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

Jason Cong and Yean-Yow Hwang

Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90024

Abstract

In this paper, we study the problem of decomposing gates in fanin-unbounded or K-bounded networks such that the mapping solutions computed by a depth-optimal mapper have minimum depth. We present several theoretical results: (1) any further decomposition of a K-bounded network will lead to an optimal mapping depth smaller than or equal to that of the original network, regardless of the decomposition algorithm used, and (2) the problem of gate decomposition for depth-optimal technology mapping is NP-hard for fanin-unbounded networks when $K \geq 3$ and remains NP-hard for K-bounded networks when $K \geq 5$. We propose a novel gate decomposition algorithm, named DOGMA, which combines level-driven node packing technique (Chortle-d) and the network flow based optimal labeling technique (FlowMap) for depth-optimal technology mapping. Experimental results show that the networks produced by DOGMA allow depth-optimal technology mappers to improve the mapping solutions by up to 11% in depth and up to 35% in area comparing to the mapping results of networks decomposed by other existing decomposition algorithms.

1. Introduction

The field programmable gate array (FPGA) is a popular technology for VLSI ASIC design and system prototyping due to its short design cycle and low manufacturing cost. The lookup-table (LUT) based FPGAs have been widely used. A K-input LUT (K-LUT) consists of 2^K SRAM cell to store a truth table which can implement an arbitrary function of up to K inputs. The goal of LUT-based FPGA technology mapping is to cover a given network using K-LUTs such that either area or delay is minimized or routability is maximized in the final LUT network. The delay of a network can be estimated by the number of levels (i.e. depth) under the unit delay model before placement and routing are performed. Several LUT-mapping algorithms have been proposed for depth minimization [FrRV91c, MuSB91a, SaTh93, ChCD92, CoDi94a] and reported good results. In particular, the FlowMap algorithm [CoDi94a] guarantees a depth-optimal mapping solution for any K-bounded network. However, it can not be applied directly to a fanin-unbounded network and depth-optimal result may not hold when *further* gate decomposition of a K-bounded network is allowed prior to the mapping.

Gate decomposition of the initial network can considerably impact the depth of mapping solutions. For example, assume $K = 3$. The network N shown in Figure 1(a) is not K-bounded since node v has 5 fanins. Assuming that the nodes u_1, u_2 and u_3 have two inputs each, if v is decomposed in such a way that PI_1 and PI_2 are merged into one node, as shown in Figure 1(b), there is no way to obtain a mapping solution of depth less than 3. However, if a different decomposition is carried out for v, as shown in Figure 1(c), a mapping solution of depth equal to 2 is obtained. Even for a K-bounded network, the depth of the optimal mapping solution may decrease if its gates are further decomposed before technology mapping. For example, the network N shown in Figure 2(a) is K-bounded for $K = 3$. The minimum depth in any mapping solution of N is 3. But if the network is further decomposed into Fig. 2(b), a mapping solution with depth 2 can be obtained. While FlowMap guarantees depth-optimal mapping solutions for K-bounded networks, it does not exploit the freedom introduced by further gate decomposition. Nevertheless, we would like to emphasize that the optimality result by FlowMap for K-bounded networks enables the study and evaluation of various gate decomposition algorithms and their impact on technology mapping. Otherwise, when a gate decomposition algorithm leads to a poor mapping solution, it would be difficult to determine whether the gate decomposition algorithm, or the mapping heuristic, or a combination of both result in the poor mapping result. To our knowledge, this work is the first in-depth study and evaluation of the impact of gate decomposition on the depth of the *final* mapping solutions, based on the depth-optimal mapping algorithm for K-bounded networks, which was available only recently.

Gate decomposition methods can be classified as (1) structural decomposition and (2) Boolean decomposition. In structural decomposition, a multi-fanin gate is replaced by a fanin tree. Structural decomposition can be applied only to networks with gates which are associative with respect to their fanins, such as networks of simple gates (e.g. AND, OR, NAND, NOR, XOR). Figure 1 and Figure 2 show examples of this type. On the other hand, Boolean decomposition decomposes a complex gate according to its functionality and representation. Examples of Boolean decomposition includes AND-OR decomposition [BrRS87], co-factoring [SaTh92], cube-packing [MuSB91a], and functional decomposition [LaPP94, WuEA95]. This paper focuses on optimal structural decomposition for

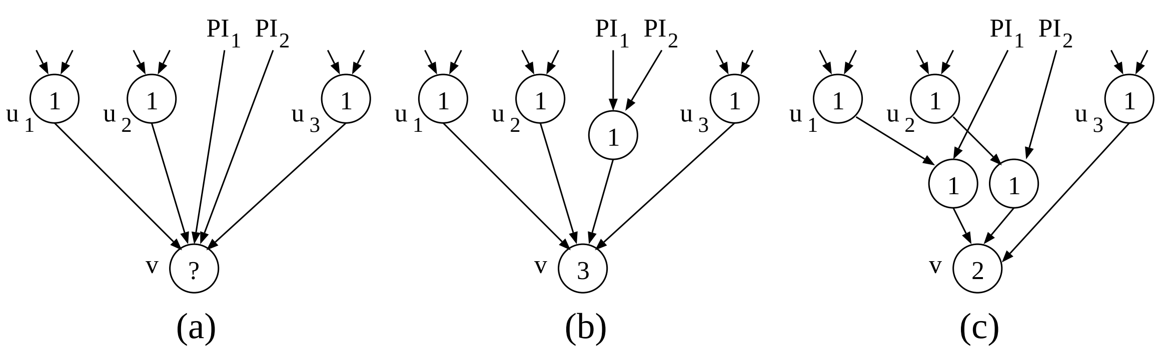
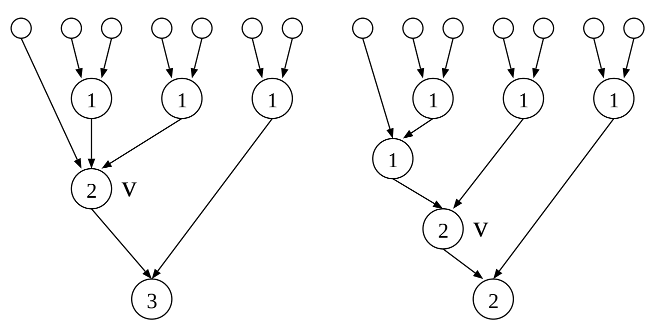


Figure 1 Gate decomposition of an unbounded networks for K-LUT mapping ($K = 3$).
(a) initial network, (b) decomposition yielding mapping depth = 3,
(c) decomposition yielding mapping depth = 2.

depth minimization in LUT mapping.

Several gate decomposition routines have been used for LUT-mapping. The *tech_decomp* routine in SIS [SeSL92] produces a height-balanced fanin tree for each decomposed gate to minimize the height of the fanin tree. The *dmig* routine [Wa89, ChCD92] uses a Huffman coding tree to minimize the level of the decomposed node in the resulting network. It guarantees a minimum level in the decomposed network but does not guarantee the optimal depth in the final LUT network. In fact, Figure 1(b) is the outcome of the *dmig* routine which is sub-optimal for depth-optimal mapping. The SIS routine *speed_up* is often used to obtain delay-optimized *2-input* networks for technology mapping. It uses Boolean decomposition based on repeated node collapsing and decomposition to speed-up the critical path. MIS-pga-delay [MuSB91a] and TechMap-D [SaTh93] used *speed_up* as well as other logic synthesis techniques to obtain delay-minimized mapping solutions. These decomposition methods, although effective to some extent, are not designed to minimize the depth of LUT networks *directly*. Chortle-d [FrRV91c] is the first algorithm which computes a depth-optimal gate decomposition and mapping solution based on a bin packing approach when the input network is a tree (may be unbounded). For general networks, it considers packing nodes on reconvergent paths into either the same or different bins for all pairs of reconvergent paths. Because Chortle-d takes a greedy approach in generating the LUTs to cover nodes, its gate decomposition and mapping solutions for general networks are suboptimal in term of depth. In this paper, we study the gate decomposition problem and algorithms for depth-optimal technology mappers so that minimum depth are obtained in the final mapping solutions.

The remainder of this paper is organized as follows. Section 2 defines basic terminology, presents properties of structural gate decomposition and gives problem formulation. Section 3 shows the complexity of the problem of gate decomposition for optimal depth in LUT-based technology mapping. A novel algorithm of gate decomposition for depth-optimal mapping, named DOGMA, is



(a) (b)

Figure 2 Further gate decomposition of a K-bounded network ($K = 3$).
(a) Initial K-bounded network. (b) After further gate decomposition.
(The label in each node is the minimum depth of the node in any mapping solution.)

presented in Section 4 based on the min-height K-feasible bin packing formulation. Experimental results are presented in Section 5 and Section 6 concludes the paper. Since we use the concepts and techniques in [CoDi94a] extensively, it is beneficial for the reader to be familiar with the results in [CoDi94a].

