FPGA向けテクノロジ・マッピングにおける深さ最小ネットワーク生成のための効率的なカット列挙手法

高田, 大河
九州大学

松永, 裕介
九州大学

出版情報：電子情報通信学会技術研究報告. 108 (412/413/414), pp.57-62, 2009-01-29. IEICE
バージョン：
権利関係：

社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

FPGA 向けテクノロジ・マッピングにおける深さ最小ネットワーク生成のための効率的なカット列挙手法

高田　大河† 　松永　裕介†

† 九州大学

あらまし　本稿では，LUT 型 FPGA 向けテクノロジ・マッピングにおいて，深さ最小なネットワークの生成を目的とした効率的なカット列挙手法を提案する．カットの数はカットのサイズに対して指数的に増加するため，サイズが大きいカットの全列挙には時間がかかる．提案手法は，深さ最小なネットワークの生成を保証しつつ限られたカットのみを列挙することによって，既存手法よりも高速にカットの列挙を行う．カットのサイズを 8 および 9 とした実験の結果，提案手法はボトムアップ型の全列挙手法と比べてそれぞれ 6 倍および 16 倍，トップダウン型の全列挙手法と比べて 2 倍の早さでカットを列挙した．全てのカットを用いて生成したネットワークの段数と提案手法で列挙したカットを用いて生成したネットワークの段数は等しい．また，全てのカットを用いて生成したネットワークの LUT 数と比較して，提案手法で列挙したカットを用いて生成したネットワークの LUT 数はわずかに 4%ほど大きかった．

キーワード　再構成可能システム，FPGA，テクノロジ・マッピング，カット列挙

An Efficient Cut Enumeration for Depth-Optimum Technology Mapping for LUT-based FPGAs

Taiga TAKATA† and Yusuke MATSUNAGA†

† Kyushu University

Abstract　This paper presents a top-down cut enumeration for depth-minimum technology mapping for LUT-based FPGAs. Enumerating all cuts with large size consumes long run-time because the number of cuts increases with the size of cuts. The proposed algorithm enumerates partial cuts with a guarantee that a depth-minimum network can be constructed, and runs faster than enumerating all cuts. The experimental results show that the proposed algorithm runs about 6 times and 16 times faster than bottom-up exhaustive enumeration for $K = 8, 9$, respectively. The proposed algorithm also runs about 2 times faster than top-down exhaustive enumeration for $K = 8, 9$, respectively. Area of network derived by the set of cuts enumerated by the proposed algorithm is only 4 % larger than that derived by exhaustive enumeration on average, and the depth is the same.

Key words　reconfigurable system, FPGA, technology mapping, cut enumeration

1. はじめに

LUT(LookUp-Table) 型 FPGA(Field Programmable Gate Array) 向けテクノロジ・マッピングは，ブーリアン・ネットワークを K 入力 LUT からなる論理的に等価なネットワークに変換する処理である(注1)．遅延最小を目的とした既存のテクノロジ・マッパー [3] [4] [6] [5] [2] [10] [9] [1] [7] における最優先の目的は深さ最小であり，最小深さにおける次の目的は面積最小である．深さは最長パスの長さを指し，面積は LUT 数を指す．

テクノロジ・マッピングは一般に，ブーリアン・ネットワークを K フィージブル・コーンで被覆する問題として扱われる．K フィージブル・コーンとはブーリアン・ネットワークの部分グラフであり，ブーリアン・ネットワークにおけるサイズが K 以下のカットから導かれる．近年のテクノロジ・マッパーは，まずカットを列挙し，各カットから導かれる K フィージブル・コーンを使ってブーリアン・ネットワークを被覆するという枠組みを採用している．被覆を行う前に良いカットを知ることは難しいため，良いネットワークを生成するためには多数のカットを列挙する必要がある．しかしカットのサイズが 8 以上の場合，カットの全列挙は非常に大きい実行時間を要するという問題がある．

既存のカット列挙手法は，ボトムアップなアルゴリズムとトップダウンなアルゴリズムに分類できる．既存の多くのカット列挙手法 [8] [5] [2] [10] [1] は，ボトムアップなアルゴリズムに基づいている．ボトムアップなアルゴリズムとは，各ノード

(注1)：本稿では，K 入力 LUT を単に LUT と表し，LUT からなるネットワークを LUT ネットワークと表す．

のファンインにおけるカットをマージすることによって新たなカットを生成するものである．ボトムアップなアルゴリズムにおいては，各ノードに対してファンインのカットの集合の直積を計算する必要がある．この直積の計算は直積のサイズに比例した手間を必要とするが，実際のカット数はその直積のサイズよりもはるかに小さい．直積の計算を必要とすることは，ボトムアップなアルゴリズムの欠点である．一方，トップダウンなアルゴリズムはそのような直積の計算を必要としない．トップダウンなアルゴリズムは，根となるノードのみを要素とするカットから始め，カットの一部をその入力方向へ展開することで新たなカットを生成する．これまでカットの展開をどこで停止すればよいか明らかでなかったため，全てのカットを列挙するためにはプライマリ・インプットまでカットの展開を繰り返す必要があった．このことからトップダウンなアルゴリズムは非効率的であるとみなされ，カット列挙にほとんど持ちいられてこなかった．近年，カットの展開の停止条件を適切に定めることによって，ボトムアップなアルゴリズムよりも効率的に全てのカットを列挙するトップダウンなアルゴリズムが提案された [11]．しかし，サイズが大きいカットの全列挙には依然として大きな実行時間を必要とする(注2)．

