Structural Gate Decomposition for Depth-Optimal Technology Mapping in LUT-Based FPGA Designs

JASON CONG and YEAN-YOW HWANG
University of California

In this paper we study structural gate decomposition in general, simple gate networks for depth-optimal technology mapping using K-input Lookup-Tables (K-LUTs). We show that (1) structural gate decomposition in any K-bounded network results in an optimal mapping depth smaller than or equal to that of the original network, regardless of the decomposition method used; and (2) the problem of structural gate decomposition for depth-optimal technology mapping is NP-hard for K-unbounded networks when $K \geq 3$ and remains NP-hard for K-bounded networks when $K \geq 5$. Based on these results, we propose two new structural gate decomposition algorithms, named `DOGMA` and `DOGMA-m`, which combine the level-driven node-packing technique (used in `Chortle-d`) and the network flow-based labeling technique (used in `FlowMap`) for depth-optimal technology mapping. Experimental results show that (1) among five structural gate decomposition algorithms, `DOGMA-m` results in the best mapping solutions; and (2) compared with `speed_up` (an algebraic algorithm) and `TOS` (a Boolean approach), `DOGMA-m` completes decomposition of all tested benchmarks in a short time while `speed_up` and `TOS` fail in several cases. However, `speed_up` results in the smallest depth and area in the following technology mapping steps.

Categories and Subject Descriptors: B.6.1 [**Logic Design**]: Design Styles; B.6.3 [**Logic Design**]: Design Aids; *Automatic synthesis*; B.7.1 [**Integrated Circuits**]: Types and Design Styles

General Terms: Design, Experimentation, Measurement, Performance, Theory

Additional Key Words and Phrases: Computer-aided design of VLSI, decomposition, delay minimization, FPGA, logic optimization, programmable logic, simplification, synthesis, system design, technology mapping

The authors would like to acknowledge the support of the NSF Young Investigator (NYI) Award MIP-9357582, grants from Xilinx, Quickturn, and Lucent Technologies under the California MICRO programs, and the donation of software by Synopsys.
Authors' address: Department of Computer Science, University of California, Los Angeles, CA 90024.

1. INTRODUCTION

Field programmable gate arrays (FPGAs) have been widely used in circuit design implementation and system prototyping due to their short design cycles and low nonrecurring engineering costs. An important class of FPGAs use lookup-tables (LUTs) as the basic logic element. A K-input LUT (K-LUT), which consists of 2^K SRAM cells, can store the truth table of an arbitrary Boolean function of up to K variables. By connecting LUTs into a network, LUT-based FPGAs can be used to implement circuit designs in a short time.

Logic synthesis for LUT-based FPGAs transforms networks of logic gates into functionally equivalent LUT networks. The process is usually divided into two tasks: *logic optimization* and *technology mapping*. Logic optimization extracts common subfunctions to reduce the circuit size and/or resynthesizes critical paths to reduce the circuit delay. Technology mapping consists of two subtasks: *gate decomposition* and *LUT mapping*. In gate decomposition, large gates are decomposed into gates of at most K inputs (that is, *K-bounded*). The resulting K-bounded network is then mapped onto (i.e., covered by) K-LUTs in the LUT mapping step. The separation of optimization and mapping tasks is artificial. Some LUT synthesis algorithms (e.g., Lai et al. [1994] and Wurth et al. [1995]) decompose collapsed networks into LUT networks directly. The objectives of these tasks include area minimization, delay minimization, routability maximization, or a combination of all of them. A comprehensive survey of gate decomposition, LUT mapping, and logic synthesis algorithms for LUT-based FPGAs can be found in Cong and Ding [1996].

The delay of an LUT network can be measured by the number of levels (or *depth*) in the network under the unit delay model. A number of algorithms were proposed in the past for delay-oriented LUT mapping. We classify them into two classes. The first class of algorithms, such as `Chortle-d` [Francis et al. 1991b]; `DAG-Map` [Chen et al. 1992]; and `FlowMap` [Cong and Ding 1994a] perform LUT mapping without logic resynthesis. Among these algorithms, `Chortle-d` guarantees depth-optimal technology mapping for simple gate tree networks, and `FlowMap` guarantees depth-optimal LUT mapping for general K-bounded networks. Following `FlowMap`, `FlowMap-r` [Cong and Ding 1994b] and `CutMap` further reduce the mapping area, and `FlowMap-d` [Cong and Ding 1994c] and Edge-Map [Yang and Wong 1994] minimize delay under a more accurate net delay model. Another class of LUT mapping algorithms, such as MIS-pga-delay [Murgai et al. 1991]; TechMap-D [Sawkar and Thomas 1993]; FlowSyn [Cong and Ding 1993]; and ALTO [Huang et al. 1996] collapse critical paths followed by delay-oriented logic resynthesis. Due to resynthesis, this class of algorithms could obtain mapping depth smaller than the optimal depth computed by `FlowMap`, but usually with longer computation time.

Gate decomposition may significantly affect the network depth obtained by the algorithms in the first LUT mapping class. For example, the

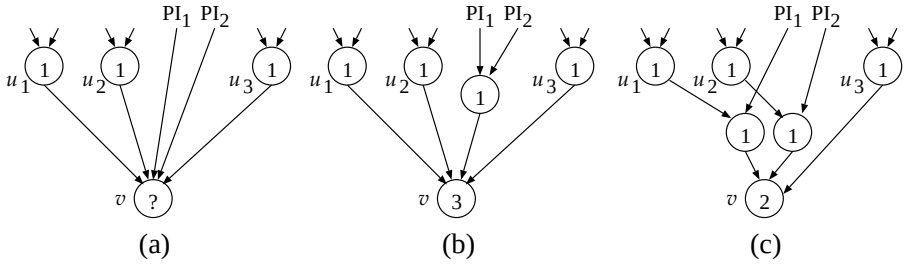

Fig. 1. Impact of gate decomposition on mapping depth for $K = 3$. (a) Initial network; (b) a decomposition resulting in a mapping depth of 3; (c) a decomposition resulting in a mapping depth of 2.

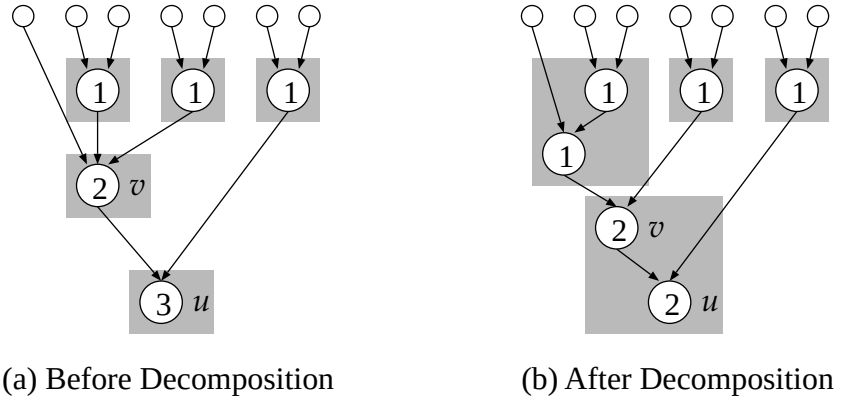

Fig. 2. Gate decomposition in a K-bounded network ($K = 3$). (a) Initial K-bounded network with a mapping depth of 3; (b) decomposed network with a mapping depth of 2.

network in Figure 1(a) is not a K-bounded network for $K = 3$. When node v is decomposed as shown in Figure 1(b), any mapping algorithm will result in a depth of 3 or larger. But if node v is decomposed in the way shown in Figure 1(c), a mapping solution with a depth of 2 can be obtained. In addition, when a K-bounded network is further decomposed, the mapping depth could be reduced. Figure 2(a) shows a 3-bounded network. For $K = 3$, `FlowMap` produces a 3-level mapping solution of 5 LUTs. (Every shaded square represents an LUT in the figure.) But if node v is further decomposed, `FlowMap` produces a 2-level network of 4 LUTs (Figure 2(b)). The two examples demonstrate that gate decomposition affects the depth obtained by LUT mapping algorithms.

We classify gate decomposition methods into *structural*, *algebraic*, or *Boolean approaches*. Structural gate decomposition can only be applied to *simple gates* (e.g., AND gates, OR gates, XOR gates). Complex gates need to be transformed into simple gates (e.g., via AND-OR decomposition) before any structural decomposition. The `tech_decomp` algorithm in `SIS` [Sentovich et al. 1992]; the `dmig` algorithm [Wang 1989; Chen et al. 1992]; and the `Chortle` family of mapping algorithms [Francis et al. 1991a; 1991b] all

perform structural gate decomposition. In *algebraic gate decomposition* approaches, networks are usually partially collapsed and gates are represented in the sum-of-product (SOP) form. Common logic subfunctions are then extracted with algebraic divisions [Rudell 1989; De Micheli 1994]. The `speed_up` algorithm in `SIS` [Sentovich et al. 1991] is an algebraic approach which collapses critical paths followed by network resynthesis for delay minimization. In *Boolean gate decomposition* approaches, logic gates are decomposed via functional operations. Shannon expansion, if-then-else (ITE) decomposition, and AND-OR decomposition are very common Boolean gate decomposition operations. Recently, *functional decomposition* techniques [Ashenhurst 1959; Curtis 1961; Roth and Karp 1962] were used in a number of LUT network synthesis algorithms [Lai et al. 1994; Wurth et al. 1995; Legl et al. 1996b]. In these algorithms, networks are completely collapsed whenever possible so that the outputs can be represented as functions of the network inputs directly. The output functions are then decomposed into composed K-input subfunctions for implementation using K-LUTs. Optional LUT mapping steps may follow to improve the synthesis results. The FGSyn algorithm [Lai et al. 1994] and the BoolMap-D algorithm [Legl et al. 1996b] take this approach for delay-oriented LUT network synthesis. Generally speaking, algebraic approaches and Boolean approaches are more effective for both area and delay minimization in technology mapping, while structural approaches are usually faster. Hybrid approaches such as algebraic decompositions followed by structural decompositions are used in many logic synthesis approaches.

In this paper we study structural gate decomposition for delay minimization in general networks with the following motivations. First, we have shown how gates are decomposed, which can affect the mapping depth computed by `FlowMap`. A good gate decomposition step allows mapping algorithms to obtain the smallest mapping depth. Second, structural gate decomposition allows arbitrary grouping of gate inputs for our optimization objective, while algebraic or Boolean approaches do not have this advantage. Third, structural gate decomposition is computationally efficient. This is an important factor for mapping large designs and estimating the mapping delay or area. Nowadays, the IC process technology has advanced to 0.18 μm and below. Million-gate FPGAs have become a reality. Structural gate decomposition algorithms can be employed in the technology mapping approaches along with this technology trend.

Several delay-oriented structural gate decomposition algorithms were proposed in the past. The `tech_decomp` algorithm [Sentovich et al. 1992] decomposes each simple gate into a balanced fanin tree to minimize the number of levels locally. The `dmig` algorithm [Wang 1989; Chen et al. 1992] is based on the Huffman coding algorithm and guarantees the minimum depth in the decomposed network. However, the mapping depth might not be the minimum. The network in Figure 1(b) is actually decomposed using `dmig` and results in a suboptimal mapping depth. The `Chortle-d` algorithm [Francis et al. 1991b] employs bin-packing heuristics to achieve

depth minimization, but is optimal for trees only. In this paper we go one step further. We shall develop structural gate decomposition algorithms for depth-optimal technology mapping on general networks.

The rest of this paper is organized as follows. Section 2 defines the terminology, presents general properties, and formulates the structural gate decomposition problems. Section 3 addresses the NP-completeness of the problems. Section 4 presents two new algorithms, DOGMA and DOGMA-m, for structural gate decomposition. Experimental results are presented in Section 5, and Section 6 concludes the paper. A preliminary version of this work was published in DAC'96 [Cong and Hwang 1995] without the proofs of theorems and considered single-gate decompositions only.

2. PROBLEM FORMULATION

2.1 Definitions and Preliminaries

A combinational Boolean network N can be represented by a directed acyclic graph $N = (V, E)$ where each node $v \in V$ represents a logic gate and each directed edge $(u, v) \in E$ represents a connection from the output of node u to the input of node v. A node v is a *simple* gate if v implements one of the following functions: AND, OR, XOR, or their inversions. Primary inputs (PIs) are nodes of in-degree zero. Other nodes are *internal*, and some are designated as primary outputs (POs). A node v is a *predecessor* of a node u if there is a directed path from v to u in N. The *depth* of a node v is the number of edges on the longest path from any PI to v. Each PI has a depth of zero. The depth of a network is the largest depth for nodes in the network. Let $input(v)$ and $fanout(v)$ represent the set of fanins and the set of fanouts of node v, respectively. Given a subgraph H of N, let $input(H)$ denote the set of distinct nodes outside H that supply inputs to nodes in H. A *fanin cone* C_v rooted at v is a connected subnetwork consisting of v and its predecessors. Node v is the *root node* of C_v, and is denoted as $root(C_v) = v$. Let K be the LUT input size. A node v is *K-bounded* if $|input(v)| \leq K$. Otherwise, v is *K-unbounded*. A network N is K-bounded if it contains only K-bounded nodes.

Given a K-bounded network N, a set $M = \{L_1, L_2, \ldots, L_m\}$ of subnetworks is a *K-LUT mapping solution* of N if

(C1) for every $L_i \in M$, L_i is a fanin cone in N and $|input(L_i)| \leq K$;

(C2) for every $L_i \in M$, $input(L_i)$ contains only PIs or root nodes of other subnetworks in M;

(C3) for every $L_i \in M$, $root(L_i)$ is either a PO or belongs to $input(L_j)$ for some $L_j \in M$; and

(C4) for every PO v of N, $v = root(L_i)$ for some $L_i \in M$.