2. Problem Formulation

2.1. Notations and Preliminaries

A combinational Boolean network can be represented by a directed acyclic graph (DAG) where a node represents a logic gate and a directed edge (u,v) represents a connection from the output of node u to the input of node v. A primary input (PI) node is a node of in-degree zero and a primary output (PO) node is a node with no outgoing edge. Other nodes are internal nodes. The depth of a node v is the number of edges on the longest path from any PI to v. A PI node has a depth of zero. The depth of a network is the largest node depth among POs. Let $input(v)$ be the set of fanin nodes of node v. Let K be the input size of an LUT. A network is K-bounded if every node v in the network satisfies $|input(v)| \leq K$. Otherwise, it is an unbounded network. Given a subgraph H of a network, we use $input(H)$ to denote the set of distinct nodes outside H which supply inputs to nodes in H.

Given a node v in network N, let N_v denote the subnetwork consisting of node v and all the predecessors of v. The *minimum mapping depth* of v, denoted $MMD_N(v)$, is defined as the minimum depth among all possible K-LUT mapping solutions of N_v. Note that $MMD_N(v)$ is the same as the $label(v)$ computed by the FlowMap algorithm. PI nodes have a mapping depth of 0. The minimum mapping depth of a network N, denoted $MMD(N)$, is the maximum mapping depth among all PO nodes. Given a K-bounded network N, the FlowMap [CoDi94a] algorithm computes $MMD_N(v)$ for every node $v \in N$ in polynomial time. The computation is based on the max-flow min-cut computation. A cut in N_v is a partition $(X_v, \bar{X}_v)$ of nodes in N_v such that PI nodes are in X_v and $v \in \bar{X}_v$. The cutset of a cut, denoted $n(X_v, \bar{X}_v)$, is defined as $input(\bar{X}_v)$. A cut is K-feasible if $|n(X_v, \bar{X}_v)| \leq K$. The height of a cut, denoted $height(X_v, \bar{X}_v)$, is the maximum mapping depth for nodes in $n(X_v, \bar{X}_v)$. We have the following lemmas based on results in [CoDi94a].

Lemma 1 A node v has $MMD_N(v) = p$ if there is a K-feasible cut of height of $p-1$ in N_v but no K-feasible cut of height of $p-2$ or smaller exists.

Lemma 2 If $u \in input(v)$ then $MMD_N(u) \leq MMD_N(v)$.

2.2. Properties of Structural Decomposition

Let N be a network and node $v \in N$ satisfies $|input(v)| > 2$. Given a decomposition algorithm D, we define a *decomposition step* of D at v, denoted D_v, as follows: Decomposition step D_v (i) chooses two nodes u_1 and u_2 from $input(v)$, (ii) removes edges (u_1, v) and (u_2, v), (iii) introduces a new node w and new edges (u_1, w), (u_2, w), (w, v) and adds them to N. The resulting network is denoted as $D_v(N)$. For example, Figure 3(a) shows a node v with fanin nodes u_1, u_2, u_3, u_4, u_5. Figure 3(b) is the network after one decomposition step at v. Since gates are associative, the introduced node w have the same gate type as v.

We present two properties of the structural decomposition. (1) *Any* decomposition method will not increase the minimum mapping depth of a network. It can only decrease the depth. This phenomena has been observed in [CoDi94a] without an explanation. (2) Under certain circumstance, further decomposition of a gate will not decrease the minimum mapping depth. Under this circumstance, it is irrelevant which decomposition algorithm is used for those gates. The circumstance occurs in the packing-based decomposition methods. These two properties are given in Lemma 3 and Lemma 4 respectively.

Lemma 3 Given a K-bounded network N, any decomposition algorithm D, and any node $v \in N$, $MMD(D_v(N)) \leq MMD(N)$.

Proof Because the decomposition step D_v does not remove nodes, every node in N is still in $D_v(N)$. Let $u \in N$ be an arbitrary node and $u' \in D_v(N)$ be the node corresponding to u in the new network. Every K-feasible cut for u in N is still a K-feasible cut for u' in $D_v(N)$ (but not vice versa as the new node w introduced by D_v may be in some K-feasible cut in $D_v(N)$, but w does not exist in N.) Hence $MMD_{D_v(N)}(u') \leq MMD_N(u)$. Since u is arbitrary, $MMD(D_v(N)) \leq MMD(N)$. □

Lemma 4 If $MMD_N(u) = MMD_N(v)$ for all $u \in input(v)$ in a K-bounded network N, then $MMD(N) = MMD(D_v(N))$ for any decomposition algorithm D.

Proof Let $N' = D_v(N)$. Let w be the intermediate node introduced in D_v and $u \in input(w)$. According to Lemma 2, $MMD_{N'}(u) \leq MMD_{N'}(w) \leq MMD_{N'}(v)$. Since $MMD_N(u) = MMD_N(v) \geq MMD_{N'}(v)$ (Lemma 3) and $MMD_N(u) = MMD_{N'}(u)$ (as $N_u = N'_u$), we have $MMD_{N'}(u) = MMD_{N'}(w) = MMD_{N'}(v)$. We now use mathematical induction according to topological

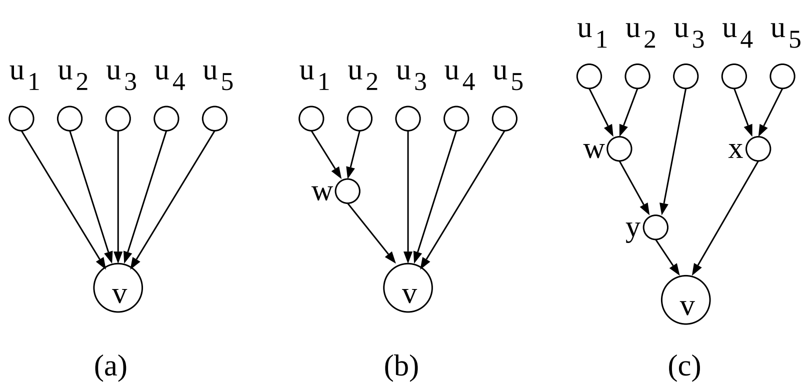


Figure 3 Decomposition of node v. (a) Before decomposition,
(b) $D_v(N)$ after one decomposition step of D_v, (c) after a sequence of decomposition steps.

ordering from PIs in N' to show that for any node $t \in N'$, $MMD_{N'}(t) = MMD_N(t)$. Base: when node t is a PI, it is obviously true. Induction hypothesis: Assume that $MMD_{N'}(t') = MMD_N(t')$ for any $t' \in N'_t$ before t. For the min-height K-feasible cut $(X_t, \bar{X}_t)$ in N'_t, (i) if w is not in the cutset $n(X_t, \bar{X}_t)$, the cut also exists in N_t. According to the hypothesis, every node in the cutset has the same depth in N and N'. As a result $MMD_{N'}(t) = MMD_N(t)$; (ii) if w is in the cutset $n(X_t, \bar{X}_t)$, since the decomposition tree of v is a fanout-free cone, we can replace w by v to form a new cut of equal or smaller size in N'_t. Since $MMD_{N'}(w) = MMD_{N'}(v)$, the new cut is of the same height as $(X_t, \bar{X}_t)$, which also corresponds to a K-feasible cut in N of the same height. Therefore, $MMD_{N'}(t) = MMD_N(t)$. Since the result holds for any $t \in N' = D_v(N)$, we have $MMD(N) = MMD(D_v(N))$ for any decomposition algorithm D. □

2.3. Problem Definition

According to Lemma 3, the further a network is decomposed, the smaller the mapping depth might be. Therefore, we decompose every node v such that its fanins form a binary tree in the decomposed network. For example, Figure 3(c) is a complete decomposition of node v. Every decomposition step introduces one intermediate node and $|input(v)| - 2$ steps are required to decompose v. A network is fully decomposed if every multi-fanin node in the network is decomposed into a binary tree. In this paper, we study the problem of decomposing an unbounded or K-bounded network into 2-input networks such that the minimum mapping depth can be obtained by a depth-optimal technology mapper such as FlowMap. We formulate the following problems.

Structural Gate Decomposition for K-LUT Mapping (SGD/K) Given a simple-gate unbounded network N_∞, decompose N_∞ into a 2-input network N_2 such that for any other 2-input network decomposition N'_2 of N, $MMD(N_2) \le MMD(N'_2)$.

Structural Gate Decomposition for K-LUT Mapping of K-bounded Network (K-SGD/K) Given a simple-gate K-bounded network N_K, decompose N_K into a 2-input network N_2 such that for any other 2-input network decomposition N'_2 of N, $MMD(N_2) \le MMD(N'_2)$.

There are two issues related to the problem formulation of gate decomposition.