カット列挙が大きな実行時間を要する主な原因は，カットの数が K に対して指数的に増加することである．この問題に対し，カットの全列挙を避け限られたカットのみを列挙することで高速に動作するボトムアップなアルゴリズム [7] が存在する．しかし文献 [7] の手法はヒューリスティクスに基づいており，ネットワークの最小深さを保証できない．さらに，文献 [7] の手法が生成するネットワークの面積は，全てのカットを用いて生成されるネットワークの面積よりも大きくなりやすい(注3)．質が高いネットワークの生成を求められるアプリケーションに対して，文献 [7] の手法は不適切であると思われる．

本稿は，列挙されたカットを用いて深さ最小な被覆を行った場合の LUT ネットワークの最小深さを保証しつつ，限られたカットのみを列挙することによって高速にカット列挙を行うアルゴリズムを提案する．提案手法は，各ノードにおいて少なくとも 1 つ以上のラベル・カットを列挙する．あるノードのラベル・カットとは，そのノードの論理を深さ最小なネットワークで実現するために必要となるカットである．本稿で，ラベル・カットのみから生成されるネットワークが深さ最小であることを証明する．また提案手法は，列挙するカットを限定することによる面積の増大を抑えるため，面積の最小化に有効と思われるカットを列挙する．実験の結果，提案手法は既存のボトムアップな全列挙手法と比べ，K が 8 および 9 の場合にそれぞれ 6 倍および 16 倍の早さでカットを列挙した．また，提案手法はトップダウンな全列挙手法と比べ，K が 8 および 9 の場合にどちらもおよそ 2 倍の早さでカットを列挙した．提案手法と全列挙手法のどちらもネットワークの最小深さは保証されるため，生成したネットワークの深さは等しかった．提案手法により生成したネットワークの面積は，全てのカットを用いて生成したものよりもわずかに 4% ほど大きかった．

本稿は，以下の構成をとる．第 2 章で基本的な定義を述べ，第 3 章で既存のカット全列挙手法を解説する．第 4 章で提案手法のアルゴリズムを解説する．第 5 章で実験結果を示し，第 6 章で本稿をまとめる．

2. 準　　備

テクノロジ・マッピングの入力は，ブーリアン・ネットワークと自然数 K である．与えられるブーリアン・ネットワークは，**サブジェクト・グラフ**と呼ばれる．自然数 K は LUT の最大入力数に対応し，デバイスによって決まる．テクノロジ・マッピングの出力は，**LUT ネットワーク**である．

サブジェクト・グラフは，ノードの集合と辺の集合からなる DAG(Directed Acyclic Graph:非循環有向グラフ) である．サブジェクト・グラフの各ノードは，K 入力以下の論理関数を表す．ノード i がノード j の入力であるとき，有向辺 (i, j) が存在する．ノード v のファンインとは，v の入力辺を介して v と隣接するノードを指す．v のファンインの集合は $FI(v) = \{u \mid \exists(u, v) \in E\}$ である．サブジェクト・グラフの各ノードに対し一般に，ファンインの数が K 以下でなければならないという制約がある．本稿では議論の簡単化のため，各ノードのファンイン数は 2 以下であると仮定する．以降の議論は，各ノードのファンインの数が 2 以上の場合にも適用できる．ノード v のファンアウトとは，v の出力辺を介して v と隣接するノードを指す．v のファンアウトの集合は $FO(v) = \{w \mid \exists(v, w) \in E\}$ である．$FI(v) = \phi$ であるノードを**プライマリ・インプット**，$FO(v) = \phi$ であるノードを**プライマリ・アウトプット**と呼ぶ．プライマリ・インプットの集合およびプライマリ・アウトプットの集合をそれぞれ PI, PO で表す．ノード v の推移的ファンインとは，全てのプライマリ・インプットから v までのパスに含まれるノードの集合である．正確には，v の推移的ファンイン $TFI(v)$ は $TFI(v) = \{v\} \cup \bigcup_{u \in FI(v)} TFI(u)$ で定義される．v の**推移的ファンイン・グラフ**とは $TFI(v)$ で誘導される部分グラフであり，$TFIG(v)$ で表される．

$TFIG(v)$ のセパレータ s とは，プライマリ・インプットから v までの全てのパス p に対して，p に含まれる少なくとも 1 つのノードが必ず s に含まれるようなノードの集合である．以下の条件を満たす s を，$TFIG(v)$ の極小セパレータと呼ぶ．