A mapping solution M is *duplication-free* if $L_i \cap L_j = \emptyset$ for all $L_i \neq L_j$ in M. By implementing every subnetwork in M using a K-LUT, we obtain a K-LUT network that is functionally equivalent to N. The *mapping area* and the *mapping depth* of M is the LUT count (i.e., $|M|$) and the depth in the K-LUT network that implements M, respectively.

Given a K-bounded network N, let $S_K(N)$ represent the set of K-LUT networks that implement all mapping solutions of N. The *minimum mapping depth* of N, denoted $MMD(N)$, is the minimum network depth for all K-LUT networks in $S_K(N)$. Let N_v represent the largest fanin cone rooted at v in N. The minimum mapping depth of a node $v \in N$, denoted $MMD_N(v)$, is $MMD(N_v)$. The mapping depth of any PI is 0. Given a K-bounded network N, the `FlowMap` algorithm [Cong and Ding 1994a] computes $MMD_N(v)$ for every node $v \in N$ in polynomial time. A *cut* in N_v is a partition $(X_v, \bar{X}_v)$ of N_v such that $\bar{X}_v$ is a fanin cone rooted at v and X_v is $N_v - \bar{X}_v$. The *cutset* of the cut, denoted $n(X_v, \bar{X}_v)$, is defined as $input(\bar{X}_v)$. The cut is *K-feasible* if $|n(X_v, \bar{X}_v)| \leq K$. The height of the cut, denoted $height(X_v, \bar{X}_v)$, is $\max\{MMD_N(u) \mid u \in n(X_v, \bar{X}_v)\}$. `FlowMap` computes a min-height K-feasible cut in the fanin cone of each node v to obtain $MMD_N(v)$.

The following two lemmas are on the minimum mapping depth in general networks. Lemma 1 states the *monotone property* of minimum mapping depth and Lemma 2 gives a way to compute $MMD_N(v)$.

LEMMA 1. *[Cong and Ding 1994a]. Let $N = (V, E)$ be a K-bounded network and let node $v \in V$. Then $MMD_N(u) \leq MMD_N(v)$ for every fanin $u \in input(v)$.*

LEMMA 2. *[Cong and Ding 1994a]. Let $N = (V, E)$ be a K-bounded network, node $v \in V$, and let $\max\{MMD_N(u) \mid u \in input(v)\} = p$. Then $MMD_N(v) = p$ if there exists a K-feasible cut of height $p - 1$ in N_v. Otherwise, $MMD_N(v) = p + 1$.*

2.2 Properties of Structural Gate Decomposition

Simple gates allow arbitrary grouping of their fanins in decomposition. However, the grouping and the resulting gate size in decomposition can significantly affect the depth and area in the final mapping solution. In this section, we show that the best mapping results can only be obtained from completely decomposed networks.

Let node v be a simple gate in a network N and let $|input(v)| \geq 3$. Given a structural gate decomposition algorithm D, a *decomposition step* D_v on node v (i) chooses two fanins u_1 and u_2 of v; (ii) removes edges (u_1, v) and (u_2, v); and (iii) introduces a node w and three edges (u_1, w), (u_2, w), and (w, v) to reconnect u_1, u_2 and v. Because v is a simple gate, D_v can always

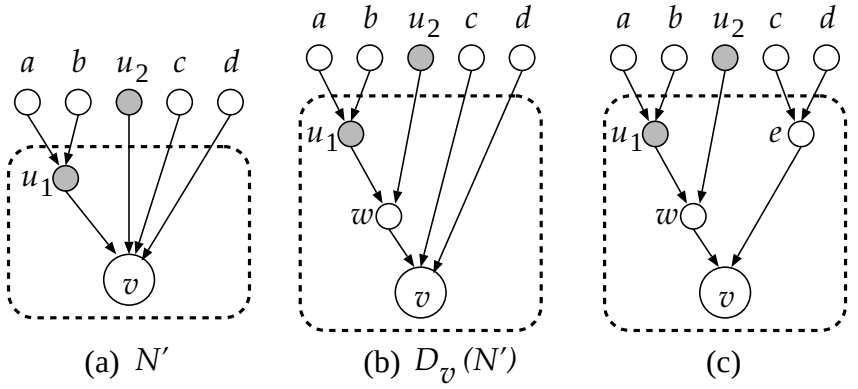

Fig. 3. Decomposition of node v. (a) Before decomposition; (b) $D_v(N)$ after one decomposition step D_v; (c) complete decomposition of v.

be applied. Node w has the same gate type as node v. For any subnetwork $N' = (V', E')$ of N and a decomposition step D_v, we define

$$D_v(N') = (V' \cup \{w\}, E' - \{(u_1, v), (u_2, v)\} \cup \{(u_1, w), (u_2, w), (w, v)\})$$

if $v \in V'$, and $D_v(N') = N'$ if $v \notin N'$. A network is completely decomposed when it becomes 2-bounded. In Figure 3(a), N' contains nodes u_1 and v with $input(N') = \{a, b, u_2, c, d\}$. Figure 3(b) shows $D_v(N')$ after one decomposition step D_v. The subnetwork is completely decomposed in Figure 3(c). We have the following theorem.

THEOREM 1. *Let $N = (V, E)$ be a K-bounded network, node $v \in V$ be a simple gate, and $|input(v)| \geq 3$. Then $S_K(N) \subset S_K(D_v(N))$ for any structural gate decomposition algorithm D.*

PROOF. Let w be the node introduced by D_v. Let $M = \{L_1, L_2, \ldots, L_m\}$ be an arbitrary mapping solution of N. We claim $M' = \{D_v(L_1)D_v(L_2), \ldots, D_v(L_m)\}$ is a mapping solution of $D_v(N)$. First, N and $D_v(N)$ have the same set of PIs and POs. From Figure 3, it should be clear that L_i and $D_v(L_i)$ have the same set of inputs as well as the same output node. As a result, M' satisfies conditions (C1) to (C4) as a mapping solution of $D_v(N)$. The K-LUT that implements L_i also implements $D_v(L_i)$. Hence the K-LUT network that implements M also implements M'. Therefore, $S_K(N) \subseteq S_K(D_v(N))$. However, a mapping solution M'' of $D_v(N)$ cannot be a mapping solution for N if w is the root node of some subnetwork in M'' (due to $w \notin N$). There exists at least one such mapping solution that is $D_v(N)$ itself. As a result, $S_K(N) \subset S_K(D_v(N))$. □

Corollary 1.1 Let $N = (V, E)$ be a K-bounded network, node $v \in V$ be a simple gate, and $|input(v)| \geq 3$. Then $MMD(D_v(N)) \leq MMD(N)$ for any structural gate decomposition algorithm D.

PROOF. Since $S_K(N) \subset S_K(D_v(N))$ for any decomposition algorithm D, by definition, $MMD(D_v(N)) \leq MMD(N)$. □

Note that Theorem 1 and Corollary 1.1 hold as long as the decomposition step at v (structural, algebraic, or Boolean) can be carried out, regardless whether v is a simple gate or not. However, the algebraic or functional decomposition for a complex gate may not always be possible.

Since the set of all possible functionally equivalent K-LUT networks expands whenever a simple gate is decomposed (Theorem 1), it is always beneficial to decompose simple-gate networks into 2-bounded networks for LUT mapping algorithms to exploit the larger mapping solution space. The experimental results reported in Cong and Ding [1994a] confirm this conclusion. In their experiments, the input networks were first transformed into simple gate networks and then decomposed *structurally* into 5-bounded, 4-bounded, 3-bounded, or 2-bounded networks before LUT mapping. The resulting mapping depth decreases monotonically along with the decrease of gate sizes in decomposition. An interesting contrast comes from the results reported in Legl et al. [1996a], where networks were first collapsed completely and then decomposed *functionally* into 5-bounded, 4-bounded, or 3-bounded networks for LUT mapping. The best mapping solutions in terms of area and depth are mostly from the 5-bounded networks. The two experiments show an important difference between structural and functional decompositions: logic signals are preserved in structural decompositions, while new gates are synthesized during functional decompositions. In Legl et al. [1996a], the 5-bounded, 4-bounded, and 3-bounded networks contain totally different sets of internal gates, which are synthesized independently in three functional decomposition processes. In fact, according to Corollary 1.1, if the 5-bounded networks in Legl et al. [1996b] were further decomposed before LUT mapping, even smaller mapping depth could be obtained in their experiments.

The following lemma specifies a condition where the structural gate decomposition will not cause further mapping depth reduction.

LEMMA 3. *Let $N = (V, E)$ be a K-bounded network, node $v \in V$ be a simple gate, and $|input(v)| \geq 3$. Assume that nodes u_1, $u_2 \in input(v)$ and $MMD_N(u_1) = MMD_N(v)$ (see Figure 4(a)). Let D_v be the decomposition step that merges u_1, u_2 into an intermediate node w (see Figure 4(b)). Then $MMD(N) = MMD(D_v(N))$.*

PROOF. Assume $MMD_N(u_1) = MMD_N(v) = p$. First, $MMD_{D_v(N)}(u_1) = p$ (as $N_{u1} = D_v(N_{u1})$). Next, Lemma 1 (monotone property) assures that $p \leq MMD_{D_v(N)}(w) \leq MMD_{D_v(N)}(v)$. Then, according to Corollary 1.1, we have $MMD_{D_v(N)}(v) \leq MMD_{(N)}(v) = p$. Therefore, $MMD_{D_v(N)}(w) = MMD_{D_v(N)}(v) = p$ (see Figure 4(b)). Now we show $MMD(D_v(N)) = MMD(N)$. Suppose this is not the case. Then $MMD(D_v(N)) < MMD(N)$, and there exists a mapping solution $M = \{L_1, L_2, \ldots, L_m\}$ for $D_v(N)$ such that M has a

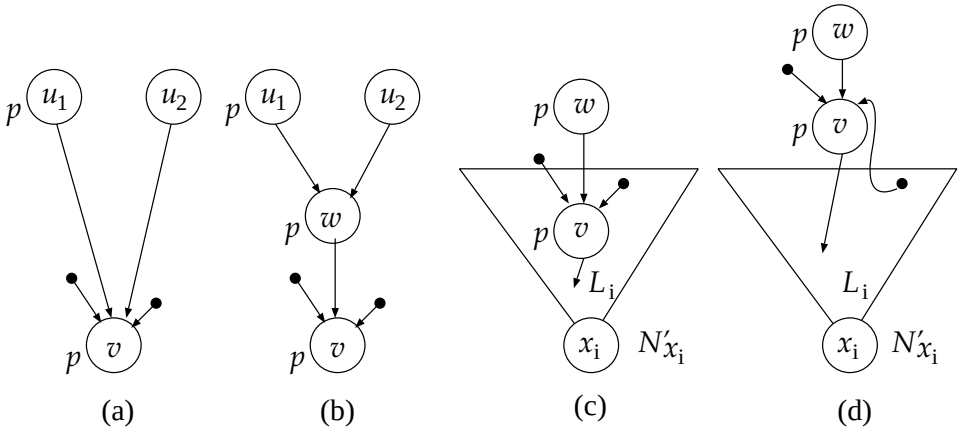

Fig. 4. (a) Before D_v; (b) after D_v; (c) $w \in input(L_i)$; (d) v is moved out of L_i.

depth smaller than $MMD(N)$. Let x_i represent the output node of each K-feasible subnetwork L_i in M. First, $w = x_i$ for some i. Otherwise, M would be a mapping solution of N (by collapsing w into v) and $MMD(D_v(N))$ would not be smaller than $MMD(N)$. Next, there must exist some x_i such that $MMD_{D_v(N)}(x_i) < MMD_N(x_i)$. We call node x_i a *depth-reduced* node. There are two cases for any depth-reduced node x_i. (i) $w \notin input(L_i)$. Then we can find another node $x_j \in input(L_i)$ such that $MMD_{D_v(N)}(x_j) < MMD_N(x_j)$. Otherwise node x_i won't be a depth-reduced node. We continue to trace depth-reduced nodes towards PIs. This tracing, however, won't reach PIs since PIs have a depth of 0. At certain depth, the second case must occur. (ii) $w \in input(L_i)$. Then $(N'_{x_i} - L_i, L_i)$ is a cut in the fanin cone N'_{x_i} in $D_v(N)$ (see Figure 4(c)). But we can move node v from L_i to $N'_{x_i} - L_i$ and obtain another K-feasible cut of height p in N'_{x_i} (see Figure 4(d)), since w is fanout-free and w and v have the same mapping depth p. This implies $MMD_{D_v(N)}(x_i) < MMD_N(x_i)$. As a result, x_i is not a depth-reduced node. Contradiction. So we proved $MMD(D_v(N)) = MMD(N)$. □

LEMMA 4. *Let $N = (V, E)$ be a K-bounded network, node $v \in V$ be a simple gate, and $|input(v)| \geq 3$. If $MMD_N(u_i) = MMD_N(v)$ for every fanin $u_i \in input(v)$, then $MMD(N) = MMD(D_v(N))$ for any structural gate decomposition algorithm D.*

PROOF. Since the intermediate node w has the same depth as node v, this lemma is true according to Lemma 3. □

2.3 Integrated versus Two-Step Technology Mapping

Gate decomposition and LUT mapping can be performed in two different ways. In an *integrated* mapping approach, the input network is decomposed and covered by LUTs simultaneously, while in a *two-step* mapping approach, the input network is decomposed into a K-bounded network *before*

LUT mapping is performed. For example, Chortle-d is an integrated mapping approach, while FlowMap fits only into a two-step mapping approach. The separation of gate decomposition and LUT mapping is a restriction in general, since integrated approaches allow more informative gate decomposition and LUT mapping decisions, while two-step approaches do not have this advantage. It may appear that the minimum mapping depth for all integrated mapping approaches will be smaller than the minimum mapping depth for all two-step mapping approaches. However, we show that this is not the case for structural gate decomposition.