(1) **Independent Versus Simultaneous Gate Decomposition:** A smaller depth might be obtained when several gates are decomposed together instead of independently. For example, assume both u and v are AND-type gates in Figure 4(a) and $K = 3$. We have $MMD_N(u) = MMD_N(v) = 2$. When u and v are decomposed independently to obtain N_1 in Figure 4(b), $MMD_{N_1}(x) \ge 3$ because the min-cut of height 1 for x has at least 4 nodes and therefore is not 3-feasible. However, if u and v are decomposed together to obtain N_2 in Figure 4(c), we have $MMD_{N_2}(x) = 2$. Simultaneous gate decomposition, however, must take into account the node functionality (e.g. requires nodes u and v are of the same or compatible gate types), and is no longer a pure structural gate decomposition. We study only independent gate decomposition in this paper.

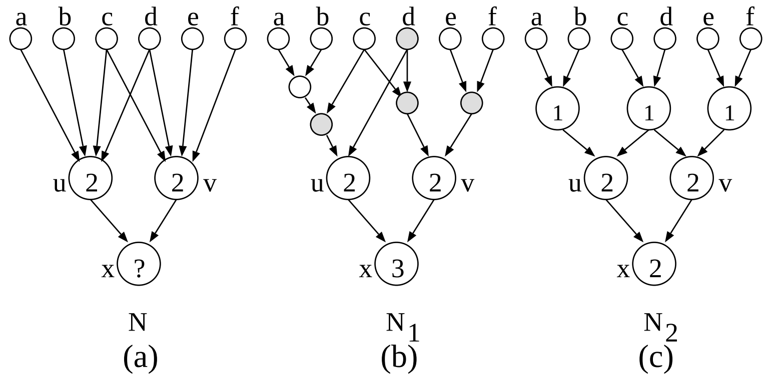


Figure 4 Computing $MMD_N(x)$ when $K = 3$.(a) original network N,
(b) N_1 obtained by independent gate decomposition,
(c) N_2 obtained by simultaneous gate decomposition.

(2) **Decomposition Before Mapping Versus Decomposition During Mapping:** Gate decomposition can be performed before the mapping phase in a *2-step* approach or embedded into the mapping process being part of an *integrated* approach. For example, the *dmig* algorithm was used in a 2-step approach followed by DAG-MAP or FlowMap while Chortle-d is an integrated approach (since it decomposes gates and generates LUTs in an interleaving manner). It may appears that the integrated approach can produce better mapping depth because it can choose any decomposition techniques during technology mapping. However, this is not the case. For an input network N, assume that the integrated approach produces a depth-optimal mapping solution M_N. Then M_N also induces a decomposed K-bounded network N'. A depth-optimal mapper for K-bounded networks (such as 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 $depth(M_{N'}) \le depth(M_N)$. However, since M_N is optimal with respect to N, we have $depth(M_N) \le depth(M_{N'})$. Therefore, $depth(M_N) = depth(M_{N'})$. This implies the best integrated approach and the best 2-step approach result in the same optimal depth. In this paper, we take the 2-step approach, i.e., gates are decomposed before a depth-optimal mapping algorithm (such as FlowMap [CoDi94a] or its enhancement CutMap [CoHw95]) is applied to obtain an LUT mapping solution. Nevertheless, our gate decomposition algorithm considers the depth optimization techniques used in FlowMap and the effect of gate decomposition in the mapping phase of FlowMap in order to obtain the decomposed network which is most suitable for FlowMap to achieve optimal mapping depth.

3. Complexity of SGD and K-SGD Problems

In this section we shall show the following results of the SGD/K and the K-SGD/K problems: (1) The SGD/K problem is NP-hard for $K \ge 3$; and (2) The K-SGD/K problem is NP-hard for $K \ge 5$. We only give the construction of the NP-Complete reduction in this section. Complete proofs of the results can be found in the Appendix. For readers not interested in the NP-Complete reduction, they may proceed directly to Section 4 to follow the novel gate decomposition algorithms proposed in this paper for depth-optimal mapping.

Our proof is based on polynomial time transformations from the 3SAT problem to the decision version of the SGD/K problem and the K-SGD/K problem. We first define the 3SAT problem, which is a well-known NP-Complete problem [GaJo79].

Problem: 3-Satisfiability (3SAT)
Instance: A set of Boolean variables $X = \{x_1, x_2, \ldots, x_n\}$ and collection of m clauses $C = \{C_1, C_2, ..., C_m\}$, 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 $\overline{x}_i$ for any variable x_i.
Question: Is there a truth assignment of the variables in X such that $C_j = 1$ for $1 \le j \le m$?

We shall transform each instance of 3SAT to an instance of SGD in polynomial time. The basic idea is to relate the decision of the truth assignment in 3SAT to the decision of node decomposition in SGD/K. Since determining the truth assignment is difficult, the transformation shows that determining the decomposition is also difficult. We define the decision version of the SGD/K problem.

Problem: Simple-Gate Decomposition(SGD/K-D)
Instance: A constant $K \ge 2$, a simple-gate unbounded network N_∞ and B.
Question: Is there a way to decompose N_∞ into a 2-input 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 the 3SAT with n variables $x_1, x_2, \cdots, x_n$ and m clauses $C_i, C_2, \cdots, C_m$, we construct an 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 as x_i and $\overline{x}_i$; (b) $(2K^2 - 3K)$ PI nodes in which two of them are denoted as PI_i^1 and PI_i^2; (c) $(2K-2)$ internal nodes, denoted as $v_i^1, \cdots, v_i^{K-3}$, $u_i^1, \cdots, u_i^{K-2}$, w_i^1, w_i^2 and $s_i$respectively; The nodes are connected as shown in Figure 5. Nodes w_i^1 and w_i^2 each have $K-1$ 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 a well-defined subnetwork for $K \ge 3$ and is K-bounded for $K \ge 4$.

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 as l_j^1, l_j^2, l_j^3; (c) $(2K-5)$ internal nodes $q_j^1, \cdots, q_j^{2K-5}$ each of them is the root of a complete 2-level K-ary tree with PI nodes as leaves; (d) $(K-2)$ internal nodes $r_j^1, \cdots, r_j^{K-2}$ each of them is the root of a complete 3-level K-ary tree with PI nodes as leaves. The connection is 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 a well-defined subnetwork for $K \ge 3$. However, the output node C_j is not K-bounded.

Finally, we connect the subnetworks $N(C_j)$, $j = 1, 2, \cdots, m$, with the subnetworks $N(x_i)$, $i = 1, 2, \cdots, n$, to form the network $N(F)$. Let l_j^k $(1 \le k \le 3)$ be a literal in C_j. If for some variable x_i

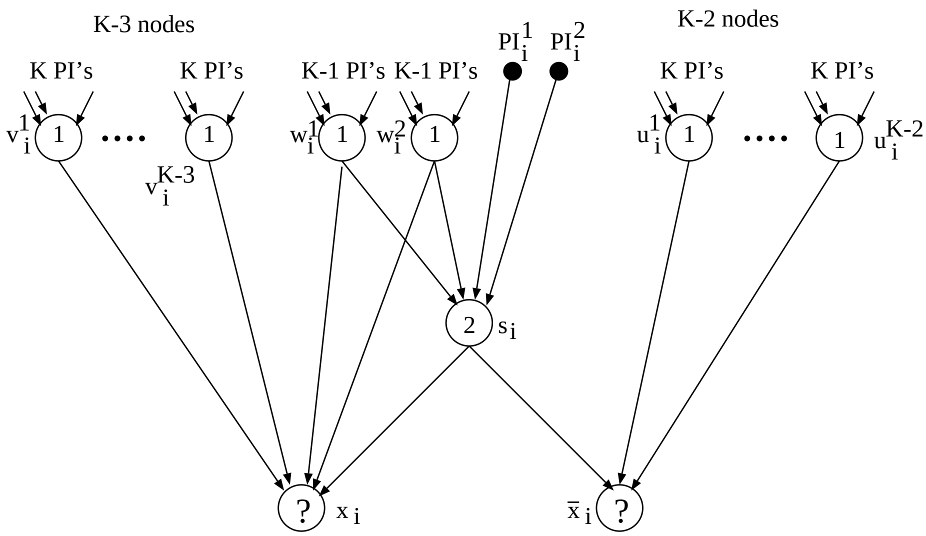
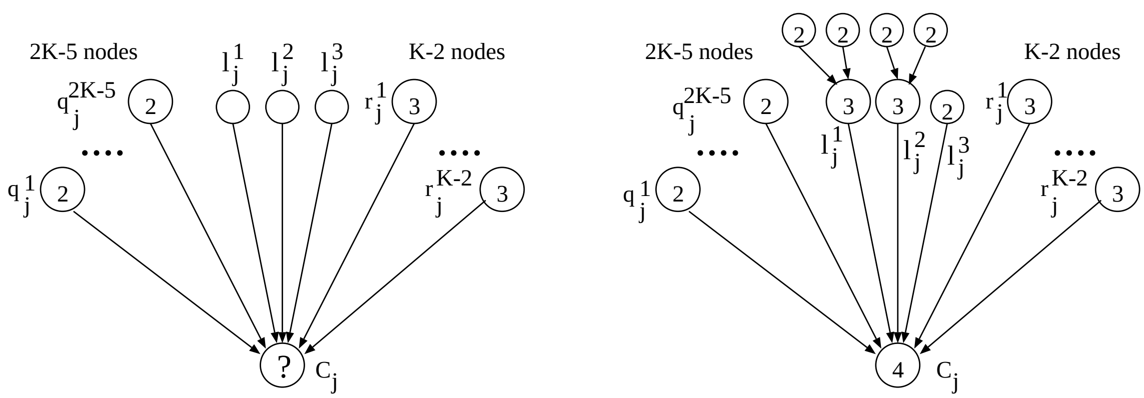


Figure 5 Construction of network $N(x_i)$ for each Boolean variable x_i.



