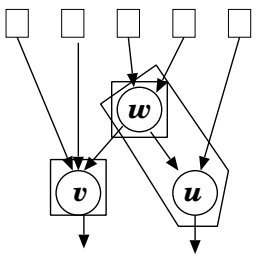

Figure 1: General Mapping ($K = 3$) treats shared logic differently.

To maintain efficiency, we still want to use the cost computation method used in duplication-free mapping. Note that, however, what we computed is no longer accurate effective cost. Let $\hat{\gamma}()$ denote the cost we computed for general mapping, we can prove that

Theorem 2 *For any node* v *in the network,* $\hat{\gamma}(v) \leq \gamma(v)$.

As a result we can compute a *lower bound* of the minimum mapped area of the network:

Corollary 2 $\sum_{v \text{ is a PO}} \hat{\gamma}(v)$ *is a lower bound on the minimum area of all mapping solutions.*

Our experimental results also show that using $\hat{\gamma}(v)$ as the evaluation function for cut ranking is very effective.

3.3 Cut Pruning

Based on the ranking techniques we can select the most desired cuts during the enumeration. On the other hand, it is more efficient if we can eliminate undesired cuts without even generating them. In this section we discuss various pruning techniques that speed up the search.

3.3.1 Non-essential Subcuts

We first consider cut enumeration for duplication-free mapping. In this case, the cuts of $MFFC_v$ are constructed partly with subcuts from w where $w \in MFFC_v$ (recall that a *subcut* of v from w is a cut of $P_w^{MFFC_v}$ induced by a cut of v). In particular, some subcuts from w are also cuts of $MFFC_w$; we call them *internal subcuts* from w. the number of nodes in the node cut-set of an internal subcuts from w that are in $MFFC_w$ is called the *internal size*.

A subcut can participate in the construction of many cuts, which may be ranked differently. However, if no cut constructed from a subcut S_w can have the best rank, S_w is said to be *non-essential*. A non-essential subcut can be discarded without losing the best cut, thus speed up the enumeration.

If we refine the ranking standard introduced in the preceding section such that a cut is ranked better if its cost is smaller or with the same cost has a smaller cut size, we have the following results regarding non-essential internal subcuts.

Theorem 3 *Let $w \in MFFC_v$ and S_w is an internal subcut of w from w. S_w is non-essential if*

(1) S_w is not a minimum cost cut of $MFFC_w$; or

(2) there is another internal subcut of the same cost, whose size is smaller than the internal size of S_w.

In each of the two cases, for every cut of v constructed from S_w there will be another cut of v constructed from S_w that either has smaller cost or has the same cost but smaller size. Note that if S_w is not a minimum cost cut for w, its node cut-set must contain at least one node inside $MFFC_w$.[2] By replacing the node cut-set of S_w with node w in any cut of $MFFC_v$, we will not increase the size of the cut; moreover, we will not increase the cost of the cut since $MFFC_w$ can now be optimally implemented, which will save at least one LUT, thus offset the added cost of implementing w.

Based on these properties we can set a few rules:

If $MFFC_w$ is K-feasible, then ($\emptyset$, $MFFC_w$) is the only best cut, and the only internal subcut needed for subsequent cut generation of the successor nodes of w.

If $MFFC_w$ is not K-feasible, only certain minimum cost cuts of $MFFC_w$ need to be kept as internal subcuts for subsequent cut generation of successor nodes of w; in particular,

- if there are minimum cost cuts of $MFFC_w$ whose node cut-sets are completely contained in $MFFC_w$, keep only the one with smallest cut size; or
- if there are minimum cost cuts whose node cut-sets are not completely contained in $MFFC_w$, keep only the ones whose internal sizes are smaller than the size of any known minimum cost cut.

[2] If S_w covers the entire $MFFC_w$, it must be the best cut.

Using these rules can usually limit the number of stored cuts to a very small value, thus significantly reduce the amount of computation in Eqn 1.

These reductions are also applicable to general mapping. Note that the properties do not depend on the fact the $MFFC_v$ is an MFFC; they are solely based on $MFFC_w$ being an MFFC. Since cuts of F_w include cuts of $MFFC_w$, we can reduce this set of subcuts using the above rules.