THEOREM 2. *Given a K-bounded network N, if only structural gate decomposition is allowed, the minimum mapping depth for all integrated mapping approaches equals the minimum mapping depth for all two-step mapping approaches.*

PROOF. Given an arbitrary K-bounded network N, assume some integrated approach results in the optimal depth $MMD(N)$ in a mapping solution M_N. Then M_N is a mapping solution of some K-bounded network N' decomposed *structurally* from N. A depth-optimal mapper (e.g., FlowMap) can take N' as input and generate a mapping solution $M_{N'}$. Since $M_{N'}$ is depth-optimal with respect to N', we have $MMD(N') \leq MMD(N)$. But M_N is depth optimal with respect to N. As a result, $MMD(N) \leq MMD(N')$. Therefore, $MMD(N) = MMD(N')$. □

Our mapping algorithms, presented in Section 4, should be considered a hybrid approach. On one hand, depth minimization is achieved in structural gate decomposition (by DOGMA or DOGMA-m) to return a network topology of the minimum mapping depth; on the other hand, the LUT mapping solution is computed in depth-optimal LUT mapping with area minimization as a second objective. As a result, the depth and the area are optimized separately in the two steps of technology mapping. Hence we consider our algorithm a hybrid approach.

2.4 The SGD/K and K-SGD/K Problems

In this paper we study structural gate decomposition of K-bounded or K-unbounded simple gate networks into 2-bounded networks such that LUT mapping algorithms (e.g., FlowMap) can obtain the smallest mapping depth. We formulate the following two problems.

Structural gate decomposition for K-LUT mapping (SGD/K). Given a simple-gate K-unbounded network N_∞, decompose N_∞ into a 2-bounded network N_2 such that $MMD(N_2) \leq MMD(N'_2)$ for any other 2-bounded decomposed network N'_2 of N_∞.

Structural gate decomposition in K-bounded network for K-LUT mapping (K-SGD/K). Given a simple gate K-bounded network N_K, decompose N_K into a 2-bounded network N_2 such that $MMD(N_2) \leq MMD(N'_2)$ for any other 2-bounded decomposed network N'_2 of N_K.

3. COMPLEXITY OF SGD/K AND K-SGD/K PROBLEMS

We show the following results: (1) the SGD/K problem is NP-hard for $K \geq 3$; and (2) the K-SGD/K problem is NP-hard for $K \geq 5$. We present the construction for the NP-complete reduction, the lemmas and theorems, and the proofs for theorems. Proofs for lemmas can be found in the Appendix.

Our results are based on polynomial-time transformations from the 3SAT problem to the decision version of the SGD/K and the K-SGD/K problems. The 3SAT problem, which is a well-known NP-complete problem [Garey and Johnson 1979], is defined as follows:

Problem: 3-Satisfiability (3SAT).

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

Question: Is there a truth assignment for the variables in X such that $C_j = 1$ for $1 \leq j \leq m$?

We transform an arbitrary instance of 3SAT to an instance of SGD/K in polynomial time. The idea is to relate the truth assignment of variables in 3SAT to the decision of gate decomposition in SGD/K. Since determining the truth assignment is difficult, the decision of gate decomposition is also difficult. We define the *decision version* of the SGD/K problem as follows:

Problem: Structural gate decomposition for K-LUT mapping (SGD/K-D).

Instance: A constant $K \geq 3$, a depth bound B, and a simple gate K-unbounded network N_∞.

Question: Is there a way to structurally decompose N_∞ into a 2-bounded network N_2 such that the depth-optimal K-LUT mapping solution of N_2 has a depth no more than B?

Given an instance F of 3SAT with n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_i, C_2, \ldots, C_m$, we construct a K-unbounded network $N(F)$ corresponding to the instance F, as follows. First, for each variable x_i, we construct a subnetwork $N(x_i)$, which consists of the following nodes: (a) two output nodes denoted x_i and $\bar{x}_i$; (b) $(2K^2 - 3K)$ PI nodes in which two of them are denoted PI_i^1 and PI_i^2; (c) $(2K - 2)$ internal nodes, denoted $v_i^1, \ldots, v_i^{K-3}$, $u_i^1, \ldots, u_i^{k-2}$, w_i^1, w_i^2 and s_i, respectively; The nodes are connected as shown in Figure 5. Each node of w_i^1 and w_i^2 has $K - 1$ PI fanins. Node s_i has 4 fanins from w_i^1, w_i^2, PI_i^1 and PI_i^2. Every other internal node has K PI fanins. Note that $N(x_i)$ is well defined for $K \geq 3$ and is K-bounded for $K \geq 4$.

Next, for each clause C_j with 3 literals l_j^1, l_j^2, l_j^3, we construct a subnetwork $N(C_j)$, which consists of the following nodes: (a) one output node denoted C_j; (b) three literal nodes denoted l_j^1, l_j^2, l_j^3; (c) $(2K - 5)$ internal

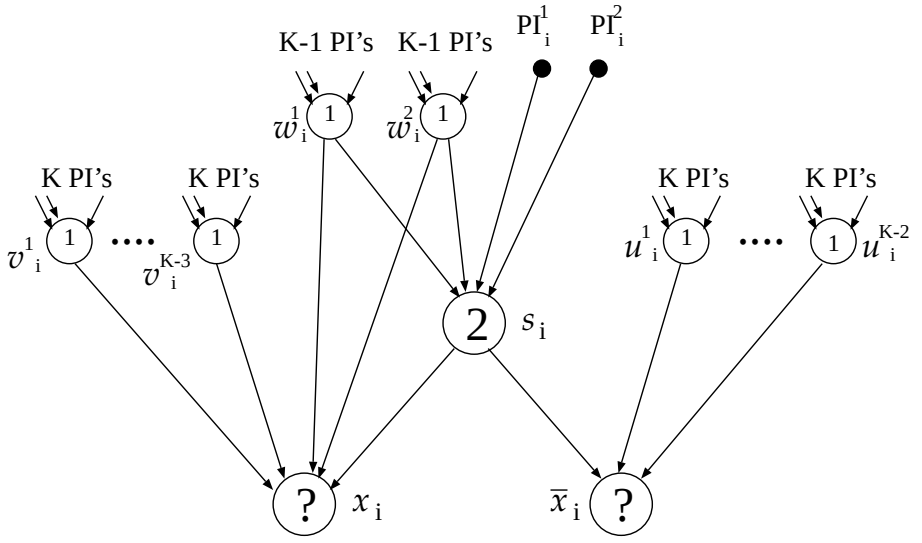

Fig. 5. Construction of network $N(x_i)$ for each Boolean variable x_i.

nodes $q_j^1, \ldots, q_j^{2K-5}$, each is the root of a complete 2-level K-ary tree with PI nodes as leaves; (d) $(K - 2)$ internal nodes $r_j^1, \ldots, r_j^{k-2}$, each is the root of a complete 3-level K-ary tree with PI nodes as leaves. The connections are shown in Figure 6(a). The output node C_j has all internal nodes as its fanins in $N(C_j)$. Note that $N(C_j)$ is well defined for $K \geq 3$. However, the output node C_j is not K-bounded.

Finally, we connect the subnetworks $N(C_j)$, $j = 1, 2, \ldots, m$ with the subnetworks $N(x_i)$, $i = 1, 2, \ldots, n$, as follows, to form the network $N(F)$. Let literal l_j^k be a literal in clause C_j. If $l_j^k = x_i$ where x_i is a variable, we connect node x_i in $N(x_i)$ as the single fanin of node l_j^k in $N(C_j)$. Similarly, if $l_j^k = \bar{x}_i$, we connect node $\bar{x}_i$ in $N(x_i)$ as the single fanin of node l_j^k in $N(C_j)$. Note that every literal node has exactly one fanin. This fanin node is called the *variable node* of the corresponding literal node. Network $N(F)$ has m primary outputs: nodes $C_1, \ldots, C_m$. We illustrate the construction of $N(F)$ by an example. Assume $F = (x_1 + \bar{x}_2 + x_3)(x_2 + \bar{x}_3 + x_4)(\bar{x}_1 + \bar{x}_3 + x_4)$. The network $N(F)$ is shown in Figure 7. Because clause $C_1 = (x_1 + \bar{x}_2 + x_3)$, we connect nodes $x_1, \bar{x}_2, x_3$ as fanins to nodes l_1^1, l_1^2, l_1^3 in $N(C_1)$, respectively. Node x_1 is the variable node of node l_1^1. We have the following lemma.

LEMMA 5. *The 3SAT instance F is satisfiable if and only if $N(F)$ can be decomposed into $D(N(F))$ such that $MMD(D(N(F))) = 4$.*

THEOREM 3. *The SGD/K problem is NP-hard for $K \geq 3$.*

PROOF. The transformation from an instance F of 3SAT to the network $N(F)$ takes $O(K^3(n + m))$ time. If the SGD/K-D problem could be solved

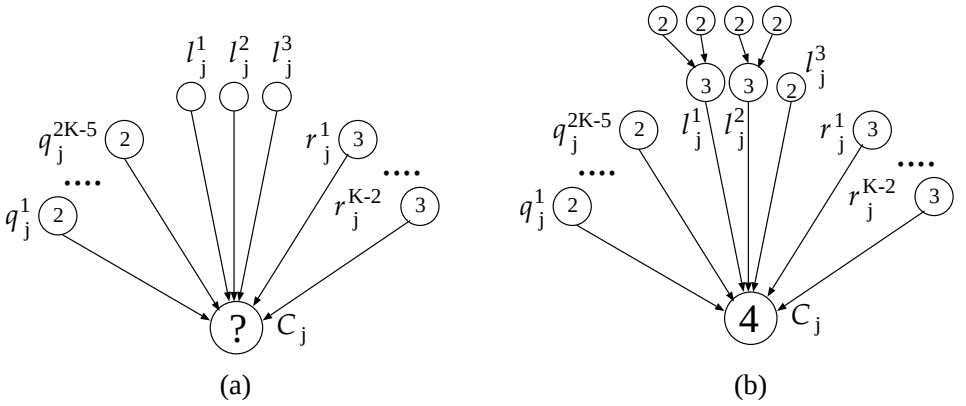

Fig. 6. (a) Construction of network $N(C_j)$ for each clause C_j; (b) exact $2K$ nodes of depth 2 appear when $MMD(l_j^3) = 2$.

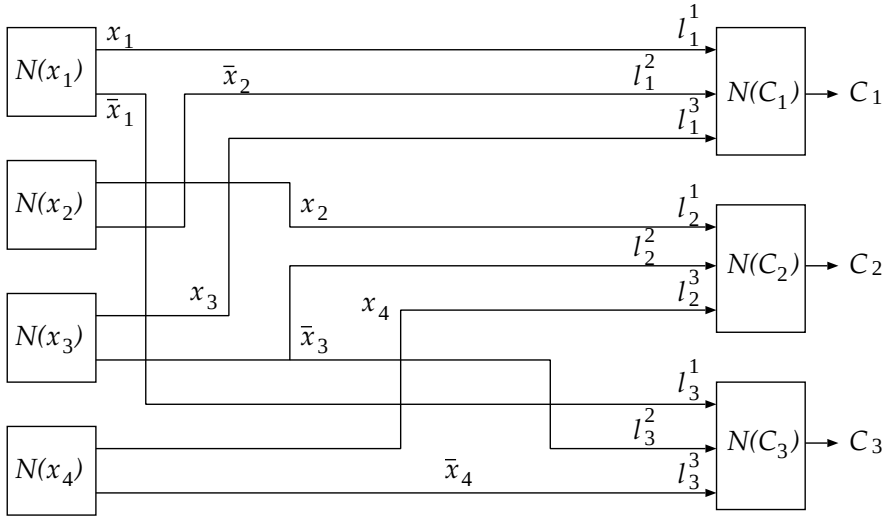

Fig. 7. The network $N(F)$ for $F = (x_1 + \bar{x}_2 + x_3)(x_2 + \bar{x}_3 + x_4)(\bar{x}_1 + \bar{x}_3 + x_4)$.

in polynomial time, we can set $B = 4$ and solve 3SAT in polynomial time. Since 3SAT is NP-hard, the SGD/K-D problem is NP-hard. For a given decomposed network $D(N(F))$ of $N(F)$, it takes polynomial time to compute its mapping depth d and verify whether $d \leq B$ (e.g., by `FlowMap`). As a result, the SGD/K-D problem is NP-complete. Since $N(x_i)$ and $N(C_j)$ are well defined for $K \geq 3$, the SGD/K-D problem is NP-complete for $K \geq 3$. Hence the SGD/K problem is NP-hard for $K \geq 3$. □

We now show the complexity of the K-SGD/K problem. In this construction of reduction, we must have every node K-bounded (note that $N(C_j)$ is not K-bounded in the previous construction). Given an instance F of the 3SAT with n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_i, C_2, \ldots, C_m$, we

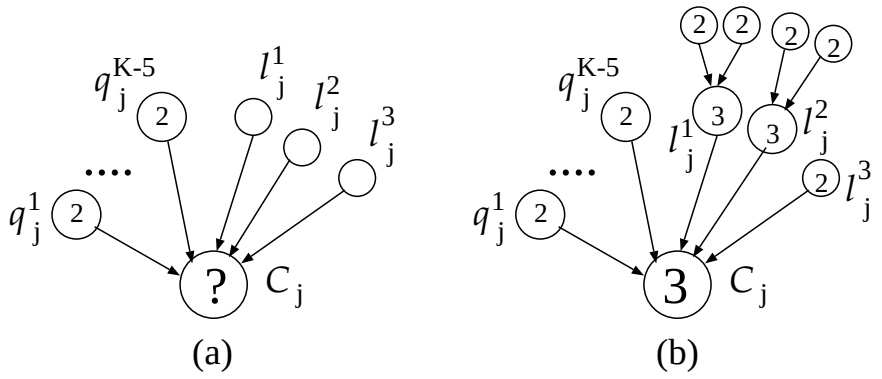

Fig. 8. Construction of K-bounded subnetwork $N_K(C_j)$ for each clause C_j.

construct a corresponding K-bounded network $N_K(F)$, as follows. For each variable x_i, construct subnetwork $N(x_i)$ as before (shown in Figure 5). However, for each clause C_j, construct subnetwork $N_K(C_j)$ consisting of (a) one output node denoted C_j; (b) three literal nodes denoted l_j^1, l_j^2, l_j^3, (c) $(K - 5)$ internal nodes $q_j^1, \ldots, q_j^{K-5}$, each of them is the root of a complete 2-level K-ary tree with PI nodes as leaves. The subnetwork $N_K(C_j)$ is shown in Figure 8(a). Note that $N_K(C_j)$ is well defined and K-bounded for $K \geq 5$. We connect subnetworks $N(x_i)$ and $N_K(C_j)$ according to the formula F as before, to obtain the network $N_K(F)$. We have the following lemma.