(a) (b)

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

$l_j^k = x_i$, we connect the output node x_i of subnetwork $N(x_i)$ as the fanin to the internal node l_j^k of subnetwork $N(C_j)$. Similarly, if $l_j^k = \bar{x}_i$, we connect node $\bar{x}_i$ of $N(x_i)$ as the fanin to node l_j^k of $N(C_j)$. We call x_i the variable node of l_j^k. $N(F)$ will have m output nodes $C_1, \cdots, C_m$. 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. Every subnetwork $N(x_i)$ provides two output signals from nodes x_i and $\bar{x}_i$. Every subnetwork $N(C_j)$ requires three input signals l_j^1, l_j^2, l_j^3. Because the first clause C_1 in F is $(x_1 + \bar{x}_2 + x_3)$, we connect nodes x_1 to l_1^1, $\bar{x}_2$ to l_1^2 and x_3 to l_1^3, respectively. We have the following lemma and theorem. The proofs are detailed in the Appendix.

Lemma 5 The 3SAT instance F is satisfiable if and only if there is a decomposition $D(N(F))$ of $N(F)$ such that $MMD(D(N(F))) = 4$.

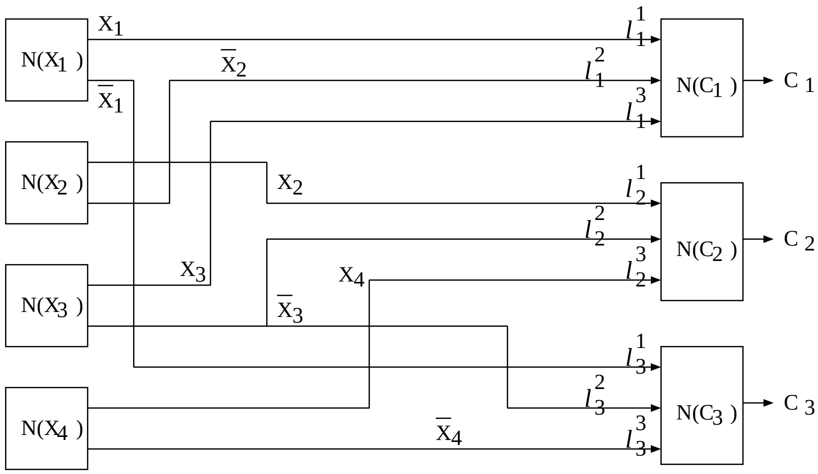


Figure 7 The network $N(F)$ for $F = (x_1 + \bar{x}_2 + x_3)(x_2 + \bar{x}_3 + x_4)(\bar{x}_1 + \bar{x}_3 + x_4)$.

Theorem 1 The SGD/K problem is NP-hard for $K \geq 3$.

We now show the complexity of the K-SGD/K problem. In Figure 5, the network $N(x_i)$ is K-bounded for $K \geq 4$. Hence the constructions of $N(x_i)$ remains the same. In Figure 6(a), the network $N(C_j)$ is unbounded. Consider an alternative construction $N_K(C_j)$ which consists of the following nodes: (a) One output node denoted C_j; (b) Three literal nodes denoted as l_j^1, l_j^2, l_j^3; (c) $(K-5)$ internal nodes $q_j^1, \cdots, q_j^{K-5}$ each of them is the root of a complete 2-level K-ary tree with PI nodes as leaves. The connection is shown in Figure 8(a). In this construction, $N_K(C_j)$ is a K-bounded subnetwork for $K \geq 5$. We connect $N(x_i)$ and $N_K(C_j)$ as before and denote the resulting network as $N_K(F)$. We have the following lemma and theorem. The proofs are detailed in the Appendix.

Lemma 6 The 3SAT instance F is satisfiable if and only if there is a decomposition $D(N_K(F))$ of $N_K(F)$ such that $MMD(D(N_K(F))) = 3$.

Theorem 2 The K-SGD/K problem is NP-hard for $K \geq 5$.

4. Gate Decomposition Algorithm for Depth-Optimal LUT Mapping

4.1. Algorithm Overview

Intuitively speaking, our decomposition algorithm, named DOGMA (Depth-Optimal Gate decomposition for MApping), combines the level-driven node packing technique in Chortle-d and the network flow based optimal labeling technique used in FlowMap. Given an unbounded network N, nodes are visited from PIs to POs in a topological order. For each node v, we decompose v (the decomposed network is denoted as $N(v)$) and label v with the minimum mapping depth $MMD_{N(v)}(v)$. When v is visited, its fanins have already been decomposed and labeled accordingly. We partition the nodes in $input(v)$ into groups such that each group consists of nodes with the same label (i.e.

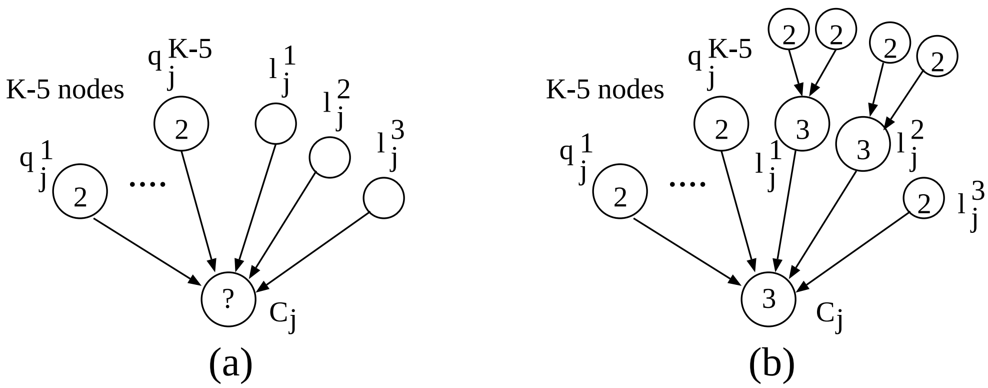


Figure 8 Construction of K-bounded subnetwork $N_K(C_j)$ for each clause C_j.

minimum mapping depth). We process these groups in an ascending order with respect to their labels. For a group of nodes labeled p, we pack the nodes into a minimum number of bins such that there exists a K-feasible cut of height $p-1$ for the nodes in each bin. Such a bin is called a *min-height K-feasible bin*. A node u_i is created for each bin B_i with fanins from nodes in B_i and a fanout to v. Node u_i has a mapping depth of p and will be given a label p. If more than 2 nodes are in the bin B_i, node u_i needs to be further decomposed to form a binary tree. Note that according to Lemma 4, no matter how u_i is decomposed, the minimum mapping depth of the network is always the same. We then proceed to the group of a next higher label $p+1$. For each bin node u_i created in the previous step, a single-fanin node w_i is created with fanin from u_i and is labeled $p+1$. These single-fanin nodes together with original nodes in the group (all of them are labeled $p+1$) are again packed into a minimal number of min-height K-feasible bins. This process continues until all nodes are packed into one bin which corresponds to the node v.

We illustrate the algorithm in Figure 9. Assume $K=3$. Figure 9(a) shows a network N in which every gate has been decomposed into a 2-fanin node except the output gate v. Every decomposed gate u has been given a label $MMD_N(u)$. Gate v has 5 fanins, among which nodes b,c,d are labeled 2 and nodes a,e are labeled 3. According to the algorithm, we first pack b,c,d into a minimal number of min-height K-feasible bins. Since a K-feasible cut of height 1 exists for b and c together, we put b,c into one bin and d into another bin, and introduce nodes f and g for each bin respectively. Note that f and g are labeled 2. Then we proceed to the group of label 3. Nodes h and i are created and given a label 3 with fanin from f and g respectively. Figure 9(b) shows the network at this point. Shaded nodes represent nodes created during the decomposition. We try to pack a,h,i,e into a minimal number of min-height K-feasible bins. By the min-cut computation, we find min-height K-feasible cuts of height 2 for $\{a,h\}$ and $\{i,e\}$ respectively. As a result we merge a,h into j and i,e into k respectively. We continue and create nodes l and m for j and k respectively, and pack l and m into a bin which corresponds to v. The decomposed network (denoted N') at this point is shown in Figure 9(c). Node v has a minimum mapping depth $MMD_{N'}(v)=4$ and is given a label 4. In the last step, single-fanin nodes *g*, *h*, *i*, *l* and *m* are removed. The final decomposed network has a minimum mapping depth of 4.