3.3.2 Common Subcuts

In the case of general mapping, multiple successors of a node v can use the subcuts from v to construct its own cuts. and if there are multiple subcuts that are equally desirable according to the (approximate) ranking, two different successors u_1 and u_2 may choose different subcuts form v to construct their desired cuts independent of each other. If u_2 and u_2 are both implemented by LUTs, however, nodes from both node cut-sets, including the nodes from the two different subcuts, must be implemented as well. This results in increased number of LUTs in the form of unnecessary logic duplication.

To alleviate this problem of excessive node duplication, for cuts with the same cost, we sort and select them in a predetermined order to avoid arbitrary selection of different min-cost cuts. To sort all the cuts, we first number all the nodes as $ID(v) = 1, 2, .., n$ for a circuit with n nodes and sort nodes in a cut in increasing order of $ID(v)$. Then we sort all the cuts $(v_1, v_2, .., v_k)$ with the same cost of a node based on a lexicographic order of $ID(v_i)$ of all nodes v_i in the cuts. If both cuts of each of the fanouts of v have the same cost, both of the fanouts tend to choose the same cut of v to form their best cuts. As a result, we tend to not duplicate the fanin cone of v and can achieve a mapping solution with smaller area. Notice that, however, this ordering is just a heuristic in helping to choose common cuts to reduce unnecessary node duplication. To solve this problem optimally, however, is NP-hard.

3.3.3 EFFC Based Enumeration

Another way to prevent excessive node duplication is to use explicit duplication control by marking certain nodes as *non-duplicable*. Once such nodes are marked, we generate only the cuts that do not cover those marked nodes. Specifically, we define *extended* MFFC of a node v, denoted $EFFC_v$ to be a cone of v containing $MFFC_v$ and any unmarked (duplicable) predecessor of v that has a path to v contained in $EFFC_v$. Then, instead of enumerating cuts in F_v, we do it in $EFFC_v$.

The general rule of selecting nodes that are to be marked non-duplicable is that they are likely to be implemented by LUTs. Examples include nodes with large fanout and roots of large MFFCs. We can also perform some preprocessing to determine non-duplicable nodes.

3.3.4 Cost Based Pruning

Even with subcut pruning we described, the number of saved cuts can still be very large especially when K is large. Therefore, we keep a fixed number of cuts with small estimated cost and prune those with larger estimated cost and in the case of equal cost, ones with larger cut sizes. Since the estimated cost of a cut is a lower-bound on the minimum effective cost on the LUT represented by the cut in any mapping solution, to prune cuts with large estimated cost is less likely

to degrade the final mapping results. This is also confirmed by our experimental data (to be presented in Section 6).

3.3.5 Decomposition vs. Shrinking

The complexity of Eqn. 1 clearly relates to the input size as well as the number of nodes: More nodes require most steps of computation, while larger input sizes result in slow computation per step. In [2] a *shrinking* method is introduced to reduce the number of nodes for duplication-free mapping. It collapses all K-feasible MFFCs prior to cut generation, after showing that the optimal solution would not be lost. It was very successful in speeding up the cut enumeration there [2].

The opposite direction is *decomposition*, which decreases input sizes by increasing the number of nodes. Decomposition can provide more freedom in forming different LUTs at the cost of more cuts to be enumerated. For area minimization, we first decompose the network into 2-bounded one. Then, for a gate v, we shrink $MFFC_v$ into one node if $MFFC_v$ is K-feasible. For heterogeneous FPGAs with LUTs of input sizes of $K_1 < K_2 < .. < K_c$, we only shrink K_1-feasible MFFCs. The reason is doing this is that decomposing a network into 2-bounded one can enlarge the solution space and potentially lead to better results. However, most of the K-feasible MFFCs usually will be packed into K-LUTs for smaller area. Shrinking those K-feasible MFFCs usually will not affect the results, while can reduce the number of K-cuts to be enumerated.