LEMMA 6. *The 3SAT instance F is satisfiable if and only if $N_K(F)$ can be decomposed into $D(N_K(F))$ such that $MMD(D(N_K(F))) = 3$.*

THEOREM 4. *The K-SGD/K problem is NP-hard for $K \geq 5$.*

PROOF. The subnetwork $N(x_i)$ is K-bounded for $K \geq 4$. The subnetwork $N_K(C_j)$ is K-bounded for $K \geq 5$. Based on similar arguments in the proof of Theorem 3, it is easy to see the K-SGD/K problem is NP-hard for $K \geq 5$. □

4. GATE DECOMPOSITION ALGORITHMS FOR DEPTH-OPTIMAL MAPPING

In this section we combine the node-packing technique in `Chortle-d` with the min-height K-feasible cut technique in `FlowMap` in structural gate decomposition of simple-gate networks. Our objective is to minimize the depth in the final mapping solution. We propose two algorithms. The first algorithm decomposes logic gates independently, as in most previous approaches, while the second algorithm decomposes multiple gates simultaneously to exploit common fanins. The advantage of multigate decomposition can be seen in one example. Nodes $a, b, \ldots, f$ in Figure 9 are primary inputs. If nodes u and v in Figure 9(a) are decomposed independently, we might obtain a network in Figure 9(b). For $K = 3$, the best

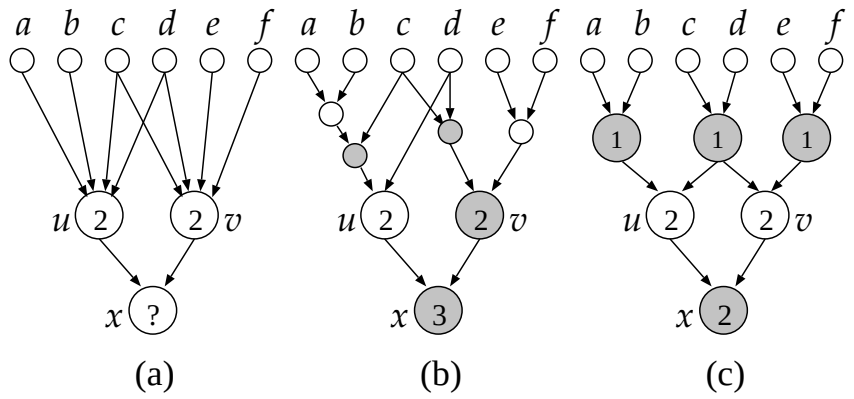

Fig. 9. Multigate decomposition. (a) Initial network; (b) single gate decomposition result; (c) multigate decomposition result. (Shaded nodes are LUT outputs).

mapping solution in this case is a 3-level network of 4 LUTs. However, if nodes u and v are decomposed together to exploit their common fanins c and d as shown in Figure 9(c), a 2-level network of 4 LUTs can be obtained. The depth is reduced in the mapping solution.

4.1 Single Gate Decomposition

We present our single gate decomposition algorithm DOGMA (Depth-Optimal Gate decomposition for MApping) in this section. Given a simple gate network N, DOGMA decomposes nodes in topological order from PIs to POs. At each node v, DOGMA decomposes and labels v with the number $l(v) = MMD_{N(v)}(v)$ where $N(v)$ denotes the decomposed network. The set of fanins of label q in $input(v)$, denoted S_q, is called a *stratum* of depth q. A K-feasible cut of height $q - 1$ exist for every node in S_q. A K-feasible cut of height $q - 1$ exists for a set B of nodes if such a cut exists for a node s created with $input(s) = B$. DOGMA groups $input(v)$ into strata according to their labels, and processes each stratum in two steps.

(1) Starting from stratum S_q of the smallest depth, DOGMA partitions S_q into a minimum number of subsets such that there exists a K-feasible cut of height $q - 1$ for each subset of nodes. The process is similar to packing objects into bins. Each bin has a size of K. The size of a node (also called an *object*) is the size of its min-cut of height $q - 1$. A set of nodes can be packed into one bin if their overall size is no larger than K. Such a bin is called a *min-height K-feasible bin*, which corresponds to a partitioned subset of S_q. Note that the overall cut size for nodes in a set could be smaller than the sum of their individual cut sizes.

(2) After partitioning S_q into subsets (or min-height K-feasible bins), an intermediate node (also called *bin node*) w_i is created for each bin B_i with $input(w_i) = B_i$ and is labeled $l(w_i) = q$. A buffer node b_i is then

created for each w_i with $input(b_i) = \{w_i\}$ and a label $l(b_i) = q + 1$. All buffer nodes are put into the set S_{q+1}. Note that if some bin B_i contains more than 2 nodes, bin node w_i needs to be further decomposed. However, according to Lemma 4, no matter how w_i is decomposed, the minimum mapping depth of the network does not change. DOGMA arbitrarily decomposes w_i into an unbalanced tree.

DOGMA repeats steps (1) and (2) for stratum $S_q + 1$, and so on, until all strata have been processed. The last bin node corresponds to node v. Note that buffer nodes are introduced only for the packing process, and will be removed when the decomposition is complete.

To determine if there exists a K-feasible cut of height $q - 1$ for a bin $B_i \subseteq S_q$ of nodes, we compute a max-flow in the flow network, constructed as follows [Cong and Ding 1994a]: (i) Create a sink node t with $input(t) = B_i$. (ii) Create a source node s that fanouts to all PIs in N_t. (iii) Assign every edge in N_t an infinite flow capacity. (iv) Replace every node $u \in N_t$, except s and t, by a subgraph (V_u, E_u) where $V_u = \{u_1, u_2\}$ and $E_u = \{(u_1, u_2)\}$ such that $input(u_1) = input(u)$ and $fanout(u_2) = fanout(u)$. Assign (u_1, u_2) an infinite flow capacity if $l(u) = q$, otherwise a unit flow capacity is assigned. (v) Finally, compute a max-flow in the constructed flow network. The amount of flow f corresponds to the min-cut size in the flow network. If $f \leq K$, there exists a min-cut of height $q - 1$ for the bin B_i of nodes.

We illustrate DOGMA for $K = 3$. The output node v in Figure 10(a) is under decomposition. Among the five fanins of v, b, c, d have labels $l(b) = l(c) = l(d) = 2$ and a, e have labels $l(a) = l(e) = 3$. As a result, $S_2 = \{b, c, d\}$ and $S_3 = \{a, e\}$. According to DOGMA, b, c will be packed into one bin, since a K-feasible cut of height 1 exists for them, and d into another bin for a total of two (which is the minimum) min-height K-feasible bins. Then bin nodes f and g with labels $l(f) = l(g) = 2$ and buffer nodes h and i with labels $l(h) = l(i) = 3$ are created for the two bins, respectively (see Figure 10(b)). DOGMA proceeds to the stratum of depth 3. Two K-feasible cuts of height 2 are found for $\{a, h\}$ and $\{i, e\}$, respectively. Again, bin nodes j and k with labels $l(j) = l(k) = 3$ and buffer nodes m and n with labels $l(m) = l(n) = 4$ are created for the two bins, respectively. Nodes m and n are then packed into a bin that corresponds to v (see Figure 10(c)). Finally, nodes g, h, i, m and n are removed and node v is completely decomposed with a label $l(v) = 4$.

The following problem has to be solved in DOGMA.

Min-height K-feasible bin-packing problem. Given a stratum S_q of depth q, pack nodes in S_q into a minimum number of min-height K-feasible bins.

In our study we developed three heuristics to solve the problem. The first-fit-decreasing (FFD) and best-fit-decreasing (BFD) are two heuristics

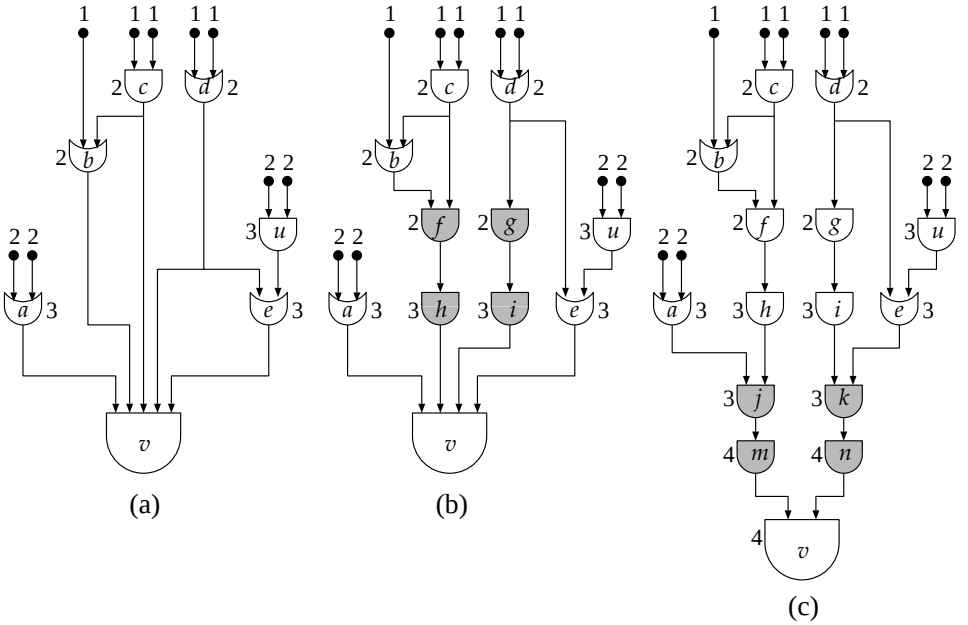

Fig. 10. Decomposition of gate v by the DOGMA algorithm. (a) Before decomposition; (b) b and c, d are packed into f and g; (c) a and h, i and e are packed into j and k.

for the bin-packing problem [Horowitz and Sahni 1978]. The FFD heuristic sorts objects into a list of objects of decreasing sizes, indexes the bins 1, 2, 3, ..., then removes the object from the list (in order) and puts it into the first bin that can accommodate it. The initial conditions on the bins and objects in the BFD heuristic are the same as in the FFD heuristic. But BFD puts the object into the bin that leaves the smallest empty space. For the min-height K-feasible bin-packing problem, we proposed two min-cut-based heuristics, MC-FFD and MC-BFD, which are analogous to FFD and BFD, except that every object is a node whose size is defined to be the size of its min-cut of height $q - 1$. A set of nodes can be packed into a K-feasible bin as long as their combined cut size is no larger than K. The third heuristic is called maximal-sharing-decreasing (MC-MSD), which encourages sharing during packing, i.e., the size of the min-cut for the packed nodes is smaller than the sum of their individual min-cut sizes. The packing that produces the maximum sharing is considered the best-fit packing when MC-MSD calls MC-BFD for a packing result.

Experimental results (Table I) show very few differences on mapping results among the three heuristics (DOGMA followed by CutMap) for MCNC benchmarks. It indicates that in most cases the same number of bins were obtained by the three heuristics. This could be due to the small bin size ($K = 5$) in the experiment. We chose MC-FFD for its efficiency. The FFD heuristic is also used in Chortle-d for packing nodes into bins. However, MC-FFD packs nodes according to the size of their min-height K-feasible cut for better performance. With reconvergent fanouts in general networks,

Table I. Comparing Packing Heuristics MC-FFD, MC-BFD, and MC-MSD In DOGMA

	Bin-Packing Heuristics in DOGMA					
	MC-FF		MC-BFD		MC-MSD	
Circuits	*D*	*A*	*D*	*A*	*D*	*A*
z4ml	2	5	2	5	2	5
count	5	31	5	31	5	31
9symml	5	96	5	96	5	97
cordic	4	23	4	23	4	23
frg1	4	72	4	72	4	72
i3	4	138	4	138	4	138
alu2	7	116	7	116	7	116
x1	4	148	4	148	4	148
C432	11	106	11	106	11	106
alu4	8	209	8	209	8	209
rot	7	267	7	267	7	267
i2	4	79	4	79	4	79
C880	8	92	8	92	8	92
C2670	9	240	9	240	9	240
Dalu	5	345	5	345	5	345
C3540	10	531	10	531	10	531
too_large	5	182	5	182	5	180
i10	13	1245	13	1245	13	1245
t481	7	519	7	519	7	523
C5315	7	439	7	439	7	439
k2	6	460	6	460	6	460
C6288	22	724	22	724	22	724
C7552	9	642	9	642	9	642
Des	5	965	5	965	5	965
total	171	7674	171	7674	171	7677

one cannot decide locally whether a set of nodes can be packed into one bin or not. For example, it is not obvious that nodes e and i in Figure 10(b) can be packed into one bin. The MC-FFD heuristic employs max-flow computation and can decide the packing feasibility correctly.