- s は $TFIG(v)$ のセパレータである．
- s は $TFIG(v)$ の他の全てのセパレータを包含しない．

カット (v, s) とは，ノード v と $TFIG(v)$ の極小セパレータ s の組である．ノード v に対し，カット $(v, \{v\})$ は v のトリビアルなカットと呼ばれる．カット (v, s) に対し，v はカットの**根**と呼ばれ $RT((v, s))$ で表される．カット (v, s) に対し，s はカットの**葉**と呼ばれ $LEAF((v, s))$ で表される．カット c に対し，$|LEAF(c)|$ は c のカット・サイズと呼ばれる．カット・サイズが K 以下のカットを，**K フィージブル・カット**と呼ぶ．本章の以下では，K フィージブル・カットを単にカットと呼ぶ．v における全てのカットの集合を $\Phi_K(v)$ で表す．カットの集合 C とカット $c \in C$ に対し，カット・ファンイン $CFI(c, C)$ を $CFI(c, C) = \{c' | c' \in C, RT(c') \in LEAF(c)\}$ で定義する．カット c に対し，**K フィージブル・コーン** $KFC(c)$ は $RT(c)$ と $LEAF(c)$ の間にあるノードの集合で誘導される部分グラフである．正確には，K フィージブル・コーンは以下で定義されるノードの集合 $V_{interv}(v, V)$ に誘導される部分グラフである．

$$V_{interv}(v, V) = \{v\} \cup \bigcup_{u \in FI(v) - V} V_{interv}(u, V)$$

カット c により導かれた K フィージブル・コーン C において，C の根とは $RT(c)$ を指し，$RT(C)$ と表される．また，C の入力の集合とは $LEAF(c)$ を指し，$INPUT(C)$ で表される．

K フィージブル・コーンにおいて，$|INPUT(C)|$ は K 以下である．従って，K フィージブル・コーンの論理関数は 1 つの LUT で実現できる．LUTL が K フィージブル・コーン $KFC(c)$ の論理関数を実現するとき，L の入力および出力はそれぞれ $INPUT(KFC(c)) = LEAF(c)$, および $RT(KFC(c)) = RT(c)$ にあたる．カットの集合 S が以下の 3 つの条件を満たすとき，S は**実現可能**であるという．

- $\forall i \in PO, (\exists c \in S, i = RT(c)) \lor i \in PI$
- $\forall c \in S,\ \forall i \in LEAF(c), (\exists c' \in S, i = RT(c')) \lor i \in PI$

(注2)：実験の結果，文献 [11] における手法は，ITC'99 ベンチマークにおける “b17_1” に対してサイズ 9 のカット列挙におよそ 4 時間ほど必要とする．

(注3)：MCNC ベンチマークや ITC'99 ベンチマークの各ネットワークに対する実験の結果，文献 [7] の手法によるネットワークの面積は，全てのカットを用いた深さ最小なマッパー [9] によるものと比較して平均 11.4% ほど大きかった．

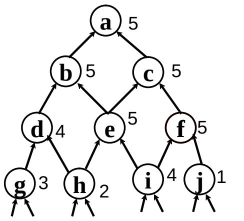

図 1　サブジェクト・グラフの例

- S にトリビアルなカットが含まれない．

テクノロジ・マッピングの問題は，実現可能なカットの集合を求める問題として定義できる．実現可能なカットの集合 S に以下の操作を行うことで，LUT ネットワークが生成される．

- サブジェクト・グラフの各プライマリ・インプットに対し，LUT ネットワークのプライマリ・インプットを生成する．
- S における各カット c に対し，$KFC(c)$ の論理関数を実現するノードを生成する．
- $KFC(c)$ の論理関数を実現する LUT ネットワークのノードを L_b と表す．S における各カット c および $CFI(c,S)$ に含まれる各カット i に対し，辺 (L_i, L_c) を生成する．

LUT ネットワークの**深さ**とは，プライマリ・インプットからプライマリ・アウトプットまでの最長なパスの長さである．

サブジェクト・グラフのノード v に対し，**ラベル** $L_K(v)$ を定義する．ラベル $L_K(v)$ は，$TFIG(v)$ に対する深さ最小な LUT ネットワークの深さである．プライマリ・インプットのラベルは 0 である．サブジェクト・グラフに対する深さ最小な LUT ネットワークの深さは，サブジェクト・グラフにおける全てのノードの最大ラベルに等しい．カット c の**高さ** $H(c)$ は，$LEAF(c)$ における最大ラベルを表し，$H(c) = \max_{i \in LEAF(c)} L_K(i)$ で定義される．v のラベルは，v を根とする各カットの高さを用いて $L_K(v) = \min_{c \in \Phi_K(v)} H(c) + 1$ と計算できる．ノード v において，$H(c) + 1 = L_K(v)$ であるような v を根とするカット c はラベル・カットと呼ばれる．$TFIG(v)$ を深さ最小となるよう被覆するためには，v のいずれかのラベル・カット c から導かれる $KFC(c)$ を用いて v を被覆する必要がある．