4 Applications on FPGA Mapping for Area Minimization

In this section, we applied the cut enumeration, ranking and pruning techniques discussed in Section 3 to FPGA mapping for area minimization. The first application is to speed up the optimal DFmap algorithm presented in [2]. The second application is to develop a general mapping algorithm with consideration of node duplication for area minimization. We shall also discuss how easily these algorithms can be extended to heterogeneous FPGA mapping.

4.1 Speedup of Optimal Duplication-Free Mapping

The DFmap algorithm [2] is the first pseudo polynomial time algorithm to compute optimal duplication-free mapping solutions. It first partitions the circuit into a set of MFFCs and then maps each one independently. For one MFFC rooted at a node v, after we select one K-feasible cut, the MFFC can again be partitioned into a set of small MFFCs and each one can also be mapped independently.

The major shortcomings of the approach are that, first, the number of all the K-feasible cuts for a node in its MFFC can grow exponentially with K. Second, to compute the cost of each cut for a node v needs $O(n)$ time by adding all the area of sub-MFFCs included in the MFFC of v.

To speed up the DFmap algorithm [2], we use the aforementioned cut enumeration, ranking and pruning techniques as follows.

For each $MFFC_v$ after the MFFC partitioning, we process all the nodes in a topological order from its inputs to its output. For each node, we enumerate its K-feasible cuts by combining the K-feasible cuts of its inputs. For an input i to the $MFFC_v$, its only K-feasible cut is a 1-cut with only i on the node cut-set. We rank each enumerated cut and prune those dominated ones. Since no node duplication is allowed, for a node u we will only select its best cut inside $MFFC_u$, i.e., we will select a cut $V(X, \overline{X})$ with the minimum cost and $\overline{X} \subseteq MFFC_u$. To check if $\overline{X} \subseteq MFFC_u$ holds for a cut, we first construct the $MFFC_u$ and mark all nodes inside $MFFC_u$ and all inputs to $MFFC_u$. A cut is *included* in $MFFC_u$ if all nodes in the node cut-set $V(X, \overline{X})$ are marked.

4.2 General FPGA Mapping for Area Minimization

For duplication-free mapping, we select a best cut for every node from K-cuts within the node's MFFC. For general mapping, however, we allow a cut to be selected from K-cuts within the node's EFFC. Recall that a node's EFFC is in general larger than its MFFC, with EFFC we can explore a larger solution space than that the duplication-free mapping algorithm can. Usually the more duplicable nodes, the larger each EFFC and the larger space the mapping algorithm can explore. However, the excessive node duplication may also be more serious and may finally lead to a poor solution. The cut ordering technique presented in Section 3.3.2 can help to reduce excessive node duplication, but the results are not good enough.

To better control the node duplication for better results, we perform a two-phase mapping. In the first phase, we mark every node as duplicable, thus, the EFFC of a node is its complete fanin cone up to the PIs. Similar to the duplication-free mapping algorithm discussed in the previous subsection, we use dynamic programming to processing node in a topological order from PIs to POs.

At the completion of the first phase, we tentatively form a mapping solution by constructing LUTs for nodes from POs to PIs. Initially, all the POs are marked as LUT roots. If a node is marked as an LUT root, we construct its LUT based on its best cut and mark all inputs to the cut as LUT roots. We repeat the process until all LUT roots are processed. After constructing the mapping solution, we mark all LUT roots as non-duplicable and the rest nodes in the original circuit as duplicable and start the second phase of re-mapping.

In the second phase, since many nodes are marked as non-duplicable, the size of EFFC of each node will be reduced. A non-duplicable node u will not be included in a node v's LUT in the second phase if u is not in the EFFC of v. By excluding u, v may try to include some other duplicable nodes in its LUT to either reduce the area directly or reduce the cut size of the LUT of v to create more freedom for a postprocessing of LUT packing to further reduce the area.