The time complexity of DOGMA is computed as follows: For every node v in the input network $N = (V, E)$, structural gate decomposition will create $|input(v)| - 2$ nodes. In total, there are $\Sigma_{v \epsilon V}(|input(v)| - 2) = |E| - 2 \cdot |V| = 0(|E|)$ nodes created. The min-height K-feasible cut computation has a time complexity of $O(K \cdot |E|)$ [Cong and Ding 1994a] where K is the LUT input size, and is carried out $O(|input(v)|^2)$ times in the worst case at each node v in the MC-FFD heuristic. Let $d_{\max}$ be the maximal fanin size for nodes in N. Then the time complexity of DOGMA is $O(K \cdot d_{\max}^2 \cdot |E|^2)$. We can reduce the time complexity of min-height cut computation to $O(K \cdot |E_p|)$ by constructing partial flow networks only to a certain depth, where E_p is the edge set of the partial flow network. Let E_{p_max} represent the edge set of the largest partial flow network constructed during decomposition. Then the time complexity of DOGMA is reduced to $O(K \cdot d_{\max}^2 \cdot |E_{p\_max}| \cdot |E|)$.

4.2 Multiple Gate Decomposition

We present our multiple gate decomposition algorithm, called DOGMA-m, and illustrate the procedure on the network shown in Figure 11(a) for $K = 3$. DOGMA-m is outlined in Figure 12.

We call the stratum of each node a *local* stratum. The union of all local strata of depth q is called the *global* stratum of depth q. For each depth q, a node v is *under decomposition* if $|input(v)| > 2$ (i.e., not yet completely decomposed) and $input(v)$ intersets with the global stratum of depth q. Starting from depth $q = 1$ and up, the nodes of the same gate type and also under decomposition will be decomposed simultaneously. In Figure 11(a), nodes $a, b, \ldots, h$ all have a label of 1. Nodes x, y, and z are under decomposition for $q = 1$. The local stratum of depth 1 is $\{a, b, c\}$ for node x, $\{b, c, d, e, f\}$ for node y, and $\{e, f, g, h\}$ for node z, respectively. The global stratum of depth 1 is $\{a, b, c, d, e, f, g, h\}$.

In initialization, buffers are created for PIs to supply inputs to the rest of the network. PIs are labeled 0 and buffers are labeled 1. In Figure 11(a), nodes $a, b, \ldots, h$ are PI buffers. Gray regions represent the global strata of depth 1 and 2 in Figure 11(a)-(c) and (d), respectively. The gate decomposition proceeds as follows:

(1) For each depth q and for each gate type f, the nodes under decomposition are collected into a set G_q^f. Then the global stratum of depth q, denoted S_q, is computed by the union of local strata of depth q for all nodes in G_q^f. In Figure 11(a), let $f = AND$, we have $G_1^f = \{x, y, z\}$ and $S_1 = \{a, b, c, d, e, f, g, h\}$. Based on G_q^f and S_q, we formulate the *Global Stratum Bin-Packing (GSBP)* problem (to be formally defined later). By solving the GSBP problem, we achieve (i) for each node in G_q^f, its local stratum of depth q is packed into min-height K-feasible bins, and (ii) there are a minimum number of min-height K-feasible bins in total. The second objective is achieved by packing common fanins for the nodes in G_q^f. Intermediate nodes (also called *bin nodes*) are created for bins. In Figure 11(b), nodes b and c, e and f, g and h are packed into bin nodes i, j and k, respectively.

(2) It is possible that some nodes in G_q^f have been decomposed completely (e.g., nodes x and z in Figure 11(b)), while the local strata of other nodes can be packed further (e.g., node y in Figure 11(b)). Both G_q^f and S_q are updated and a new instance of the GSBP problem for the same q value is formulated and solved. The process iterates until the global stratum of depth q has been minimally packed into bins (as a result, the network does not change). In Figure 11(b), we have $l(x) = 1$, $l(z) = 2$, $G_1^f = \{v, y\}$, and $S_1 = \{i, d, j, x\}$. By solving the GSBP problem for the updated G_q^f and S_1, node d and i are packed into a bin node m. Node y is now completely decomposed with a label $l(y) = 2$. The

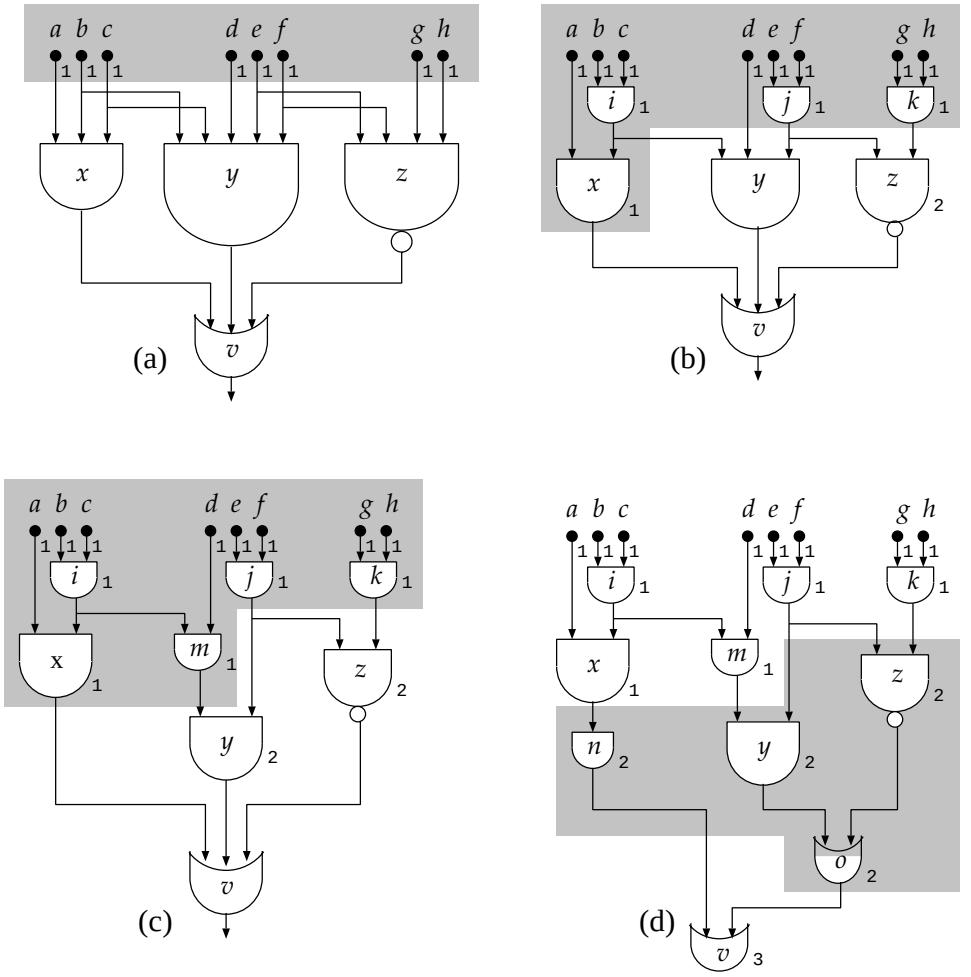

Fig. 11. Multiple gate decomposition. (a) Initial network; (b) after one $q = 1$ iteration; (c) after two $q = 1$ iterations; (d) completely decomposed network.

process iterates with updated $G_q^f = \{v\}$ and $S_1 = \{x\}$. But no further packing is possible for $q = 1$ (see Figure 11(c)).

(3) Buffer nodes are created and labeled $q + 1$ for every fanin in the global strata S_q. The decomposition process iterates steps (1) and (2) until the network is 2-bounded. In Figure 11(d), a buffer node n is created for node x, nodes y and z are then packed into a bin, and the decomposition of node v is completed.

Two points are worth mentioning. First, in DOGMA, each node is decomposed only after all its fanins have been decomposed and labeled. In DOGMA-m, however, nodes could undergo decomposition, even though some of their fanins have not been labeled. For example, node v in Figure 11(b) is

```
procedure DOGMA-m ( N, K )
/* N is the input network and K is LUT input size. */

1   Initialization
2   N_old = ∅
3   for q = 1,2,... until N is 2-bounded do
4       while N ≠ N_old do
5           N_old = N
6           for each gate function type f do
7               G_q^f = { v | func(v) = f, |input(v)| > 2, ∃u ∈ input(v) s.t. label(u) = q }
8               S_q = { u | label(u) = q, u ∈ input(v), v ∈ G_q^f }
9               Solve GSBP( G_q^f, S_q, K ) problem
10              for each min-height K-feasible bin B_i created in GSBP do
11                  create bin node w_i, label(w_i) = q
12                  add w_i to N, update fanins of nodes in G_q^f
13      for each node u_i ∈ S_q do
14          create buffer node b_i, label(b_i) = q + 1
15          add b_i to N, N_old = ∅
16  return N
```

Fig. 12. Multiple gate decomposition algorithm.

under decomposition ($v \in G_1^f$), while its fanin y is not labeled yet. Second, for each depth q and gate type f, multiple instances of the GSBP problem might be solved in order to pack local strata into a minimal number of bins. For example, two instances of the GSBP problem are solved for $q = 1$ before the local stratum of node y is minimally packed (from Figure 11(a) to (c)). In our experiments, we found that solving three instances of the GSBP problem are sufficient for each q value.

The Global Stratum Bin-Packing (GSBP) problem is formally defined as follows.

Global stratum bin-packing (GSBP) problem. Given a set G_q^f of nodes of gate type f under decomposition and a global stratum S_q of depth q that contain fanins of nodes from G_q^f, pack the fanins in S_q into a set of bins such that (i) for each node in G_q^f, its local stratum of depth q is packed into min-height K-feasible bins; (ii) there is a minimum number of min-height K-feasible bins in total.

To solve the GSBP problem, we build a matrix M where rows correspond to nodes in $G_q^f = \{v_1, v_2, \ldots, v_n\}$, columns correspond to fanins in $S_q = \{u_1, u_2, \ldots, u_m\}$, an entry $M(i, j) = 1$ if $u_j \in input(v_i)$, and $M(i, j) = 0$ if not. A *rectangle* is a subset of rows and columns, denoted by a pair (R, C), indicating the row and column subsets, where all entries are 1. C corresponds

	a	b	c	d	e	f	g	h
x	1	1	1	0	0	0	0	0
y	0	1	1	1	1	1	0	0
z	0	0	0	0	1	1	1	1
weight	1	2	2	1	2	2	1	1

(a)

	a	b	c	d	e	f	g	h
x	1	0	0	0	0	0	0	0
y	0	0	0	1	0	0	0	0
z	0	0	0	0	0	0	1	1
weight	1	0	0	1	0	0	1	1

(b)

Fig. 13. FFD bin-packing heuristic for the GSBP problem. (a) Initial M; (b) M after the first run of bin-packing.

to a bin of fanins and R corresponds to a set of nodes that share fanins in C. A solution of the GSBP problem is a rectangle cover for M, subject to a K-feasible cut of height $q - 1$ exists for fanins in each column set C. This matrix representation is similar to the cube-literal matrix used for solving the cube-extraction problem [Rudell 1989; De Micheli 1994]. However, the algorithms for cube extraction cannot be applied directly because the C in every rectangle (R, C) must satisfy the K-feasible cut constraint.

We use the MC-FFD packing heuristic to compute a rectangle cover for the GSBP problem as follows. First, compute the fanout factor $o_j = \Sigma_{i=1}^{n} M(i, j)$ and the cut size s_j of min-cut of height $q - 1$ for every fanin $u_j \in S_q$. The weight of each fanin is $o_j \cdot s_j$. Then we sort the fanins according to their weights and follow the MC-FFD bin-packing heuristic to pack fanins into bins (starting from the fanin with the largest weight). Our strategy is to group fanins of large cut sizes for obtaining a minimum number of bins and to group fanins of large fanout sizes for exploiting common fanins. A set of fanins can be packed into one bin C if (i) a K-feasible cut of height $q - 1$ exists for the fanins in C, and (ii) the largest rectangle (R, C) satisfies $|R| \geq r_{min}$ (i.e., at least r_{min} nodes in G_q^f share these fanins) where r_{min} is a user-specified parameter. By performing the MC-FFD packing heuristic, we obtain a set of rectangles. Each rectangle (R, C) that satisfies $|C| \geq c_{min}$ (another user-specified parameter) will be saved and covered with 0's in M. The MC-FFD packing procedure is repeated until M contains only 0's. A rectangle cover for M is then obtained, and the set C in each rectangle corresponds to a bin. In our implementation, we set $r_{min} = 2$ and $c_{min} = 2$ in the first pass of the MC-FFD packing procedure, and decrease both values to 1 in subsequent iterations. The decrease of values guarantees the termination of our procedure.