Our algorithm is similar to Chortle-d in that decomposition is done by packing nodes into a *minimal* number of bins according to their node labels (i.e. minimum mapping depth). However, there are two major differences. First, Chortle-d assigns LUTs during the decomposition while our algorithm simply computes minimum mapping depth for every node. Our approach is optimal in computing mapping depth (using FlowMap algorithm) while Chortle-d computes the node mapping depth based on the partially generated LUT network of the node. Since the fanin constraint is not a *monotone* clustering constraint [CoDi94a], Chortle-d might obtain inaccurate labels for nodes. Secondly, Chortle-d uses bin-packing heuristics together with enumerative consideration for nodes on reconvergent paths, while we use network flow based min-cut packing methods (to be described in details in Section 4.2) for packing nodes into min-height *K*-feasible bins. Sometime it is not obvious to decide locally whether two nodes can be packed into one min-height *K*-feasible bin unless a global min-cut computation is used. For example, nodes *e* and *i* can be packed into a min-height *K*-feasible bin but the Chortle-d algorithm may fail to do so. Of course, we obtain this packing quality at an expense of extra computation time. However, the running time is never a problem in practice in our experiments.

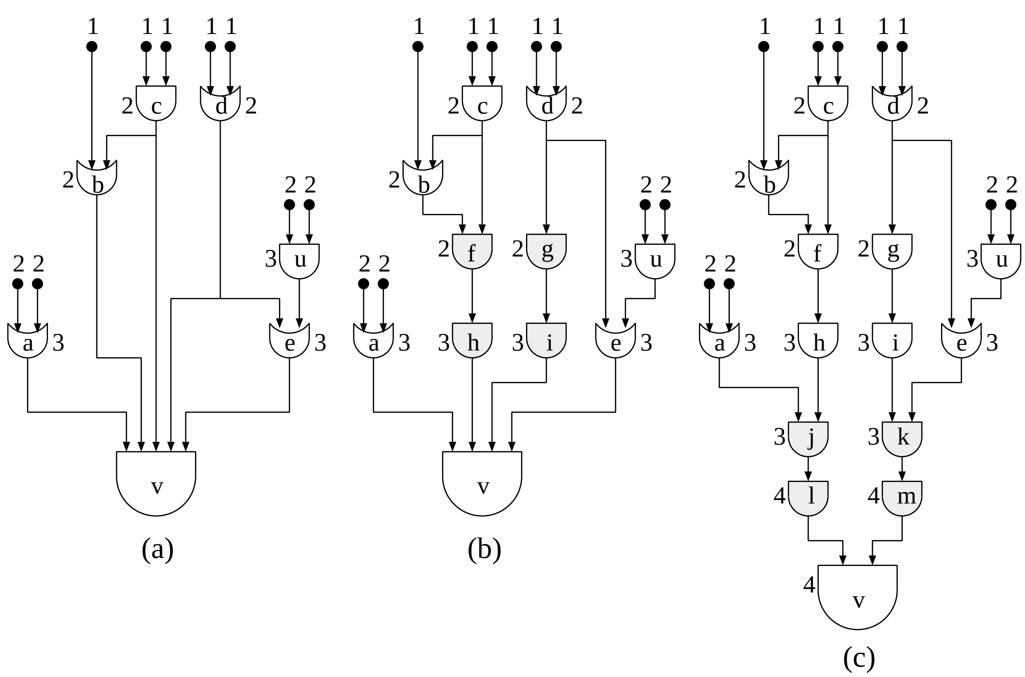


Figure 9 Decomposition of gate *v* by the DOGMA algorithm. (a) Before decomposition, (b) b,c,d are packed into f,g, (c) a,h,i,e are packed into j,k.

4.2. Network Flow Based Packing Methods

The central problem remains to be solved in our approach is the min-height K-feasible bin packing problem defined as follows. We shall give three heuristics in this section and one exact method in the next section to solve the problem.

Min-Height K-Feasible Bin Packing Problem Given a set A of nodes of minimum mapping depth p, partition A into a minimal number of bins such that there is a K-feasible cut of height $p-1$ for nodes in each bin. Such a bin is called a min-height K-feasible bin.

Our heuristics are based on the max-flow min-cut algorithm and bin-packing heuristics. The first-fit-decreasing (FFD) and best-fit-decreasing (BFD) are two heuristics for the bin packing problem [HoSa78]. Items are sorted into a decreasing order according to their size and then assigned to bins. FFD assigns each item in the sorted order to the first-fit bin while BFD assigns items to the the best-fit bin (i.e. the bin with the least residual capacity after assignment). The total size of items in any bin must not exceed the bin capacity. For the min-height K-feasible bin packing problem, we propose two min-cut based heuristics MC-FFD and MC-BFD, which are analogous to FFD and BFD except that the ‘‘total item size’’ of a group of nodes S with label p is now defined as the size of the min-cut of height $p-1$ for S. Nodes in S can be packed into one bin as long as the min-cut of height $p-1$ for these nodes is still K-feasible.

To determine if there is a K-feasible cut of height $p-1$ for a set of nodes in $S=\{v_1, v_2, \ldots, v_m\}$ with label p, we construct the flow network and apply the max-flow algorithm [CoDi94a] as follows. (1) We introduce a sink node t with fanins from every node in S. (2) We introduce a source node s connected to all PIs in the network rooted at t. (3) Every edge is given an infinite flow capacity. (4) Every node u_i in the flow network except s and t is replaced by two nodes q_i and r_i connected by a directed edge $e_i=(q_i, r_i)$ such that $input(q_i)=input(u_i)$ and $output(r_i)=output(u_i)$. If u_i is labeled p, e_i is assigned an infinite flow capacity. Otherwise, e_i is assigned a unit flow capacity. (5) Finally, we compute a max-flow to obtain a min-cut in the flow network, which can be shown to be a cut of height $p-1$ [CoDi94a]. By examining the size of this cut, we can determine if there is a K-feasible cut of height $p-1$ for nodes in S.

The third heuristic is the network flow based maximal-sharing-decreasing (MC-MSD) heuristic which emphasizes on sharing nodes among cuts. Assume both u and v are fanins of a node being decomposed and are labeled p. Let n_u, n_v and $n_{\{u,v\}}$ represent the size of min-cuts of height $p-1$ for node u, v and the nodes $\{u,v\}$, respectively. The sharing between u and v is defined as $share(u,v)=n_u+n_v-n_{\{u,v\}}$. When $share(u,v)>0$, the size of min-cut of height $p-1$ for u and v together is smaller than the sum of their individual min-cut size, which is beneficial to the packing. The MC-MSD heuristic is similar to the MC-BFD heuristic in that the *best-fit* bin is now chosen to be the bin that produces a maximal sharing.

4.3. The Optimal Min-Height K-Feasible Bin Packing Method

We compute the optimal min-height K-feasible bin packing by a dynamic programming approach which is similar to the one for the number partitioning problem. In the number partitioning problem we want to find a way to divide an integer set into two subsets of equal sum. This problem is NP-complete but can be solved in pseudo-polynomial time [GaJo79]. We first formulate a general number partitioning problem and an analogous node partitioning problem. Then we give an algorithm for the number partitioning problem. The method for the node partitioning problem is similar and hence skipped. Based on the solution, we give an algorithm which solves the min-height K-feasible bin packing problem optimally.

The k-Subset Number Partitioning Problem (k-SNP) Given an integer set A and a bin size b, is there a way to partition A into subsets (bins) $A_1, A_2, \ldots, A_k$ such that $\sum_{a_j \in A_i} a_j \le b$ for $1 \le i \le k$?

The Min-Height K-Feasible k-Bin Packing Problem ((k,K)-MBP) Given a set A of nodes with label p and a LUT input size K, partition A into k subsets such that there is a K-feasible cut of height $p-1$ for nodes in each bin.

When $k=2$ and b is half of the sum of numbers in A, the k-SNP problem becomes the number partitioning problem. We solve the k-SNP problem by dynamic programming as follows. Let $A = \{a_1, a_2, \cdots, a_n\}$ and S be the ordered set of vectors $(t_1, t_2, \cdots, t_{k-1})$ where $0 \le t_i \le b$. Clearly $|S| = (b+1)^{k-1}$. Vectors in S are ordered lexicographically. Let $\mathbf{u}_j$ represent the unit vector $(u_1, u_2, \cdots, u_{k-1})$ where $u_l = 0$ if $j \ne l$ and $u_j = 1$. Create an n rows by $|S|$ columns table T. Rows of T are associated with indexes of integers $a_1, a_2, \cdots, a_n$ respectively. Columns of T are associated with vectors in S lexicographically. An entry $T(i, \mathbf{t})$ is marked TRUE, denoted $\mathbf{T}$, if there exists disjoint sets $A'_1, A'_2, \cdots, A'_{k-1} \subseteq \{a_1, a_2, \cdots, a_i\}$ such that $\sum_{a_l \in A'_j} a_l = \mathbf{t} \cdot \mathbf{u}_j$ for $1 \le j \le k-1$. We compute T according to the following rules.