図 1 に，サブジェクト・グラフの例を示す．円および矢印は，それぞれノードおよび辺を表す．ノードの横の数字は，ノードのラベルを表す．$L_K(d), L_K(e), L_K(f)$ はそれぞれ 4, 5, 5 なので，3 フィージブル・カット $(a, \{d, e, f\})$ の高さは 5 である．a において高さ 4 以下の 3 フィージブル・カットが存在しないので，a のラベルは 6 である．また，$(a, \{d, e, f\})$ は a のラベル・カットの 1 つである．

3.　既存のカット全列挙手法

3.1　ボトムアップに基づくカット列挙手法

カット C が K フィージブル・カットである場合にのみ真を返し，それ以外のカットには偽を返す論理関数を $ISKF(C)$ と表す．ノード v と 2 つの K フィージブル・カットの集合 A, B に対し，カット・マージ $A \bowtie_{(K,v)} B$ を以下のように定義する [5]．

$$A \bowtie_{(K,v)} B = \{(v, LEAF(s) \cup LEAF(t)) \mid s \in A, t \in B, ISKF((v, LEAF(s) \cup LEAF(t))) = true\}$$

各ノードの全ての K フィージブル・カットは，以下の定理に基づいて PI から PO へトポロジカルな順序で列挙される．

［定理 1］　$FI(v) = \{u_1, u_2\}$ と表すと，v の K フィージブル・カットの集合 $\Phi_K(v)$ は次のように計算できる [5]．

$$\Phi_K(v) = \begin{cases} \{(v, \{v\})\} & v \in PI \\ \{(v, \{v\})\} \cup (\Phi_K(u_1) \bowtie_{(K,v)} \Phi_K(u_2)) & other. \end{cases}$$

カット・マージの処理 $\Phi_K(u_1) \bowtie_{(K,v)} \Phi_K(u_2)$ には問題がある．$M = |\Phi_K(u_1)|$ および $N = |\Phi_K(u_2)|$ とする．$\Phi_K(v)$ を生成するためのカット・マージの処理の計算複雑度は，$O(M \times N)$ である(注4)．もし $\Phi_K(v)$ のサイズが $O(M \times N)$ ならばこの計算複雑度は妥当であるといえるが，多くの場合は $\Phi_K(v)$ のサイズはカットの直積のサイズ $M \times N$ よりもはるかに小さい．$\{(v, LEAF(s) \cup LEAF(t)) \mid s \in u_1, t \in u_2\}$ はカット・サイズが K より大きいカットや，カットにならない要素を含む可能性がある．図 1 を用いて例を示す．定理 1 に従って $\Phi_3(a)$ を計算するならば，$\Phi_3(b) \bowtie_{(3,a)} \Phi_3(c)$ を計算しなければならない．b, c における 3 フィージブル・カットの集合はそれぞれ，$\Phi_3(b) = \{(b, \{b\}), (b, \{d, e\}), (b, \{d, h, i\}), (b, \{g, h, e\}), (b, \{g, h, i\})\}$，$\Phi_3(c) = \{(c, \{c\}), (c, \{e, f\}), (c, \{h, i, f\}), (c, \{e, i, j\}), (c, \{h, i, j\})\}$ である．$\Phi_3(b)$ と $\Phi_3(c)$ のサイズはどちらも 5 なので，両者の直積をとると 25 個の要素を持つ集合が得られる．しかしながら，a における 3 フィージブル・カットの集合は $\Phi_3(a) = \{(a, \{a\}), (a, \{b, c\}), (a, \{b, e, f\}), (a, \{c, d, e\}), (a, \{d, e, f\})\}$ の 5 つのみであり，他の 21 個の要素を生成して 3 フィージブル・カットであるかどうか調べる手間は無駄になる．ボトムアップなアルゴリズムに基づき K フィージブル・カットを列挙する限り，この非効率性は避けられない．

3.2　トップダウンに基づくカット列挙手法

3.2.1　カットの展開

カットの展開 $EXP(u, C)$ とは，カット C の葉からノード u を除き $FI(u)$ を加えることで新たなカットを生成する処理である．正確には，$EXP(u, C) = (RT(C), (LEAF(C) - \{u\}) \cup FI(u))$ で表される．