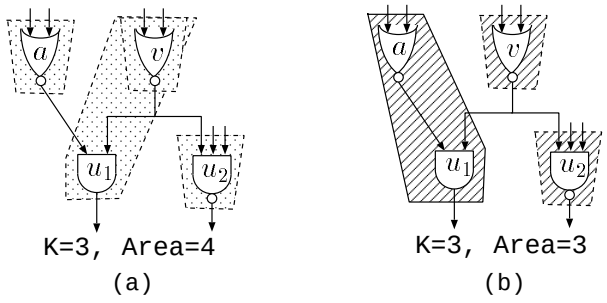

Figure 2: Excessive duplication removal.

Let us look at the examples shown in Figure 2 with $K = 3$ and the area of every LUT be 1. In Phase 1, we may

Heterogeneous FPGAs with 5,6-LUTs									
circuit	LB	PRAETOR	CPU(s)	CRF	CPU(s)	MIS	CPU(s)	TOS	CPU(s)
C1355	63	66	0.4	88	0.2	66	37	84	8
C1908	78	99	1.0	110	0.4	97	46	115	29
C2670	133	137	0.7	156	0.9	138	66	193	103
C3540	271	300	2.9	316	1.4	285	180	379	2946
C5315	310	341	2.9	371	1.2	339	125	442	281
C6288	398	461	5.6	496	2.0	481	110	593	5533
C880	89	95	0.3	96	0.2	87	65	100	79
alu2	84	103	0.6	104	0.6	94	75	44	489
alu4	150	180	1.4	186	0.9	168	274	121	1773
apex6	158	170	0.4	177	0.3	168	34	152	109
des	534	560	4.2	937	4.1	886	577	1282	5219
frg2	201	222	0.9	237	0.3	236	30	227	3016
i6	67	67	0.2	108	0.2	108	14	67	0
i7	103	103	0.2	143	0.3	103	19	103	5
i8	288	300	1.6	306	0.6	250	52	335	15992
i9	133	133	0.5	137	0.3	137	38	187	17326
i10	489	594	8.4	618	1.4	584	180	813	1179
k2	283	339	2.7	339	0.8	312	2391	512	85601
pair	303	345	1.6	361	0.6	344	42	445	416
rot	149	177	0.4	180	0.3	176	38	222	580
t481	174	195	2.8	198	17.5	194	413	393	4201
too_large	72	82	0.4	80	0.4	74	80	117	1161
vda	159	191	2.3	191	0.4	172	47	274	5693
average	204	229	1.8	258	1.5	239	214	313	6597
+%	-11%	1	1	+13%	0.8X	+5%	117X	+21%	3594X

Table 1: Comparison of PRAETOR with Chortle-crf (Column CRF), MIS-pga-new (Column MIS) and TOS-TUM (Column TOS) for Heterogeneous FPGAs with 5 and 6-LUTs.

generate a solution with four LUTs rooted at a, u_1, u_2 and v as shown in Figure 2(a). Obviously, to pack v into LUT_{u_1} has no benefit because v needs to be implemented as an LUT root according to LUT_{u_2}. In Phase 2, we will mark a, v, u_1, u_2 as non-duplicable. As a result, $v \notin EFFC_{u_1} = \{a, u_1\}$ and will not be packed into LUT_{u_1}. Instead, u_1 will select another cut by covering a to achieve a better solution with area of 3 as shown in Figure 2(b).

4.3 Heterogeneous FPGA Mapping for Area Minimization

In the previous two subsections we showed how to compute duplication-free mapping or general mapping solutions. With cut enumeration, both methods can be used for both homogeneous and heterogeneous FPGAs. The only difference in handling the two types of FPGAs is that when computing the cost of a K-feasible cut, the area of every cut is the same for homogeneous FPGAs, while the area of different cuts with different cut sizes may be different for heterogeneous FPGAs.

5 Experimental Results

We have implemented our cut enumeration, ranking and pruning techniques and applied to both duplication-free mapping and general mapping. Our algorithm is named PRAETOR. We have tested 23 MCNC combinational benchmarks (100∼4K gates) and 6 large industrial examples (26K∼100K gates) on a SUN Enterprise 4000 with 1.5 GB memory.

To evaluate the efficiency and mapping quality of our algorithm, we first compared PRAETOR with DFmap [2] for optimal duplication-free mapping on the 6 large industrial examples with $K = 6$. On average PRAETOR is over 165X faster with almost the same area. The detailed data are omitted due to space limit.