We demonstrate the MC-FFD packing heuristic on the network in Figure 11(a) for $K = 3$ for solving the GSBP problem. The initial matrix M is shown in Figure 13(a). The rows correspond to nodes in $G_1^f = \{x, y, z\}$ and the columns correspond to fanins in $S_1 = \{a, b, c, d, e, f, g, h\}$. The weight of each fanin is its fanout size (i.e., the number of 1's in each column), since every fanin is a PI buffer whose cut size is 1. Fanins are

Table II. Circuit Optimization Using the Rugged Script

	Original			Rugged			
		gate fanin size				gate fanin size	
Circuits	ckt size	3	>3	time(s)	ckt size	3	>3
z4ml	63	10%	89%	0.5	25	8%	0%
count	111	14%	0%	1.4	79	22%	20%
9symml	153	34%	8%	20.4	96	28%	35%
cordic	73	11%	8%	1.3	36	22%	28%
frg1	120	1%	94%	0.1	69	19%	43%
i3	70	0%	6%	2.2	78	0%	26%
slu2	210	17%	53%	29.9	172	19%	16%
x1	325	9%	76%	5.7	165	15%	35%
C432	159	1%	11%	55.5	101	13%	29%
alu4	416	13%	47%	22.0	374	16%	9%
rot	494	21%	39%	17.1	392	21%	18%
i2	35	20%	31%	8.8	186	80%	1%
C880	302	9%	4%	6.4	232	16%	9%
C2670	695	16%	6%	1.0	452	7%	20%
dalu	1939	10%	4%	3.0	595	42%	7%
C3540	956	18%	4%	8.0	716	27%	15%
too_large	1038	0%	100%	7.0	137	21%	35%
i10	1634	14%	17%	58.4	1481	15%	23%
t481	2056	42%	44%	2.0	381	5%	13%
C5315	1454	29%	8%	51.7	1094	15%	9%
k2	250	1%	98%	1.0	424	23%	14%
C6288	2353	0%	0%	1.0	2139	12%	1%
C7552	2114	8%	7%	1.0	1530	10%	12%
des	1805	24%	56%	3.0	2053	10%	9%
total	18824	16%	26%	317.4	13007	165	12%

sorted into the order b, c, e, f, a, d, g, h according to their weights. Nodes b and c are packed into the first bin, which corresponds to the rectangle $(R_1, C_1) = (\{x, y\}, \{b, c\})$. Although there is a 3-feasible cut of height 0 for nodes b, c, e, they cannot be packed into one bin because the rectangles for them have $|R| = |\{y\}| < r_{min} = 2$. As a result, node e is put into a separate bin and packed with node f, which corresponds to the rectangle $(R_2, C_2) = (\{y, z\}, \{e, f\})$. Then the two rectangles are covered with 0's (Figure 13(b)). We reset $r_{min} = c_{min} = 1$ and perform another run of the MC-FFD packing heuristic. Three bins are obtained but only one bin contains two fanins. Totally, three bin nodes will be created. The network in Figure 11(a) is now decomposed into the network in Figure 11(b).

5. EXPERIMENTAL RESULTS

We implemented DOGMA and DOGMA-m in the C language and incorporated them into the RASP logic synthesis system for FPGAs [Cong et al. 1996]. We prepared two sets of benchmarks in our experiments. The first set $C_{original}$ consists of 24 original multilevel MCNC benchmarks, which all

contain a large percentage of 2-unbounded gates (i.e., 3 or more inputs). We performed the rugged script in SIS [Sentovich et al. 1992] for technology-independent optimization, and obtained the second set C_{rugged} of benchmarks. Both sets of benchmarks were transformed into simple gate networks using AND-OR decomposition. Table II shows the circuit sizes and fanin distributions of the two sets of simple gate networks. The benchmark set $C_{original}$ contains 18,824 simple gates with 42% of them being 2–unbounded, while the benchmark set C_{rugged} contains 13,007 simple gates with 28% of them 2–unbounded. Clearly, both circuit size and fanin size were reduced by performing the rugged script. The total runtime is less than six minutes on ULTRA2.

We compared DOGMA and DOGMA-m with three structural gate decomposition algorithms, as well as DOGMA-m with algebraic and Boolean decomposition approaches. The three structural gate decomposition algorithms used for comparison are the tech_decomp algorithm [Sentovich et al. 1992]; the dmig algorithm [Wang 1989; Chen et al. 1992]; and our implementation of the Chortle-d algorithm [Francis et al. 1991b]. After gate decomposition by each of these algorithms, CutMap [Cong and Hwang 1995] was employed to obtain depth-optimal mapping solutions. For a comparison across structural, algebraic, and Boolean gate decompositions, we employed DOGMA-m, speed_up in SIS [Sentovich 1992] and the TOS package [Legl et al. 1996a] to perform decompositions, respectively. Again, CutMap was employed to perform LUT mapping, except for TOS, since it produced LUT networks directly. The objective of gate decomposition and LUT mapping in our experiments was to minimize mapping depth. CutMap also minimizes mapping area as the second objective. All experiments were performed on a Sun ULTRA2 workstation with 256M of memory.

We first demonstrate the impact of *further* gate decomposition on depth and area in technology mapping. According to Theorem 1, the mapping solution space expands regardless of the gate decomposition algorithm used. We use tech_decomp to decompose benchmarks in C_{rugged} into 5-bounded networks, and subsequently into 2-bounded networks, followed by LUT mapping to obtain mapping solutions. The sizes of 5-bounded networks increase substantially compared to the 5-unbounded networks in C_{rugged}. However, the percentages of 2-unbounded gates are about the same. We employed CutMap [Cong and Hwang 1995] and DFMap [Cong and Ding 1994b] to produce depth-optimal and *duplication-free* area-optimal mapping solutions, respectively. In Table III, we see that both the optimal mapping depth (by CutMap) and the optimal duplication-free mapping area (by DFMap) are reduced by 16% when the 5-bounded networks are further decomposed into 2–bounded networks.

We compared five structural gate decomposition algorithms (tech_decomp, dmig, Chortle-d, DOGMA, and DOGMA-m) next, on benchmarks in $C_{original}$ and C_{rugged}, using CutMap as the mapping engine. The depth and area of mapping solutions as well as the runtimes of the compared algorithms (not including CutMap time) for the two sets of benchmarks are presented in

Table III. Comparison of Results for 5-Bounded and 2-Bounded Networks

	5-bounded			CutMap				DFMap			
	ckt	gate fanin		5-bounded		2-bounded		5-bounded		2-bounded	
Circuits	size	3	>3	*D*	*A*	*D*	*A*	*D*	*A*	*D*	*A*
z4ml	25	8	0	2	5	2	5	4	11	4	11
count	79	22	20	5	31	5	31	16	46	17	46
9symml	131	24	28	7	90	6	105	8	93	10	87
cordic	61	15	15	6	24	5	21	6	29	6	23
frg1	108	15	30	6	61	5	65	10	68	11	59
i3	142	0	59	4	126	4	154	4	142	5	130
alu2	201	20	16	12	118	9	124	19	139	18	122
x1	231	16	24	4	150	4	160	7	167	6	134
C432	158	8	18	15	109	11	112	18	133	18	84
alu4	434	21	9	13	223	10	241	21	258	25	230
rot	459	21	12	11	243	9	270	14	254	14	236
i2	300	50	12	6	120	5	94	6	141	6	88
C880	259	14	8	9	113	8	99	14	106	14	100
C2670	556	9	13	12	245	10	250	16	264	18	201
dalu	721	36	5	7	287	6	344	16	306	16	301
C3540	883	24	12	13	483	11	563	22	536	22	399
too_large	219	18	23	7	157	6	162	9	162	10	130
i10	2188	13	16	23	1248	17	1217	39	1403	39	1036
t481	790	16	12	9	502	8	525	13	495	13	442
C5315	1118	15	8	7	478	7	430	13	582	14	413
k2	552	23	14	8	394	7	433	17	435	17	396
C6288	2139	12	1	22	684	22	727	88	1238	88	1209
C7552	1894	10	10	17	770	13	638	20	835	19	601
des	2359	13	8	7	1151	6	1087	12	1173	12	1082
total	16007	16	11	232	7812	196	7857	412	9016	422	7560
ratio				1.00	1.00	0.84	1.01	1.00	1.00	1.02	0.84

Tables IV and V. In comparison to DOGMA-m, we see that the other four algorithms result in up to 11% larger mapping depth and 50% larger mapping area on the benchmark set $C_{original}$, and up to to 16% larger mapping depth and 10% larger mapping area on the benchmark set C_{rugged}. The differences in mapping depth obtained by DOGMA-m, dmig, or DOGMA are marginal, while the differences in mapping area are more significant. Regarding runtime, DOGMA-m is comparable to DOGMA runtime, but is 8 to 33 times slower than the runtimes of the other three algorithms. However, DOGMA-m runtime is in the same order of magnitude as the time spent in performing the rugged script or CutMap.

Comparing Tables IV and V, we see that the mapping area for C_{rugged} is 30% to 50% *smaller*, while the mapping depth for C_{rugged} is 1% to 7% *larger* than that for $C_{original}$. It shows that the rugged script, which performs logic optimization based on algebraic divisions, is very effective for area minimization, but not as effective for depth minimization. A benefit resulting from the area reduction is the significant decrease of runtime for all decomposition algorithms. For benchmarks in C_{rugged}, DOGMA-m results in 10% smaller

Table IV. Comparing Results Using tech_decomp, dmig, chortle-d, DOGMA and DOGMA-m for Gate Decomposition Followed by CutMap for Circuits in $C_{original}$

	Structural Gate Decomposition Algorithms														
	tech_decomp			dmig			chortle-d			DOGMA			DOGMA-m		
Circuits	*D*	*A*	T(s)	*D*	*A*	T(s)	*D*	*A*	T(s)	*D*	*A*	T(s)	*D*	*A*	T(s)
z4ml	4	32	0.3	3	31	0.3	4	32	0.6	3	17	0.6	3	19	0.5
count	5	31	0.2	5	31	0.2	5	31	0.6	5	31	1.4	5	31	0.3
9symml	5	60	0.3	5	60	0.4	5	60	0.8	4	57	1.1	4	59	0.6
cordic	4	19	0.1	4	18	0.1	4	18	0.4	3	17	0.7	3	17	0.3
frg1	5	252	0.9	5	235	2.2	5	209	3.0	5	209	5.8	5	180	7.9
i3	3	42	0.2	3	42	0.2	3	42	0.4	3	42	0.5	3	42	0.4
alu2	10	268	0.7	9	245	0.9	9	253	2.0	8	240	3.9	8	192	5.7
x1	5	633	2.2	5	662	3.4	5	571	7.3	5	557	10.5	5	370	13.1
C432	9	81	0.3	8	79	0.3	9	75	1.0	8	79	11.3	8	82	2.7
alu4	11	422	1.4	9	423	1.7	9	427	3.9	9	440	13.5	8	354	15.8
rot	10	424	1.6	8	385	1.9	8	377	4.5	8	381	10.4	8	367	17.7
i2	5	70	0.2	5	75	0.5	4	66	0.7	4	60	1.2	4	56	1.1
C880	7	145	0.5	7	144	0.6	7	140	1.7	7	141	9.9	7	142	1.7
C2670	7	223	1.6	7	218	2.0	7	223	4.5	7	218	20.9	7	218	5.7
dalu	9	507	3.7	9	513	4.3	9	507	12.0	9	506	175.8	9	497	19.9
C3540	11	469	2.2	10	455	2.4	11	468	6.5	10	459	98.8	10	462	24.1
too_large	7	4867	26.3	7	4700	297.4	6	3913	137.6	6	3867	456.9	6	2124	1680.7
i10	12	924	4.3	10	958	5.2	10	963	13.6	10	958	128.1	10	853	162.2
C481	7	1588	5.6	6	1569	6.6	7	1321	20.5	7	1321	21.1	7	925	123.9
C5315	8	592	3.6	8	565	4.3	8	558	11.1	7	565	45.8	7	526	32.0
k2	6	1121	3.2	6	1177	13.1	5	1164	12.5	5	1125	45.4	5	712	78.2
C6288	22	728	4.5	22	728	5.1	22	728	85.9	22	728	854.9	22	728	42.6
C7552	6	648	5.1	7	649	5.6	7	648	16.6	6	644	101.5	6	633	56.5
des	5	1880	8.5	5	1822	11.5	6	2264	30.2	5	2203	71.8	5	1105	750.3
total	183	16026	77.5	173	15784	370.2	175	15058	377.9	166	14865	2091.8	165	10694	3043.9
ratio	1.11	1.50	0.03	1.05	1.48	0.12	1.06	1.41	0.12	1.01	1.39	0.69	1.00	1.00	1.00

area compared to the other four algorithms. It shows that DOGMA-m can exploit common fanins for area minimization, in addition to the rugged script.

Finally, we employed DOGMA-m, speed_up and TOS for a comparison across structural, algebraic, and Boolean gate decomposition approaches. We configured TOS for delay-oriented synthesis in the *medium-effort* mode performing both single-output (TOS-s) and multioutput (TOS-m) functional decompositions. The input circuits to TOS were prepared as follows. We first tried to collapse each benchmark in C_{rugged} into a flat logic network within 30 minutes of CPU time. If this could not be done, we used the reduce_depth -depth d command in TOS to collapse benchmarks into networks of the smallest depth d where $d \geq 2$. We allocated 30 minutes of CPU time for each depth d starting from $d = 2$ and up. Among all benchmarks after collapsing, rot and C880 have a depth of 2, C432, C2670, C5315, and C7552 have a depth of 3, 3540 and i10 have a depth of 4, and C6288 has a depth of 6. The remaining benchmarks are completely collapsed.