ノード v において $(v, \{v\})$ を除くいずれのカットも，他のいずれかのカットに対するカットの展開によって生成できる．従って，$(v, \{v\})$ からスタートしカットの展開を繰り返し実行することによって，ノード v における全てのカットの集合を生成することができる．図 1 のサブジェクト・グラフに対して，カットの展開の繰り返しによって生成される a のカットの集合を図 2 に示す．図 2 において，カットの根はすべて a であるため省略しカットを葉で表している．図中の辺は，矢印の尾のカットに対するカットの展開によって矢印の頭のカットが生成されたことを表す．辺の隣のアルファベットは，カットの展開によって葉から取り除かれたノードの名を表す．カットの展開を単純に繰り返す際には，同じカットが重複して生成される可能性がある．例えば，$C = EXP(b, (a, \{b, c\}))$ がまず実行された後に $EXP(c, C)$ が実行されれば，結果として $C'' = (a, \{d, e, f\})$ が得られる．一方で，$C' = EXP(c, (a, \{b, c\}))$ がまず実行された後に $EXP(b, C')$ が実行されれば，この場合もまた C'' が得られる．このような無駄な重複を避けるためには，全てのノードに対してカットの展開の際に葉から取り除く優先順序を決めておき，その順序に基づいてカットの展開を行えばよい．カット列挙の間にノード間の順序関係が変わらない限り，全てのノード間の順序は任意である．

サブジェクト・グラフに再収斂するパスが存在した場合，カット・サイズが K より大きいカットを展開することによって K フィージブル・カットが得られる場合がある．しかし，カットの葉が全てプライマリ・インプットになるまでカットの展開を繰り返すことは効率が悪い．効率的に K フィージブル・カットを列挙するためにはカットの展開を止める条件が必要となる．

3.2.2　カットの展開の停止条件

本章では，全ての K フィージブル・カットを列挙するためのトップダウンに基づく効率的なアルゴリズムを示す．提案するアルゴリズムは，ノード v の K フィージブル・カットを列挙する際に，$(v, \{v\})$ からスタートしてカットの展開を繰り返すことで新たな K フィージブル・カットを生成する．提案するアル

(注4)：K フィージブル・カットの葉のサイズは高々K なので，1 つの K フィージブル・カットを扱う手間は定数とみなす．

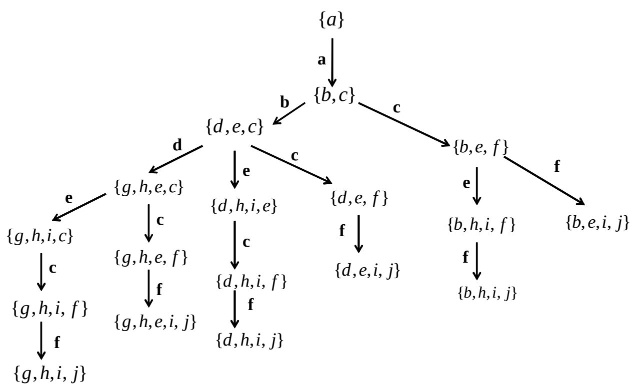

図 2　Cut Enumeration based on Expansions of Cuts

ゴリズムは，カットの展開を停止する条件を適切に定めることによって探索する範囲を狭め，効率良く処理を行う．あるノードにおける K フィージブル・カットの集合を $X = \{C_1, C_2, C_i\}$ とおき，$\Psi(X) = \bigcup_{i=1}^{n} LEAF(C_i)$ と表す．$\Psi(\Phi_K(v))$ は，v の全ての K フィージブル・カットの葉に含まれるノードの集合を意味する．定理 1 より，以下の補題が導かれる．

［補題 1］ プライマリ・インプットを除く全てのノードに対し，以下の条件が成り立つ．

$$\Psi(\Phi_K(v)) \subseteqq \{v\} \cup \Psi(\Phi_K(u_1)) \cup \Psi(\Phi_K(u_2)) \quad (1)$$
$$where \quad FI(v) = \{u_1, u_2\}$$

証明

プライマリ・インプットでない任意のノード v に対し，定理 1 から以下の式が導かれる．

$$\begin{aligned}\Psi(\Phi_K(v)) &= \Psi(\{(v, \{v\})\} \cup (\Phi_K(u_1) \bowtie_{(K,v)} \Phi_K(u_2))) \\ &= \{v\} \cup \Psi(\Phi_K(u_1) \bowtie_{(K,v)} \Phi_K(u_2)) \quad (2)\end{aligned}$$

一方で，K フィージブル・カットの集合 A と B において，以下の条件が成り立つ．

$$\begin{aligned}\Psi(A \bowtie_{(K,v)} B) &\subseteqq \Psi(\bigcup_{s \in A} \bigcup_{t \in B} LEAF(s) \cup LEAF(t)) \\ &= \Psi(\bigcup_{s \in A} LEAF(s) \cup \bigcup_{t \in B} LEAF(t)) \\ &= \Psi(A) \cup \Psi(B) \quad (3)\end{aligned}$$

式 (2) と式 (3) より，以下の式が導かれる．

$$\Psi(\Phi_K(v)) \subseteqq \{v\} \cup \Psi(\Phi_K(u_1)) \cup \Psi(\Phi_K(u_2)) \quad (4)$$