(1) $T(i, \mathbf{t}_0) = \mathbf{T}$,
(2) $T(1, a_1 \mathbf{u}_j) = \mathbf{T}$ for $1 \le j \le k-1$,
(3) $T(i, \mathbf{t}) = \mathbf{T}$ if $T(i-1, \mathbf{t}) = \mathbf{T}$,
(4) $T(i, \mathbf{t}) = \mathbf{T}$ if $T(i-1, \mathbf{s}) = \mathbf{T}$ where $\mathbf{t} = \mathbf{s} + a_i \mathbf{u}_j$ for $1 \le j \le k-1$.

A solution to the k-SNP problem exists if and only if for some $\mathbf{t} = (t_1, \ldots, t_{k-1})$, $T(n, \mathbf{t}) = \mathbf{T}$ and $\sum_{a_i \in A} a_i - \sum_{j=1}^{k-1} \mathbf{t} \cdot \mathbf{u}_j \le b$. The computation takes $O(n|S| \cdot k)$ time where $n|S|$ is from the size of table T and k is from the length of vector. The time may decrease if the table symmetry is exploited.

It is clear the (k,K)-MBP problem can be solved similarly. The difference is that the size of the min-cut of height $p-1$ is now considered as the ''total size'' of this subset during bin packing. Again, the cut size can be found in linear time using the network flow computation. Starting from $k=2$, we can do a linear search or a binary search on k to find the minimum k such that there is a solution for the (k,K)-MBP problem. This minimum k leads to an optimal solution for the min-height

K-feasible bin packing problem. We refer to this algorithm as the MC-DP algorithm.

5. Experimental Results

We have implemented the DOGMA algorithm with MC-FFD, MC-BFD, MC-MSD heuristics and the optimal MC-DP method using the C language and incorporated our implementation into the SIS package [SeSL92]. We apply the DOGMA algorithm on MCNC combinational benchmarks in our experiments. These benchmark circuits are first optimized for area and then decomposed into simple-gate networks using standard SIS routines. We apply various gate decomposition routines, including DOGMA, to obtain 2-input networks which are fed as inputs to a depth-optimal technology mapper to obtain final LUT networks so that we can compare the effects of various gate decomposition algorithms on the final depth of mapping solutions. We choose the LUT input size to be $K = 5$ in the experiments.

We first compare the performance of the heuristics and the optimal method for min-height K-feasible bin packing in DOGMA. The test circuits are flattened (2-level) area-optimized MCNC benchmarks. We decompose them by MC-FFD, MC-BFD, MC-MSD and MC-DP respectively then use the FlowMap (FM) technology mapper to get mapping solutions. Note that FlowMap obtains optimal depth for the decomposed networks. The mapping results are shown in Table 1. Some entries under MC-DP+FM are missing due to the long computation time using pseudo-polynomial time dynamic programming. We observe the four methods perform almost equally well. Because MC-FFD is faster than other three methods, the DOGMA algorithm employs the MC-FFD heuristic to solve the min-height K-feasible bin packing problem.

We now compare the DOGMA algorithm with two other decomposition routines: the *tech_decomp* (td) routine in the SIS package and the *dmig* routine developed in DAG-Map [ChCD92]. The *tech_decomp* routine is based on a balanced-tree heuristic which only minimizes the gate level locally. The *dmig* routine minimizes the gate level of the decomposed networks. In this experiment, the flattened 2-level MCNC circuits are decomposed and then fed to CutMap (CM) [CoHw95] to obtain final LUT networks. The mapping results are shown in Table 2. From the results, we see CutMap produces the same or better depth when circuits are decomposed by the DOGMA algorithm. On average, DOGMA achieves 10% and 4% depth reduction compared to *tech_decomp* and *dmig*, respectively. In the case where the mapping solution has the same depth, CutMap often obtains a smaller area for circuits decomposed by DOGMA. Note that CutMap not only guarantees optimal depth (as FlowMap) but also minimizes area during the mapping phase. One of the techniques used in CutMap for area minimization is exploiting timing slack on non-critical paths. We believe the smaller area in mapping solutions of circuits decomposed by DOGMA is from the additional timing slacks created by DOGMA and then exploited by CutMap.

We now compare our mapping solutions with previous results by various gate decomposition and technology mapping algorithms. Chortle-crf [FrRV91a] uses a set of (multi-level) area-

Circuit	MC-FFD+FM LUT	MC-FFD+FM depth	MC-BFD+FM LUT	MC-BFD+FM depth	MC-MSD+FM LUT	MC-MSD+FM depth	MC-DP+FM LUT	MC-DP+FM depth
5xp1	25	3	25	3	25	3	25	3
9sym	60	5	60	5	60	5	60	5
apex2	157	5	157	5	157	5	156	5
apex4	771	5	771	5	770	5	-	-
clip	38	3	38	3	38	3	38	3
con1	3	2	3	2	3	2	3	2
duke2	178	4	178	4	178	4	-	-
e64	108	9	108	9	108	9	108	9
misex1	16	2	16	2	16	2	16	2
misex2	36	2	36	2	36	2	-	-
misex3	190	16	190	16	190	16	187	16
rd73	24	3	24	3	24	3	25	3
rd84	53	4	53	4	53	4	53	4
sao2	44	4	44	4	44	4	42	4
vg2	29	3	29	3	29	3	27	3
Total	1732	70	1732	70	1731	70	-	-

Table 1 Comparison of MC-FFD, MC-BFD, MC-MSD and MC-DP methods in the DOGAM algorithm.

Circuit	td+CM LUT	td+CM depth	dmig+CM LUT	dmig+CM depth	**DOGMA**+CM LUT	**DOGMA**+CM depth
5xp1	24	3	23	3	24	3
9sym	66	5	66	5	59	5
apex2	154	6	155	5	151	5
apex4	770	7	792	6	770	5
clip	37	4	37	4	38	3
con1	3	2	3	2	3	2
duke2	160	5	173	4	177	4
e64	108	9	108	9	108	9
misex1	18	2	18	2	16	2
misex2	32	3	31	3	36	2
misex3	179	17	174	16	176	16
rd73	25	3	27	3	23	3
rd84	52	4	52	4	54	4
sao2	47	4	44	4	42	4
vg2	23	4	29	3	29	3
Total	1698	78	1732	73	1706	70
	-0.5%	+11%	+2%	+4%	1	1

Table 2 Comparison of *tech_decomp*, *dmig* and MC-FFD decomposition methods.

optimized MCNC benchmarks as input networks in their experiments. These benchmarks, after decomposed by the SIS *speed_up* routine or the *dmig* routine, are used by Chortle-d [FrRV91c], FlowMap [CoDi94a] and CutMap [CoHw95] as inputs to obtain LUT networks. MIS-pga-delay [MuSB91a] and TechMap-D [SaTh93] also start with area-optimized MCNC benchmarks (different from those used by Chortle-crf) followed by a delay reduction script and the *speed_up* routine to get input networks in their experiments. Both MIS-pga-delay and TechMap-D combine logic synthesis with technology mapping in their approaches. In our experiments, we take the benchmarks used by Chortle-crf as initial networks, which are subsequently decomposed by DOGMA and then fed to CutMap to obtain LUT networks. It has been shown in [CoDi94a] that FlowMap outperforms Chortle-d and MIS-pga-delay by 5% to 7% in depth and 9% to 50% in area. Therefore we compare our results with results by FlowMap, TechMap-D and CutMap respectively in Table 3. The corresponding gate decomposition and technology mapping methods taken by these approaches are listed in the second row. Comparing our results with results in [CoHw95], we can see that *speed_up* and DOGMA decompose gates equally well in terms of depth. However, 15% less LUTs are generated by CutMap for circuits decomposed by DOGMA than by *speed_up*. It indicates DOGMA is an area-efficient gate decomposition algorithm for depth-optimal technology mapping. Comparing with the results in [CoDi94a], we see *dmig* performs fairly well even though its decomposition goal is to minimize the level in the decomposed network rather than the final LUT network. Among four data sets in Table 3, TechMap-D obtains the smallest overall depth (partly due to resynthesis) at an expense of a large area overhead. On average, *speed_up* + TechMap-D generates 35% more LUTs than DOGMA + CM.