Table VI collects the mapping results from DOGMA-m + CutMap, speed_up + CutMap, TOS-s, and TOS-m. Subtotal 1, subtotal 2, and subtotal 3 are totals of the mapping results for benchmarks that speed_up, TOS-s, and TOS-m finish, respectively, and the ratios measure the relative

Table V. Comparing Results Using tech_decomp, dmig, chortle-d, DOGMA and DOGMA-m for Gate Decomposition Followed by CutMap for Circuits in C_{rugged}

	Structural Gate Decomposition Algorithms														
	tech_decomp			dmig			chortle-d			DOGMA			DOGMA-m		
Circuits	*D*	*A*	T(s)	*D*	*A*	T(s)	*D*	*A*	T(s)	*D*	*A*	T(s)	*D*	*A*	T(s)
z4ml	2	5	0.0	2	5	0.1	2	5	0.1	2	5	0.2	2	5	0.0
count	5	31	0.2	5	31	0.2	5	31	0.5	5	31	0.7	5	31	0.4
9symml	6	105	0.3	5	102	0.4	5	107	0.7	5	96	1.4	5	95	1.5
cordic	5	21	0.1	4	24	0.1	4	23	0.3	4	23	0.4	4	20	0.2
frg1	5	65	0.2	5	63	0.3	4	81	0.6	4	72	1.0	4	63	1.1
i3	4	154	0.3	4	154	0.4	4	138	0.8	4	138	1.3	4	73	1.3
alu2	9	124	0.4	8	122	0.5	8	121	1.1	7	116	3.6	7	107	2.5
x1	4	160	0.5	4	160	0.5	4	156	1.1	4	148	1.6	3	131	1.5
C432	11	112	0.3	11	115	0.3	11	106	0.9	11	106	5.8	11	99	3.5
alu4	10	241	0.8	9	211	1.0	10	226	2.6	8	209	15.5	8	207	6.9
rot	9	270	1.0	7	259	1.1	8	265	2.5	7	267	6.2	7	261	5.4
i2	5	94	1.0	5	120	6.4	5	100	4.5	4	79	7.7	4	82	7.9
C880	8	99	0.5	8	100	0.6	8	97	1.4	8	98	5.6	8	90	1.9
C2670	10	250	1.1	9	234	1.3	10	229	2.9	9	241	7.0	8	222	5.7
dalu	6	344	1.5	5	358	1.7	5	359	3.8	5	345	8.1	5	325	8.5
C3540	11	563	1.8	10	540	2.1	11	523	6.0	10	540	56.2	10	530	27.7
too_large	6	162	0.4	5	161	0.6	5	185	1.1	5	182	2.6	5	149	2.9
i10	17	1217	4.2	13	1182	5.0	14	1192	14.3	13	1248	86.3	13	1158	306.7
C481	8	525	1.3	8	526	1.9	7	561	4.3	7	519	17.4	7	315	24.4
C5315	7	430	2.4	7	434	2.6	7	410	6.8	7	438	24.0	7	445	10.4
k2	7	433	1.2	6	439	1.5	6	490	2.9	6	460	9.0	6	423	10.2
C6288	22	727	4.3	22	690	4.9	22	690	19.4	22	724	769.2	22	723	192.0
C7552	13	638	3.7	9	685	4.0	11	614	10.4	9	639	51.1	9	621	87.9
des	6	1087	4.5	5	1058	5.3	6	1127	14.4	5	965	49.0	5	969	208.5
total	196	7857	32.0	176	7773	42.8	182	7836	103.4	171	7689	1130.9	169	7144	919.0
ratio	1.16	1.10	0.03	1.04	1.09	0.05	1.08	1.10	0.11	1.01	1.08	1.23	1.00	1.00	1.00

performance of these approaches with respect to DOGMA-m + CutMap. Time T(s) reports the computation time in seconds. In Table VI, we see that DOGMA-m + CutMap is able to map all benchmarks in 23 minutes, while speed_up + CutMap, TOS-s, and TOS-m fail to map some benchmarks in 2 hours. Compared to DOGMA-m + CutMap, speed_up + CutMap takes more than 5 hours (98% consumed by *speed_up*) to map 23 benchmarks (not including des), but obtains significantly better results: 13% smaller mapping depth and 6% smaller mapping area. The results for C432 show the largest contrast between the performance of speed_up and the efficiency of DOGMA-m. speed_up results in a mapping depth of 8 in more than 2 hours, while DOGMA-m results in a mapping depth of 11 in 6.6 seconds. TOS-s and TOS-m do not return mapping solutions in allocated CPU times for 3 and 8 benchmarks, respectively. Compared to the other two approaches, TOS-s obtains smaller mapping depth on count, 9symml, alu2, alu4, and t481. TOS-m obtains a smaller mapping area on 9symml, cordic, x1, alu2, and t481. It is worth noting that TOS is extremely successful for 9symml and t481. The results indicate that mapping approaches based on functional decomposition require longer computation time to obtain good results, especially on circuits of medium to large sizes.

Table VI. Mapping Results from Structural (DOGMA-m), Algebraic (speed_up), and Boolean (TOS) Gate Decomposition Approaches to C_{rugged}

	Technology Mapping Gate Decomposition and LUT Mapping Algorithms											
	DOGMA-m			speed_up			TOS-TOM (medium effort)					
	CutMap						single-output			multiple-output		
Circuits	*D*	*A*	T(s)	*D*	*A*	T(s)	*D*	*A*	T(s)	*D*	*A*	T(s)
z4ml	2	5	0.2	2	5	2.8	2	7	0.2	2	5	0.5
count	5	31	1.0	3	52	12.2	2	42	8.4	3	38	24.7
9symml	5	95	4.0	5	85	54.2	3	7	0.5	3	7	0.5
cordic	4	20	0.8	3	17	8.5	3	13	12.0	3	11	23.1
frg1	4	63	4.1	4	51	149.9	7	52	17.2	8	54	36.1
i3	4	73	7.6	3	42	17.3	--	--	--	--	--	--
alu2	7	107	6.6	7	127	318.1	4	46	3.1	5	51	48.3
x1	3	131	5.7	3	121	12.0	4	129	25.8	5	116	89.3
C432	11	99	6.6	8	95	7392.6	9	250	173.1	10	193	615.8
alu4	8	207	23.4	8	250	596.4	7	286	37.3	-	-	-
rot	7	261	13.2	6	251	71.1	7	404	117.5	8	291	612.0
i2	4	82	14.4	4	69	14.1	--	--	--	--	--	--
C880	8	90	4.4	6	135	49.0	8	272	104.5	9	196	726.5
C2670	8	222	12.7	6	186	801.0	8	391	77.1	7	302	199.8
dalu	5	325	37.3	5	350	1992.8	9	606	5216.7	--	--	--
C3540	10	530	56.3	10	444	320.6	12	662	214.1	12	511	1373.0
too_large	5	149	7.5	5	112	24.3	9	324	465.0	8	168	1395.4
i10	13	1158	416.0	11	838	572.5	11	1987	2278.3	--	--	--
t481	7	315	54.8	6	265	52.0	3	5	1.9	3	5	1.9
C5315	7	445	25.4	6	443	83.3	8	834	255.6	8	683	639.0
k2	6	423	36.1	6	457	668.2	11	3012	4368.2	--	--	--
C6288	22	723	213.0	18	977	5555.8	--	--	--	--	--	--
C7552	9	621	123.3	7	454	142.9	8	1261	829.3	9	918	3956.3
des	5	969	263.9	--	--	--	4	704	1586.8	--	--	--
subtotal1	164	6175	1074	142	5826	18912						
subtotal2	139	6266	1103.3				139	11294	15792.6			
subtotal3	102	3184	326.6							103	3549	9742.2
ratio	1.00	1.00	1.00	0.87	0.94	17.60	1.00	1.80	14.31	1.01	1.11	29.83

Overall, from these experiments, we conclude that DOGMA-m can obtain the *best* mapping results among the five structural gate decomposition algorithms, and is much more efficient in terms of runtime (over 17 and 30 times faster) compared to the algebraic decomposition algorithm speed_up and the functional decomposition approach TOS. However, speed_up obtains the best results among the approaches.

6. CONCLUSION

In this paper we presented an in-depth study of structural gate decomposition for depth-optimal technology mapping in LUT-based FPGA designs. We show that any structural gate decomposition in K-bounded networks can only result in a smaller depth in K-LUT mapping solutions, regardless of the decomposition algorithm used. Therefore, it is always beneficial to

decompose circuits into 2-bounded networks for depth minimization when structural decompositions are applied. We prove that the structural gate decomposition problem in-depth-optimal technology mapping is NP-hard for K-unbounded networks when the LUT input size $K \geq 3$, and remains NP-hard for K-bounded networks when $K \geq 5$. We propose two new algorithms, called DOGMA and DOGMA-m, which combine the level-driven node-packing technique in Chortle-d and the network flow-based labeling technique in FlowMap, for structural gate decomposition. DOGMA-m decomposes multiple gates simultaneously to exploit common fanins. The following experimental results were observed: First, the optimal mapping depth and the optimal duplication-free mapping area can be reduced when 5-bounded networks are decomposed structurally into 2–bounded networks. Second, applying the rugged script for technology-independent logic optimization before technology mapping can result in a 40% to 50% reduction, with only a marginal increase in depth, while significantly reducing the runtime of the structural decomposition algorithms. Third, DOGMA-m results in the smallest mapping depth and mapping area among the five structural gate decomposition algorithms. Finally, comparing three algorithms, DOGMA-m, speed_up, and TOS, which take, respectively, structural, algebraic, and Boolean (functional decomposition) gate decomposition approaches, DOGMA-m can decompose all tested benchmarks in a short time, while speed_up and TOS fail to obtain results on some benchmarks. However, speed_up results in 13% smaller depth and 6% smaller area final mapping solutions, compared to DOGMA-m.

APPENDIX: Proofs

We prove Lemma 5 and Lemma 6 here. In Figure 5, every internal node in $N(x_i)$ has a mapping depth of 1, except node s_i has a mapping depth of 2. The mapping depth of nodes x_i and $\bar{x}_i$ depend on the way node s_i is decomposed. In Figure 6, nodes $q_j^k (1 \leq k \leq 2K - 5)$ in $N(C_j)$ have a mapping depth of 2, since each of them is the root of a complete 2-level K-ary tree with PI nodes as leaves. Similarly, nodes $r_j^k (1 \leq k \leq K - 2)$ have a mapping depth of 3. The mapping depth of node C_j in $N(C_j)$ depends on the mapping depth of its three literal nodes l_j^1, l_j^2, l_j^3. Since every literal node has single fanin from its variable node, the mapping depth of nodes $C_j (1 \leq j \leq m)$ depend on how nodes $s_i (1 \leq i \leq n)$ are decomposed. The network $N(F)$ has m primary outputs. Let $D(N(F))$ be a decomposed network of $N(F)$. Then $MMD(D(N(F)))$, the mapping depth of $D(N(F))$, depends on how the node s_i is decomposed in every $N(x_i)$.

PROPOSITION 1. *Let $D(N(x_i))$ be a decomposed subnetwork of $N(x_i)$. If $MMD_{D(N(x_i))}(x_i) = 2$, then $MMD_{D(N(x_i))}(\bar{x}_i) = 3$, and if $MMD_{D(N(x_i))}(\bar{x}_i) = 2$, then $MMD_{D(N(x_i))}(x_i) = 3$.*

PROOF. Node s_i can be decomposed in six different ways, as shown in Figure 14. In Figure 14 (a and b), nodes PI_i^1 and PI_i^2 feed into an

intermediate node t_i and subsequent decomposition steps are applied. A cut of height 1 is created for s_i with cutset $\{w_i^1, w_i^2, t_1\}$. Together with nodes v_i^1 through v_i^{K-3}, there exists a K-feasible cut of height 1 for node x_i. Hence $MMD_{D(N(x_i))}(x_i) = 2$. Under this decomposition, however, the min-cut of height 1 for node $\bar{x}_i$ has a size of $K + 1$. Hence $MMD_{D(N(x_i))}(\bar{x}_i) = 3$. In Figure 14(c), the min-cut of height 1 for node s_i has a size of 4. Hence $MMD_{D(N(x_i))}(x_i) = MMD_{D(N(x_i))}(\bar{x}_i) = 3$. In Figure 14 (d and e), the min-cut of height 1 for node s_i has a size of 3. But due to the edge (w_i^2, x_i), the min-cut of height 1 for node x_i still has a size of $K + 1$. Hence $MMD_{D(N(x_i))}(x_i) = MMD_{D(N(x_i))}(\bar{x}_i) = 3$. In Figure 14(f), w_i^1 merges with PI_i^1 and w_i^2 merges with PI_i^2. A cut of height 1 is created for node s_i with cutset $\{t_1, t_2\}$. Together with nodes u_i^1 through u_i^{K-2}, a K-feasible cut of height 1 for node $\bar{x}_i$ exists. Hence $MMD_{D(N(x_i))}(\bar{x}_i) = 2$. But under this decomposition, the min-cut of height 1 for node x_i has a size of $K + 1$, since both w_i^1 and w_i^2 connect to x_i. Hence $MMD_{D(N(x_i))}(x_i) = 3$. From above enumerative decompositions of node s_i, it is easy to see whether $MMD_{D(N(x_i))}(x_i) = 2$, then $MMD_{D(N(x_i))}(\bar{x}_i) = 3$ (Figure 14 (a and b)), and if $MMD_{D(N(x_i))}(\bar{x}_i) = 2$, then $MMD_{D(N(x_i))}(x_i) = 3$ (Figure 14(f)). This proves Proposition 1. □

PROPOSITION 2. *There exist 2–feasible cuts of height 2 for nodes x_i and $\bar{x}_i$ respectively.*

PROOF. A 2-feasible cut of height 2 for node x_i can be obtained by decomposing x_i as follows: Build a binary tree with nodes $v_i^1, \ldots, v_i^{K-3}$ and w_i^1, w_i^2 as leaves and denote the root node as v_i. Let v_i feed into x_i. Then v_i has a mapping depth of 2 and $\{s_i, v_i\}$ is a 2-feasible cut of height 2 for node x_i. Similarly, build a binary tree with nodes $u_i^1, \ldots, u_i^{K-2}$ as leaves and denote the root node as u_i. Let u_i feed into $\bar{x}_i$. Then u_i has a mapping depth of 2 and $\{s_i, u_i\}$ is a 2-feasible cut of height 2 for node $\bar{x}_i$ □

PROPOSITION 3. *Let $D(N(C_j))$ be a decomposed subnetwork of $N(C_j)$. Then $MMD_{D(N(C_j))}(C_j) = 4$ if and only if $MMD_{D(N(C_j))}(l_j^k) = 2$ for some $1 \leq k \leq 3$.*