□

補題 1 より，以下の定理が導かれる．

［定理 2］ $i \notin \{v\} \cup \Psi(\Phi_K(u_1)) \cup \Psi(\Phi_K(u_2))$ である任意のノード i および i を葉に持つ任意のカット C に対して，カットの展開 $EXP(i, C)$ は K フィージブル・カットを生成しない．

定理 2 に従って，PI から PO へ向かってトポロジカルな順序で各ノードにおける全ての K フィージブル・カットを効率的に列挙することができる．$m = |\Psi(\Phi_K(u_1))|$, $n = |\Psi(\Phi_K(u_2))|$ とおくと $\{v\} \cup \Psi(\Phi_K(u_1)) \cup \Psi(\Phi_K(u_2))$ の計算複雑度は $O(m+n)$ である．もしカットの展開によって生成される K フィージブル・カットの数が E ならば，トップダウンなアルゴリズムの計算複雑度は $O(E)$ である．カットの展開によって生成される全てのカットが K フィージブル・カットではないが，しかし上記にあるカットの展開の停止条件はタイトであるので，提案するトップダウンなアルゴリズムの計算複雑度はおおむね列挙する K フィージブル・カット数に比例すると考えられる．

```
Partial_Enumeration ( SubjectGraph G, int K ){
  (v1, v2, ..., vn) ← topological_sort (G) ;
  for each vi {
    // Calculate the label of vi with labeling in FlowMap [3]);
    CalcLabel(vi);
  }
  for each vi { //cuts enumeration phase
    u1, u2 ← FI(vi) ;
    Check-mark all the nodes w,
        where w ∈ {v} ∪ Ψ(ΩK(u1)) ∪ Ψ(ΩK(u2));
    LabelCutsEnumeration(vi);
    AreaCutsEnumeration(vi);
  }
}
```

図 3　提案するトップダウンなカット列挙アルゴリズムの枠組み

4. 列挙するカットを限定したトップダウンなカット列挙手法

4.1 枠　組　み

提案手法は，深さ最小な被覆アルゴリズムと組み合わせた場合の最小深さを保証しつつ，限られた K フィージブル・カットのみを列挙するトップダウンなアルゴリズムである．提案手法は，各ノードにおいてラベル・カットを少なくとも 1 つ以上含む限定的なカットを列挙する．提案手法はラベル・カットを効率よく列挙するために，トップダウンなアルゴリズムにおけるカットの展開と FlowMap [3] におけるラベル付けを組み合わせている．以降は K フィージブル・カットでないカットを扱わないので，K フィージブル・カットを単にカットと呼ぶ．

提案するアルゴリズムの枠組みは，図 3 の通りである．図中の $\Omega_K(v_i)$ はノード v_i において列挙されたカットの集合を表す．$\Psi(S)$ は 3.2 節と同様，カットの集合 S における全てのカットの葉に含まれるノードの集合を指す．提案アルゴリズムは 2 つの段階からなる．すなわち，ラベル付けとカット列挙である．

ラベル付けの段階においては，サブジェクト・グラフの各ノードのラベルが計算される．各ノードのラベルは，全てのカットを列挙することなく，FlowMap [3] において用いられているラベル付けの技術を用いて計算される．m, n をそれぞれサブジェクト・グラフにおける辺の数およびノードの数とすると，全てのノードのラベルは $O(Kmn)$ で計算できる．本稿では紙面の都合上，ラベル付けの詳細は省略する．

カット列挙の段階において，各ノードのカットは PI から PO へ向かうトポロジカルな順序で計算される．1 つのノードのカット列挙は，2 つの段階からなる．すなわち，ラベル・カットの列挙とエリア・カットの列挙である．$FI(v) = \{v_1, v_2\}$ であるとき，条件 $w \in \{v\} \cup \Psi(\Omega_K(v_1)) \cup \Psi(\Omega_K(v_2))$ を満たすノード w の集合を $FTP(v)$ とおく．カット列挙において，$FTP(v)$ に含まれないノード i に対するカット展開 $EXP(i, C)$ は実行しない．これは，定理 2 に基づくヒューリスティックである．

4.2 ラベル・カットの列挙

ノード v のラベル・カットを列挙することを考える．v のラベルを p とする．v のカットの集合は 2 つのカットの集合に分割できる．すなわち，ラベル p のノードを葉に含むカットの集合と，それ以外のカットの集合である．v においてラベル p のノードを葉に含むカット C は，$H(C) + 1 > p$ であるためラベル・カットではない．一方，ノード v のラベルは v よりプライマリ・インプット側にあるノードのラベルより小さくなることはないので [3]，$TFIG(v)$ における各ノードのラベルは p であるかまたは p より小さい．ラベル p のノードを葉に含まないカット C' において $LEAF(C')$ の各ノードのラベルは p よりも小さいことから，$H(C') + 1 \leqq p$ である．このとき，v のラベルは p であることから $H(C') + 1 = p$ である．従って，ラ