	[SaTh93]		[CoHw95]		[CoDi94a]		Ours	
	speed_up+TM-D		speed_up+CM		dmig+FM		**DOGMA**+CM	
Circuit	LUT	depth	LUT	depth	LUT	depth	LUT	depth
5xp1	17	2	23	3	22	3	24	3
9sym	9	3	62	5	60	5	59	5
9symml	9	3	58	5	55	5	50	4
C499	148	4	143	5	68	4	68	4
C880	213	7	205	8	124	8	98	8
alu2	197	8	144	8	155	9	138	9
apex6	252	5	233	4	238	5	231	5
apex7	86	4	80	4	79	4	68	4
count	71	4	69	3	31	5	31	5
des	1395	8	986	5	1310	5	938	5
duke2	175	4	178	4	174	4	173	4
misex1	18	2	15	2	16	2	16	2
rd84	16	3	45	4	46	4	53	4
rot	315	6	239	6	234	7	210	7
vg2	36	4	39	4	29	3	27	3
z4ml	9	2	12	3	5	2	5	2
Total	2966	69	2531	73	2646	75	2189	74
	+35%	-7%	+16%	-1%	+21%	+1%	1	1

Table 3 Comparison with recent previous results.

6. Conclusion

In this paper, we present an in-depth study of the structural gate decomposition problem for depth-optimal technology mapping in LUT-based FPGA. We show that further decomposition of a K-bounded network often decreases the depth in the final mapping solutions. Therefore it is always beneficial to decompose a K-bounded network into a 2-input network for depth-optimal mapping. We also show the problem of gate decomposition for depth-optimal technology mapping is NP-hard for fanin-unbounded networks when $K \geq 3$ and remains NP-hard for K-bounded networks when $K \geq 5$. We present a novel algorithm, named DOGMA, for decomposing gates in a given unbounded or K-bounded network such that a depth-optimal technology mapper can achieve a minimal depth in the final mapping solution. A key step in DOGMA is to solve the min-height K-feasible bin packing problem, for which we have developed both efficient heuristics and an optimal algorithm. The experimental results show that DOGMA outperforms some existing gate decomposition methods, such as *tech_decomp* and *dmig*, by an average of 4% to 10% in depth in the final LUT mapping solutions with almost identical LUT usage. We compare the mapping results obtained by DOGMA + CutMap with results by other gate decomposition and technology mapping algorithms. The comparison indicates that networks decomposed by DOGMA allow the depth-optimal mapper CutMap to produce LUT networks of depth comparable to previous approaches while with an average of 16% to 35% reduction in area.

Acknowledgement

This work is partially supported by NSF Young Investigator (NYI) Award MIP-9357582, and grants from Xilinx, Xerox PARC, and AT&T under NSF NYI and California MICRO programs. The authors would like to thank Dr. Robert Francis for his helpful discussions.

References

[BrRS87] Brayton, R. K., R. Rudell, and A. L. Sangiovanni-Vincentelli, ‘‘MIS: A Multiple-Level Logic Optimization,’’ *IEEE Transactions on CAD*, pp. 1062-1081, Nov. 1987.

[ChCD92] Chen, K. C., J. Cong, Y. Ding, A. B. Kahng, and P. Trajmar, ‘‘DAG-Map: Graph-based FPGA Technology Mapping for Delay Optimization,’’ *IEEE Design and Test of Computers*, pp. 7-20, Sep. 1992.

[CoDi94a] Cong, J. and Y. Ding, ‘‘An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs,’’ *IEEE Trans. on Computer-Aided Design*, Vol. **13**, pp. 1-12, Jan. 1994.

[CoHw95] Cong, J. and Y.-Y. Hwang, ‘‘Simultaneous Depth and Area Minimization in LUT-Based FPGA Mapping,’’ *Proc. ACM 3rd Int’l Symp. on Field Programmable Gate Arrays*, pp. 68-74, Feb. 1995.

[FrRV91a] Francis, R. J., J. Rose, and Z. Vranesic, ‘‘Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs,’’ *Proc. 28th ACM/IEEE Design Automation Conference*, pp. 613-619, June 1991.

[FrRV91c] Francis, R. J., J. Rose, and Z. Vranesic, ‘‘Technology Mapping of Lookup Table-Based FPGAs for Performance,’’ *Proc. IEEE Int’l Conf. on Computer-Aided Design*, pp. 568-571, Nov. 1991.

[GaJo79] Garey, M. and D. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness,* Freeman, San Francisco (1979).

[HoSa78] Horowitz, E. and S. Sahni, *Fundamentals of Computer Algorithms,* Computer Science Press, Potomac, Maryland (1978).

[LaPP94] Lai, Y.-T., K.-R. R. Pan, and M. Pedram, ‘‘FPGA Synthesis using Function Decomposition,’’ *Proc. Int’l Conf. on Computer Design: VLSI in Computers,* pp. 30-35, Oct. 1994.

[MuSB91a]
Murgai, R., N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, ‘‘Performance Directed Synthesis for Table Look Up Programmable Gate Arrays,’’ *Proc. IEEE Int’l Conf. on Computer-Aided Design*, pp. 572-575, Nov. 1991.

[SaTh92] Sawkar, P. and D. Thomas, ‘‘Technology Mapping for Table-Look-Up Based Field Programmable Gate Arrays,’’ *ACM/SIGDA Workshop on Field Programmable Gate Arrays*, pp. 83-88, Feb. 1992.

[SaTh93] Sawkar, P. and D. Thomas, ‘‘Performance Directed Technology Mapping for Look-Up Table Based FPGAs,’’ *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 208-212, June 1993.

[SeSL92] Sentovich, E., K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephen, R. Brayton, and A. Sangiovanni-Vincentelli, ‘‘SIS: A System for Sequential Circuit Synthesis,’’ *U.C.Berkeley Technical Report UCB/ERL M92/41*, May, 1992.

[Wa89] Wang, A., ‘‘Algorithms for Multi-level Logic Optimization,’’ *U.C.Berkeley Memorandum No. UCB/ERL M89/50*, April 1989.

[WuEA95] Wurth, B., K. Eckl, and K. Antreich, ‘‘Functional Multiple-Output Decomposition: Theory and an Implicit Algorithm,’’ *Proc. ACM/IEEE Design Automation Conf.*, pp. 54--59, Jun. 1995.

Appendix

The internal nodes in $N(x_i)$ (except s_i) have a mapping depth of 1. Node s_i has a mapping depth of 2. The minimum mapping depth for nodes x_i and $\overline{x}_i$ depends on how s_i is decomposed. The mapping depth for nodes q_j^k $(1 \le k \le 2K-5)$ in C_j is 2 since each of them is the root of a complete 2-level K-ary tree with PI nodes as leaves. Similarly, the mapping depth for nodes r_j^k $(1 \le k \le K-2)$ is 3. The minimum mapping depth for node C_j depends on the mapping depth of its three literal nodes. The minimum mapping depth of $N(F)$ after decomposition is the maximal mapping depth among all decomposed C_j nodes. Hence $MMD(N(F))$ depends on how the node s_i is decomposed in every $N(x_i)$. Before we prove Theorems 1 and 2, we introduce the following lemmas.

Lemma 7 For the subnetwork $N(x_i)$ in Figure 5 and its decomposition $D(N(x_i))$, if $MMD_{D(N(x_i))}(x_i)=2$ then $MMD_{D(N(x_i))}(\overline{x}_i)=3$, and if $MMD_{D(N(x_i))}(\overline{x}_i)=2$ then $MMD_{D(N(x_i))}(x_i)=3$.

Proof The minimum mapping depth for nodes x_i and $\overline{x}_i$ after decomposition depends on how the node s_i is decomposed in $N(x_i)$. We denoted the decomposed $N(x_i)$ as $D(N(x_i))$. Figure 10 shows all the six possible decompositions of s_i. In Figure 10(a) and (b), nodes PI_i^1 and PI_i^2 are merged into a node t_1 followed by different subsequent decomposition steps. It creates a min-cut of height 1 for node s_i with the cutset $\{w_i^1, w_i^2, t_1\}$. Together with nodes v_i^1 through v_i^{K-3}, a K-feasible cut of height 1 exists for node x_i. Hence $MMD_{D(N(x_i))}(x_i)=2$. On the other hand, the min-cut of height 1 for node $\overline{x}_i$ has a size of $K+1$. Hence $MMD_{D(N(x_i))}(\overline{x}_i)=3$. In Figure 10(c), the min-cut of height 1 for node s_i has a size of 4. As a result both $MMD_{D(N(x_i))}(x_i)=3$ and $MMD_{D(N(x_i))}(\overline{x}_i)=3$. In Figure 10(d) and (e), the min-cut of height 1 for node s_i has a size of 3. However, due to the edge connection (w_i^2, x_i) shown in Figure 5, the min-cut of height 1 for node x_i still has a size of $K+1$. As a result both $MMD_{D(N(x_i))}(x_i)=3$ and $MMD_{D(N(x_i))}(\overline{x}_i)=3$. In Figure 10(f), node w_i^1 is merged with PI_i^1 and w_i^2 is merged with PI_i^2 which create a 2-feasible cut of height 1 for node s_i. Together with nodes u_i^1 through u_i^{K-2}, node $\overline{x}_i$ has a K-feasible cut of height 1. Hence $MMD_{D(N(x_i))}(\overline{x}_i)=2$. However, both nodes w_i^1 and w_i^2 connects to node x_i. As a result the min-cut of height 1 for node x_i after decomposition has a size of $K+1$. So $MMD_{D(N(x_i))}(x)=3$. From the above enumeration of decompositions for node s_i, it is easy to see Lemma 7 holds. □

Lemma 8 For the subnetwork $N(x_i)$ shown in Figure 5, there exist a 2-feasible cut of height 2 for node x_i after decomposition. Similarly, there exists a 2-feasible cut of height 2 for node $\overline{x}_i$ after decomposition.