We also compared PRAETOR with Chortle-crf [6], MIS-pga-new [9] and TOS-TUM [4, 14] for general mapping with node duplication for FPGA with 6-LUTs. Our results show that MIS-pga-new and TOS-TUM can achieve considerably better results on some small examples, because both of them use the Boolean optimization techniques to explore a much larger solution space. However, for large designs, since they cannot afford to search the entire solution space with Boolean optimization, in practice they may not always be able to generate better results than that a good mapping algorithm can achieve. PRAETOR can outperform Chortle-crf for almost all the large industrial and MCNC examples (except for alu2 in which we used only 3 more 6-LUTs). On average, PRAETOR can achieve results with 18%, 11% or 31% fewer 6-LUTs comparing with Chortle-crf, MIS-pga-new and TOS-TUM, respectively. Moreover, PRAETOR can achieve results which are only 14% larger than the lower-bounds on the minimum area computed based our cut enumeration and evaluation techniques. As this set of results is similar to the results for a more general heterogeneous FPGA mapping to be shown in the next paragraph, we omit the detailed data to save space. To show the scalability of our algorithm, we also compared PRAETOR with Chortle-crf on the 6 large industrial examples. (Neither MIS-pga-new nor TOS-TUM is able to map those large examples in a reasonable amount of time.) Our test results show that PRAETOR can achieve results with 10% fewer 6-LUTs and in 53% shorter CPU time on average.

For heterogeneous FPGAs, we compared PRAETOR with Chortle-crf [6], MIS-pga-new [9] and TOS-TUM [4, 14] on the set of MCNC examples.[3] We tested for one type of FPGA which has both K-LUTs and $(K+1)$-LUTs where

[3] We did not compare with those in [7, 8], because they assumed a different kind of heterogeneous FPGAs, where the numbers of LUTs with different sizes have a pre-determined ratio.

each (K+1)-LUT is twice as large as a K-LUT. This architecture is similar to both ORCA2C FPGAs and Xilinx XC4K FPGAs. In our test, we set $K = 5$. The script for MIS-pga-new is recommended in [11] for the best quality. The script of TOS-TUM is based on the script mmap_h_a_5.scr for area minimization with high optimization effort recommended in [4]. For TOS-TUM we run SIS commands *collapse* to first collapse the circuits into 2 levels, because TOS-TUM favors two level circuits. If *collapse* cannot finish in 30 minutes, we use *reduce_depth* to partially collapse the circuits for TOS-TUM.

Table 1 lists our experimental results. In all the mapping solutions, we assign each LUT with 6 inputs as a 6-LUT, each LUT with 5 or less inputs as a 5-LUT. For Chortle-crf, MIS-pga-new and TOS-TUM, we tried mapping for both K=5 and 6, and select the solution with smaller area for each example. The CPU time in seconds includes the CPU time for both run, except for MIS-pga-new we only list the CPU time for K=5 because for all the examples its results with K=5 are always better than the results with K=6. PRAETOR, as a comparison, considers both 5- and 6-LUTs simultaneously. In Table 1, Columns PRAETOR, CRF, MIS and TOS list the area of the results by our algorithm, Chortle-crf, MIS-pga-new and TOS-TUM, respectively, for each example. The results show PRAETOR can reduce the area by 13% or 5% or 21% over Chortle-crf or MIS-pga-new or TOS-TUM, respectively. Both PRAETOR and Chortle-crf are very fast for this set of examples. They can finish in a few seconds for every example. MIS-pga-new needs 214 seconds for K=5 on average. TOS-TUM needs more than 1.8 hours for two runs with K=5 and 6 on average.

		with pruning		without pruning	
circuit	nodes	total	max	total	max
big1	26138	44811	67	800466	173
big2	52278	46712	31	2649413	303
big3	29920	28206	37	1678626	424
big4	30385	17476	55	1452035	523
big5	92021	91005	47	5224315	276
big6	101711	111541	53	2897107	310
average	56035	56625	48	2450327	335
ratio		1	1	43X	7X

Table 2: Effectiveness of cut pruning for $K = 6$.