ベル p のノードを葉に含まないカットの集合は，v のラベル・カットの集合と等しい．以上より，ラベル p のノード v におけるカット C がラベル・カットである必要十分条件は以下の通りである．

$$\forall i \in LEAF(C), L_K(i) \neq p$$

上記に基づき，ラベル p のノードを葉に含まないカットが列挙される．ラベル p のノードを一時的に v と一緒にまとめて v' とした後，カット $(v, \{v'\})$ から始めてカットの展開を繰り返すことで v のラベル・カットを列挙する．これは，$\exists i \in LEAF(C), L_K(i) = p$ であるようなカット C に対して，i を葉に含んだままカットの展開を繰り返すことを避ける処理である．図 1 を用い例を示す．ノード a のラベル・カットを列挙する場合に，a とラベル値が同じである b, c, e, f を用いたカットの展開，例えば $EXP(b, (a, \{b, c\}))$, $EXP(c, (a, \{b, c\}))$, $EXP(c, (a, \{c, d, e\}))$ などはスキップされ，カットの展開は $(a, \{d, h, i, j\})$ から始まる．図 1 には a のカットが 18 個あるが，a のラベル・カットのみを列挙するためには $(a, \{d, h, i, j\})$ および $EXP(d, (a, \{d, h, i, j\}))$ を調べるだけでよい．

$FTP(v)$ は $\{v\} \cup \Psi(\Phi_K(v_1)) \cup \Psi(\Phi_K(v_2))$ の部分集合であるため，提案するアルゴリズムは全てのラベル・カットを列挙することを保証していない．ノード v のラベル・カットが 1 つも見つからなかった場合，全ての葉がプライマリ・インプットになるかもしくは決まった数のラベル・カットが得られるまで，$FTP(v)$ による停止条件を用いずにカットの展開を再び実行する．この処理で，各ノードに少なくとも数個のラベル・カットが列挙されることを保証する．

4.3 提案手法で列挙したカットを用いて得られるネットワークの深さ

本節では，ラベル・カットのみを用いて得られる LUT ネットワークの深さが最小であることを示した後に，提案手法で得られるカットの集合を用いて深さ最小な被覆を行って得られる LUT ネットワークが深さ最小であることを示す．

［定理 3］ サブジェクト・グラフと自然数 K に対し，ラベル・カットからなる実現可能なカットの集合 S から導かれる LUT ネットワーク N は深さ最小である．

証明

$LEAF(C) \subseteq PI$ であるようなラベル・カット $C \in S$ において，$KFC(C)$ は明らかに $TFIG(RT(C))$ を深さ最小となるように被覆している．S における任意のラベル・カット C に対して，$CFI(C, S)$ の各ノード i における $TFIG(i)$ が深さ最小となるように被覆されていると仮定する．このとき，$TFIG(RT(C))$ を被覆する N の部分グラフもまた深さ最小である．

上記の 2 つの条件に基づいて，PI から PO へ向かってトポロジカルな順序で，$\exists C \in S, v = RT(C)$ であるようなサブジェクト・グラフの各ノード v に対する部分グラフ $TFIG(v)$ が深さ最小となるよう被覆されていることが確認できる．S における全てのカット C に対し，$TFIG(RT(C))$ を被覆する N の部分グラフは深さ最小である．従って，与えられたサブジェクト・グラフに対して N は深さ最小な LUT ネットワークである． □

提案手法は各ノードに対して少なくとも 1 つのラベル・カットを列挙する．従って，提案手法で列挙したカットの集合はラベル・カットのみからなる実現可能なカットの集合を少なくとも 1 つ包含する．ラベル・カットのみからなる実現可能なカットの集合から得られる LUT ネットワークは深さ最小であるため，提案手法で列挙したカットを組み合わせて得られる LUT ネットワークには深さ最小なものが含まれていることになる．従って，提案手法で得られるカットの集合を用いて深さ最小な被覆を行って得られる LUT ネットワークの深さは最小である．

4.4 エリア・カットの列挙

サブジェクト・グラフにおける 1 つのノードが複数の K フィージブル・コーンで被覆されることをノードの複製と呼ぶ．LUT ネットワークの深さを最小化するためにしばしばノードの複製が必要となる一方で，不必要なノードの複製は面積を増加させる．サブジェクト・グラフのうち深さに関してクリティカルでないノードについては，深さを最小化するために必ずしもラベル・カットを用いて被覆する必要はない．深さに関してクリティカルでないノードをラベル・カットで被覆することは不必要なノードの複製を発生させる可能性がある．エリア・カットの列挙の段階においては，面積の最小化に望ましいと考えられるカットをヒューリスティックな手法で列挙する．直感的に，多くのファンアウトを持つノードにおいてはノードの複製が発生しやすい．もし複数のファンアウトを持つノード v のファンアウトの全てが 1 つの K フィージブル・コーンで被覆されたなら，v においてはノードの複製は発生しない．一方で，もしファンアウトが複数の K フィージブル・コーンで被覆され，かついずれかのファンアウトが v を入力に持たない K フィージブル・コーンで被覆されていた場合，v ではノードの複製が起こる．本節において，葉の全てのノードが複数のファンアウトを持つカットをエリア・カットと呼ぶ．エリア・カット列挙の段階においては，全てまたは一部のエリア・カットが列挙される．