PROOF. Because every literal node has only one fanin, the mapping depth of a literal node is equal to the mapping depth of its variable node. In subnetwork $N(C_j)$, assume one literal node has a mapping depth equal to 2 (e.g. l_j^3 in Figure 6(b)). For the other two literal nodes of mapping depth 3, there exist 2-feasible cuts of height 2 for each of them (Proposition 2). Together with nodes $q_j^1, \ldots, q_j^{2K-5}$ in the fanins of node C_j, there will be $2K$ nodes of mapping depth 2. Hence two intermediate nodes of mapping depth 3 will be introduced during the decomposition of node C_j. Since node C_j has $K - 2$ fanins of mapping depth 3 (i.e., $r_j^1, \ldots, r_j^{K-2}$), a K-feasible cut of height 3 exists for node C_j. Hence $MMD_{D(N(C_j))}(C_j) = 4$ (see Figure 6(b)). If C_j contains more than one literal node of mapping depth 2, the

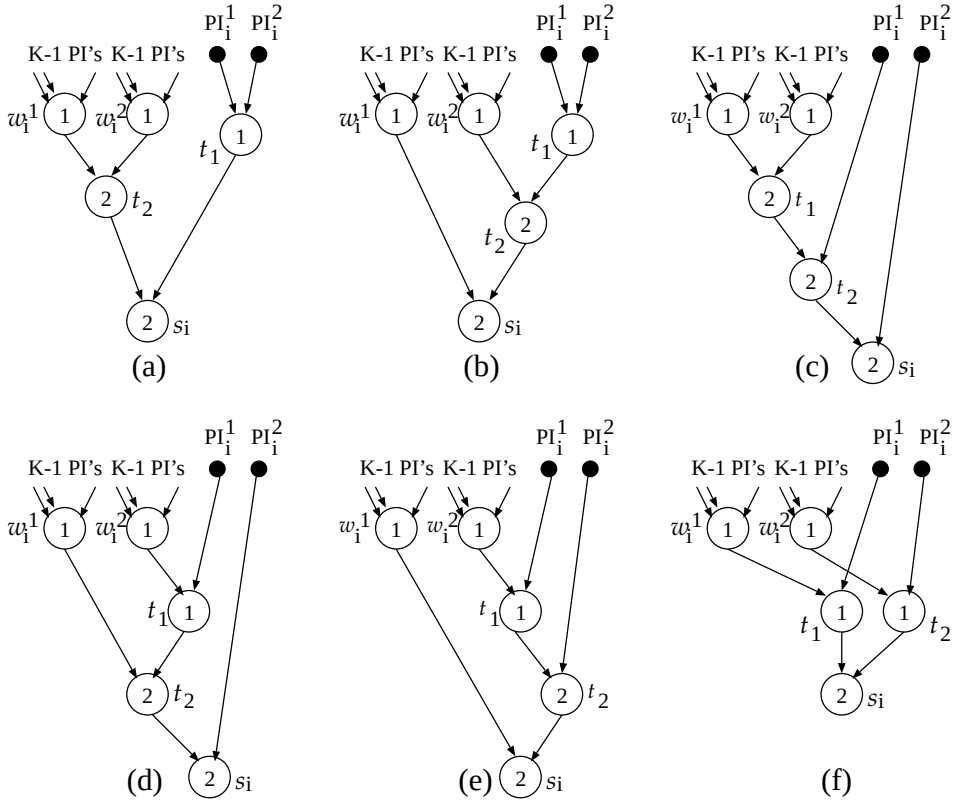

Fig. 14. Six different ways to decompose node s_i in $N(x_i)$. In (a) and (b), $MMD(x_i) = 2$ and $MMD(\bar{x}_i) = 3$; in (c), (d), and (e), $MMD(x_i) = MMD(\bar{x}_i) = 3$; only in (f) $MMD(x_i) = 3$ and $MMD(\bar{x}_i) = 2$.

result still holds. One the other hand, if all three literal nodes have mapping depth equal to 3, there will be at least $2K + 1$ nodes of mapping depth 2 in the fanin cone of node C_j. Three intermediate nodes of mapping depth 3 must be introduced during the decomposition of node C_j. A K-feasible cut of height 3 for node C_j becomes impossible. So $MMD_{D(N(C_j))} \cdot (C_j) = 5$. This proves Proposition 3. □

We show the correspondence betwen the truth assignment of variables in an instance F of 3SAT and the decomposition of $N(x_i)$, $1 \le i \le n$, as follows: variable $x_i = 1$ in 3SAT if and only if $MMD_{D(N(s_i))}(x_i) = 2$, and variable $x_i = 0$ if and only if $MMD_{D(N(s_i))}(\bar{x}_i) = 2$. We now prove Lemma 5 by this correspondence.

PROOF OF LEMMA 5. (*If*). It is obvious $MMD(D(N(F))) = 4$ if and only if $MMD_{D(N(C_j))}(C_j)$ for $1 \le j \le m$. This implies at least one literal node (and its variable node) in each subnetwork $N(C_j)$ has a mapping depth of 2 (Proposition 3). According to the correspondence, we can obtain a truth assignment of variables from the literal nodes of mapping depth 2. Since

each $N(C_j)$ contains at least one literal node of mapping depth 2, each clause C_j contains a literal of value 1. Hence F is satisfied by the truth assignment.

(*Only if*). Assume F is satisfiable. Then there is a truth assignment of variables that allows at least one literal in each clause to have a a value of 1. According to the correspondence, this truth assignment of variables specifies how each subnetwork $N(x_i)(1 \leq i \leq n)$ is decomposed and determines the mapping depth of nodes x_i and $\bar{x}_i$ in each $N(x_i)$. Literal nodes have the same mapping depth as their variable nodes. Since each clause C_j contains a literal of true value, each $N(C_j)$ contains a literal node of mapping depth 2. Hence $MMD(D(N(F))) = 4$. This proves Lemma 5. □

PROPOSITION 4. *Let $N_K(C_j)$ be a decomposed subnetwork of $N_K(C_j)$. Then $MMD_{D(N_K(C_j))}(C_j) = 3$ if and only if $MMD_{D(N_K(C_j))}(l_j^k) = 2$ for some $1 \leq k \leq 3$.*

PROOF 10. When the subnetwork $N_K(C_j)$ contains at least one literal node of mapping depth 2, together with nodes $q_j^1, \ldots, q_j^{K-5}$, a K-feasible cut of height 2 exists for node C_j. Hence node C_j has a mapping depth equal to 3. Otherwise, node C_j has a mapping depth of 4. □

PROOF OF LEMMA 6. Similar to the proof of Lemma 5, and is omitted. □

ACKNOWLEDGMENTS

The authors are very grateful to C. Legl in Antreich's group at the Institute of Electronic Design Automation, Technical University of Munich, Germany, for providing us with the TOS logic synthesis package.

REFERENCES

ASHENHURST, R. L. 1959. The decomposition of switching functions. *Ann. Comput. Lab. Harvard Univ. 29*, 74–116.

CHEN, K. C., CONG, J., DING, Y., AND KAHNG, A. B. 1992. Dag-map: Graph-based fpga technology mapping for delay optimization. *IEEE Des. Test*, 7–20.

CONG, J. AND DING, Y. 1993. Beyond the combinatorial limit in depth minimization for LUT-based FPGA designs. In *Proceedings of the International Conference on Computer-Aided Design* (ICCAD '93, Santa Clara, CA, Nov. 7–11), M. Lightner and J. A. G. Jess, Eds. IEEE Computer Society Press, Los Alamitos, CA, 110–114.

CONG, J. AND DING, Y. 1994a. Flowmap: An optimal technology mapping algorithm for delay optimization in lookup-table based fpga designs. *IEEE Trans. Comput.-Aided Des.* (Jan. 1994), 1–12.

CONG, J. AND DING, Y. 1994b. On area/depth trade-off in lut-based fpga technology mapping. *IEEE Trans. Very Large Scale Integr. Syst. 2*.

CONG, J. AND DING, Y. 1994c. On nominal delay minimization in LUT-based FPGA technology mapping. *Integr. VLSI J. 18*, 1 (Dec. 1994), 73–94.

CONG, J. AND DING, Y. 1996. Combinational logic synthesis for LUT based field programmable gate arrays. *ACM Trans. Des. Autom. Electron. Syst. 1*, 2, 145–204.

CONG, J. AND HWANG, Y.-Y. 1995. Simultaneous depth and area minimization in LUT-based FPGA mapping. In *Proceedings of the Third International ACM Symposium on Field-Programmable Gate Arrays* (FPGA '95, Monterey, CA, Feb. 12–14), P. K. Chan and J. Rose, Eds. ACM Press, New York, NY, 68–74.

CONG, J., PECK, J., AND DING, Y. 1996. RASP: A general logic synthesis system for SRAM-based FPGAs. In *Proceedings of the 1996 ACM Fourth International Symposium on*

Field-Programmable Gate Arrays (FPGA '96, Monterey, CA, Feb. 11–13), J. Rose and C. Ebeling, Eds. ACM Press, New York, NY, 137–143.

CURTIS, H. A. 1961. A generalized tree circuit. *J. ACM 8*, 4, 484–496.

DE MICHELI, G. 1994. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, Inc., New York, NY.

FRANCIS, R., ROSE, J., AND VRANESIC, Z. 1991a. Chortle-crf: Fast technology mapping for lookup table-based FPGAs. In *Proceedings of the 28th ACM/IEEE Conference on Design Automation* (DAC '91, San Francisco, CA, June 17–21), A. R. Newton, Ed. ACM Press, New York, NY, 227–233.

FRANCIS, R. J., ROSE, J., AND VRANESIC, Z. 1991b. Technology mapping of lookup table-based fpgas for performance. In *Proceedings of the IEEE International Conference on Computer-Aided Design*, IEEE Computer Society Press, Los Alamitos, CA, 568–571.

GAREY, M. AND JOHNSON, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY.

HOROWITZ, E. AND SAHNI, S. 1978. *Fundamentals of Computer Algorithms*. Computer Science Press, Inc., New York, NY.

HUANG, J.-D., JOU, J.-Y., AND SHEN, W.-Z. 1996. An iterative area/performance trade-off algorithm for LUT-based FPGA technology mapping. In *Proceedings of the 1996 IEEE/ACM International Conference on Computer-Aided Design* (ICCAD '96, San Jose, CA, Nov. 10–14), R. A. Rutenbar and R. H. J. M. Otten, Eds. IEEE Computer Society Press, Los Alamitos, CA, 13–17.

LAI, Y.-T., PAN, K.-R. R., AND PEDRAM, M. 1994. FPGA synthesis using function decomposition. In *Proceedings of the IEEE International Conference on Computer Design* (Cambridge, MA, Oct. 10-12), IEEE Computer Society Press, Los Alamitos, CA, 30–35.

LEGL, C., ECKL, K., AND WURTH, B. 1996a. Performance-directed technology mapping for lut-based fpgas - what role do decomposition and covering play? In *Proceedings of the International Workshop on Field Programmable Logic and Applications*.

LEGL, C., WURTH, B., AND ECKL, K. 1996b. A Boolean approach to performance-directed technology mapping for LUT-based FPGA designs. In *Proceedings of the 33rd Annual Conference on Design Automation* (DAC '96, Las Vegas, NV, June 3–7), T. P. Pennino and E. J. Yoffa, Eds. ACM Press, New York, NY, 730–733.

MURGAI, R., SHENOY, N., BRAYTON, R. K., AND SANGIOVANNI-VINCENTELLI, A. 1991. Performance directed synthesis for table look up programmable gate arrays. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design* (ICCAD '91, Santa Clara, CA, Nov. 11-14), IEEE Computer Society Press, Los Alamitos, CA, 572–575.

ROTH, J. P. AND KARP, R. M. 1962. Minimization over boolean graphs. *IBM J. Res. Dev.*, 227–238.

RUDELL, R. 1989. Logic synthesis for VLSI design. Computer Science Department, University of California at Berkeley, Berkeley, CA.

SAWKAR, P. AND THOMAS, D. 1993. Performance directed technology mapping for look-up table based FPGAs. In *Proceedings of the 30th International Conference on Design Automation* (DAC '93, Dallas, TX, June 14–18), A. E. Dunlop, Ed. ACM Press, New York, NY, 208–212.

SENTOVICH, E., SINGH, K., LAVAGNO, L., MOON, C., MURGAI, R., SALDANHA, A., SAVOJ, H., STEPHAN, P., BRAYTON, R., AND SANGIOVANNI-VINCENTELLI, A. 1992. SIS: A system for sequential circuit synthesis. Tech. Rep. UCB/ERL M92/41. UC Berkeley, Berkeley, CA.

WANG, A. R. R. 1991. Algorithms for multilevel logic optimization. Ph.D. Dissertation. Computer Science Department, University of California at Berkeley, Berkeley, CA.

WURTH, B., ECKL, K., AND ANTREICH, K. 1995. Functional multiple-output decomposition: Theory and an implicit algorithm. In *Proceedings of the 32nd ACM/IEEE Conference on Design Automation* (DAC '95, San Francisco, CA, June 12–16), B. T. Preas, Ed. ACM Press, New York, NY, 54–59.

YANG, H. AND WONG, D. F. 1994. Edge-map: Optimal performance driven technology mapping for iterative LUT based FPGA designs. In *Proceedings of the 1994 IEEE/ACM International Conference on Computer-Aided Design* (ICCAD '94, San Jose, CA, Nov. 6–10), J. A. G. Jess and R. Rudell, Eds. IEEE Computer Society Press, Los Alamitos, CA, 150–155.

Received: September 1997; revised: February 1998; accepted: September 1998