Table 2 shows the effectiveness of our cut pruning techniques (in combining with ranking) for $K = 6$ for the six large industrial examples. Column "nodes" lists the number of (2-input) gates in the original circuits. Columns "total" list the total number of 6-cuts enumerated for all the gates with or without pruning. Columns "max" list the maximum number of 6-cuts enumerated for every gate with or without pruning. The results show that our pruning technique can reduce the total number of 6-cuts by a factor of 43, yet with good mapping quality. Note that the total number of K-cuts we enumerated may be less than the total number of 2-input gates in the original circuits, because our algorithm will shrink many K-feasible MFFCs into sigle nodes.

6 Discussions and Future Work

In this paper, we present a set of efficient and effective cut enumeration, ranking and pruning techniques for FPGA mapping. Those techniques can be used to design highly efficient and adaptive algorithms for both homogeneous and heterogeneous FPGAs and FPGAs with special architectures, like carry and cascade chain. Our test results show that our area-oriented algorithms based on those techniques are highly scalable to both circuit size and LUT size, and can outperform the state-of-the-art algorithms in both runtime and quality. In the future, we plan to extend our cut enumeration techniques for area/delay tradeoff.

7 Acknowledgements

This work is partially supported by National Science Foundation Young Investigator Award MIP9357582 and grants from Quickturn Design Systems and Lucent Technologies under the California MICRO program.

References

[1] J. Cong and Y. Ding. FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. *IEEE Trans. on Computer-Aided Design of Integrated Circuits And Systems*, 13(1):1–12, 1994.

[2] J. Cong and Y. Ding. On Area/Delay Trade-off in LUT-based FPGA Technology Mapping. *IEEE Trans. on VLSI Systems*, 2(2):137–148, June 1994.

[3] J. Cong and Y. Ding. Combinational Logic Synthesis for SRAM Based Field Programmable Gate Arrays. *ACM Transactions on Design Automation of Electronic Systems*, 1(2):145–204, 1996.

[4] K. Eckl, C. Legl, A. Lu, and B. Rohfleisch. *TOS-2.2 Technology Oriented Synthesis.* Institute of Electronic Design Automation, Technical University of Munish, 1996.

[5] A. Farrahi and M. Sarrafzadeh. Complexity of the Lookup-Table Minimization Problem for FPGA Technology Mapping. *IEEE Trans. on Computer-Aided Design of Integrated Circuits And Systems*, 13(11):1319–1332, 1994.

[6] R. J. Francis, J. Rose, and Z. Vranesic. Chortle-crf: Fast Technology Mapping For Lookup Table-Based FPGAs. In *28th ACM/IEEE Design Automation Conference*, pages 613–619, 1991.

[7] J. He and J. Rose. Technology Mapping for Heterogeneous FPGAs. In *FPGA'92*, 1994.

[8] M. Korupolu, K. Lee, and D. Wong. Exact Tree-based FPGA Technology Mapping for Logic Blocks with Independent LUTs. In *Prod. 35th ACM/IEEE Design Automation Conference*, pages 708–711, 1998.

[9] R. Murgai, N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli. Improved Logic Synthesis Algorithms For Table Lookup Architectures. In *IEEE International Conference on CAD*, pages 564–567, 1991.

[10] P. Pan and C. Liu. A New Retiming-based Technology Mapping Algorithm for LUT-based FPGAs. In *ACM Int'l Symp. on Field Programmable Gate Arrays*, pages 35–42, 1998.

[11] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli. *SIS: A System for Sequential Circuit Synthesis.* Electronics Research Laboratory, Memorandum No. UCB/ERL M92/41, 1992.

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

[13] Vantis and AMD Company. *Vantis VF1 Field Programmable Gate Array.* 1998.

[14] B. Wurth, K. Eckl, and K. Antreich. Functional Multiple-Output Decomposition: Theory and an Implicit Algorithm. In *Proc. ACM/IEEE Design Automation Conference.*, pages 54–59, 1995.

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