Proof In $input(x_i)$ shown in Figure 5, $K-1$ nodes has a mapping depth of 1 while the node s_i has a depth of 2. The $K-1$ nodes can be merged into a node which has a minimum mapping depth of 2. As a result, there exists a 2-feasible cut of height 2 for node x_i. Similarly, there exists a 2-feasible cut of height 2 for node $\overline{x}_i$. The 2-feasible cut of height 2 is illustrated in Figure 6(b) and 8(b) by 2 nodes (labeled 2) respectively. □

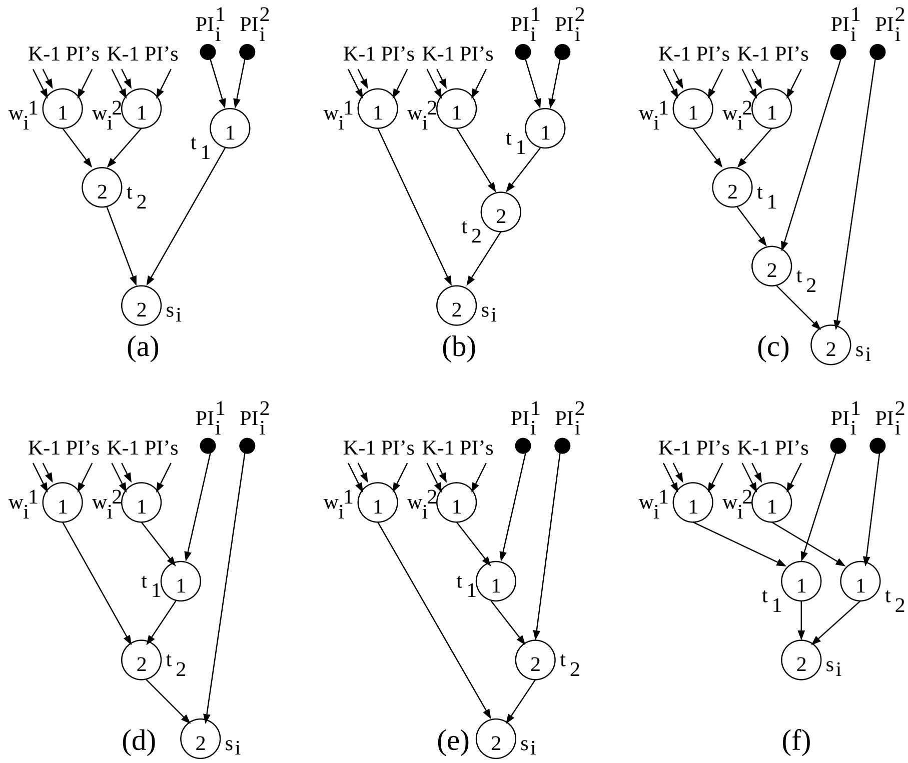


Figure 10 Six different ways to decompose node s_i in $N(x_i)$.
In (a) and (b), $MMD(x_i)=2$ and $MMD(\overline{x_i})=3$; In (c), (d) and (e), $MMD(x_i)=MMD(\overline{x_i})=3$;
Only in (f) $MMD(x_i)=3$ and $MMD(\overline{x_i})=2$.

Lemma 9 For the network $N(C_j)$ in Figure 6(a) and its decomposition $D(N(C_j))$, $MMD_{D(N(C_j))}(C_j)=4$ if and only if $MMD_{D(N(C_j))}(l_j^k)=2$ for some $1 \le k \le 3$.

Proof In the subnetwork $N(C_j)$, $input(C_j)$ contains $2K-5$ nodes of minimum mapping depth 2, $K-2$ nodes of minimum mapping depth 3, and 3 literal nodes l_j^1, l_j^2, l_j^3. Let $D(N(C_j))$ denoted the decomposed $N(C_j)$. Consider the following two cases in computing $MMD_{D(N(C_j))}(C_j)$. First, assume $MMD_{D(N(C_j))}(l_j^1)=MMD_{D(N(C_j))}(l_j^2)=MMD_{D(N(C_j))}(l_j^3)=3$. According to Lemma 8, there exist 2-feasible cuts of height 2 for nodes l_j^1, l_j^2, l_j^3 respectively. Together with the $2K-5$ nodes of depth 2 in $input(C_j)$, the cut of height 2 will have a size of $2K+1$. As a result, during the decomposition of C_j, 3 nodes of mapping depth 3 have to be introduced in order to merge the $2K+1$ nodes in the cut of height 2. But there are already $K-2$ nodes of mapping depth 3 in $input(C_j)$. Hence the min-cut of height 3 for C_j in $D(N(C_j))$ have a size of $K+1$. This will cause $MMD_{D(N(C_j))}(C_j)=5$. On the other hand, if at least one literal node has a depth of 2, as shown in the example in Figure 6(b)) where the node l_j^3 has a mapping depth of 2, only 2 nodes of mapping depth 3 will be introduced. Together with the $K-2$ nodes of depth 3 in $input(c)$, $MMD_{D(N(C_j))}(C_j)=4$ can be obtained. □

Lemma 10 The decomposed network $D(N(F))$ of $N(F)$ satisfies $MMD(D(N(F)))=4$ if and only if there is a way to decompose the node s_i in every $N(x_i)$ such that for all $N(C_j)$ at least one literal node of $N(C_j)$ has a mapping depth of 2.

Proof The minimum mapping depth of $N(F)$ after decomposition is the maximal mapping depth among m output nodes $C_1, C_2, \cdots, C_m$. In order to have a mapping depth of 4, it must be the case $MMD_{D(N(C_j))}(C_j)=4$ for $1 \le j \le m$. According to Lemma 9, at least one of the literal nodes in each $N(C_j)$ must have a mapping depth of 2. Literal nodes are from either node x_i or node $\overline{x}_i$ of some network $N(x_i)$. From Lemma 7, only one of them can have a mapping depth of 2. Which one has a depth of 2 depends on whether the node s_i is decomposed in the way specified by Figure 10(a) or Figure 10(f). As a result the minimum mapping depth of $N(F)$ after decomposition would be 4 if and only if there is a way to decompose the node s_i in each $N(x_i)$ such that at least one literal node in each $N(C_j)$ has a mapping depth of 2. This proves Lemma 10. □

[Proof of Lemma 5] We link the truth assignment of Boolean variables in an instance F of 3SAT to the decomposition of gate s_i in all $N(x_i)$ as follows: a Boolean variable $x_i=1$ in 3SAT if $MMD_{D(N(x_i))}(x_i)=2$, and $\overline{x}_i=1$ in 3SAT if $MMD_{D(N(x_i))}(\overline{x}_i)=2$. It leads to the conclusion that the 3SAT instance F is satisfiable if and only if $MMD(N(F))=4$. This proves Lemma 5. □

[Proof of Theorem 1] The transformation from an instance F of 3SAT to the network $N(F)$ takes $O(K^3(n+m))$ time. The network $N(F)$ consists of subnetworks $N(x_i)$ for $1 \le i \le n$ and subnetworks $N(C_j)$ for $1 \le j \le m$. Every $N(x_i)$ and $N(C_j)$ is well-defined for $K \ge 3$. From Lemma 5, F is satisfiable if and only if $N(F)$ has a decomposition $D(N(F))$ such that $MMD(D(N(F)))=4$. As a result the SGD/K-D problem is NP-Complete. Hence the SGD/K problem is NP-hard. □

[Proof of Lemma 6] If all the three literal nodes l_j^1, l_j^2, l_j^3 have a minimum mapping depth of 3 after decomposition, according to Lemma 8, the min-cut of height 2 for node C_j will have a size of $K-5+6=K+1$. Hence node C_j has a minimum mapping depth of 4 after decomposition. On the other hand, if at least one of the literal node has a mapping depth of 2, a K-feasible cut of height 2 exists for node C_j. An example is shown in Figure 8(b) where the literal node l_j^3 is assumed a mapping depth of 2 after decomposition. As a result node C_j has a mapping depth of 3 after decomposition if and only if at least one of the three literal nodes l_j^1, l_j^2, l_j^3 has a mapping depth of 2 after decomposition. This proves Lemma 6. □

[Proof of Theorem 2] Similar to the proof of Theorem 1 and omitted. □