ノード v におけるエリア・カットの列挙は $(v, \{v\})$ から始まり，カットの展開が繰り返される．ファンアウトを 1 つしか持たないノードを葉に含むカットを除外するため，葉においてファンアウトを 1 つしか持たないノードは常に最優先で展開される．図 1 の例において，カットの展開は $(a, \{a\})$ で始められる．まず，$EXP(a, (a, \{a\}))$ により $(a, \{b, c\})$ が得られる．この場合，$|FO(b)| = 1$ および $|FO(c)| = 1$ なので b と c は優先的に展開され，$(a, \{d, e, f\})$ が得られる．ここで注意する点は，$(a, \{d, e, c\})$ や $(a, \{b, e, f\})$ は列挙されず，それらに対する展開も行われないことである．さらに，d や f もファンアウトを 1 つしか持たない．従って，d と f もまた優先的に展開され，$(a, \{g, h, e, i, j\})$ が得られる．葉の全てのノードが複数のファンアウトを持つ場合，カットの展開は 4.1 節で述べられた通りに繰り返される．ラベル・カットは既に列挙済みであるので，このカットの展開は葉における全てのノードが根より小さいラベルを持った時点で停止する．

5. 実　　験

本章では，既存のボトムアップな全列挙のアルゴリズム，トップダウンな全列挙のアルゴリズム，および提案するアルゴリズムを比較する実験について述べる．提案するアルゴリズム，ボトムアップな全列挙のアルゴリズムおよびトップダウンな全列挙のアルゴリズムは，C++を用いてプログラムとして実装された．LUT ネットワークを生成するために，サブジェクト・グラフおよび列挙したカットに対して深さ最小を保証する被覆アルゴリズム [9] が実行された．比較するアルゴリズムのいずれも深さ最小な被覆アルゴリズムと組み合わせることで LUT ネットワークの最小深さが保証されるので，各アルゴリズムに対して与えられるサブジェクト・グラフと K が等しければ，生成する LUT ネットワークの深さも等しい．各アルゴリズムを比較するために，全体の実行時間および LUT ネットワークの面積を評価した．ここで述べる全体の実行時間とは，カット列挙の実行時間と被覆の実行時間の和である(注5)．実験に用いたマシンの CPU は Intel Xeon 3.00GHz，メインメモリは 16GB である．サブジェクト・グラフは MCNC ベンチマークと ITC'99 ベンチマークの各ネットワークの各ノードを 2 入力以下のノードに分解することで得られた．また，K は 8 および 9 とした．

提案するアルゴリズム，ボトムアップな全列挙のアルゴリズムおよびトップダウンな全列挙のアルゴリズムの実行時間と面積における比較を表 1 に示す．規模が小さいベンチマークに対

(注5)：実際には，実験結果においてカット列挙の実行時間が全体の実行時間のほとんどを占める

表 1　実験結果における提案手法と全列挙手法の比較

Circuits	K=8								K=9							
	#LUT			Run Time (sec)					#LUT			Run Time (sec)				
	All	P	P/All	Bu	Td	P	P/Bu	P/Td	All	P	P/All	Bu	Td	P	P/Bu	P/Td
C3540	170	173	102%	55.3	19.0	5.5	10%	29%	159	160	101%	685.9	102.2	36.7	5%	36%
C5315	217	218	100%	230.6	53.9	20.0	9%	37%	205	205	100%	4014.3	252.2	92.9	2%	37%
C6288	261	261	100%	2825.0	524.2	317.7	11%	61%	247	247	100%	50943.9	3333.8	1852.4	4%	56%
C7552	334	334	100%	953.6	104.8	54.5	6%	52%	289	289	100%	20637.4	572.8	235.6	1%	41%
att15	243	248	102%	21.5	4.4	1.4	6%	31%	188	191	102%	169.5	14.1	3.7	2%	26%
att16	260	260	100%	7.3	2.7	1.6	22%	59%	235	235	100%	32.8	8.4	2.9	9%	35%
att21	1083	1087	100%	19.5	10.1	5.8	30%	57%	976	1006	103%	132.7	32.4	15.2	11%	47%
att22	157	156	99%	2.8	1.0	0.5	18%	51%	136	137	101%	18.1	2.6	1.0	6%	39%
att8	226	241	107%	2.5	1.2	0.7	27%	58%	187	196	105%	16.5	3.4	1.8	11%	53%
des	369	369	100%	363.6	67.8	34.6	10%	51%	550	550	100%	16674.2	529.0	197.2	1%	37%
rot	198	194	98%	4.4	2.9	1.5	34%	53%	160	161	101%	33.5	10.6	4.6	14%	43%
b12	203	203	100%	9.0	4.0	3.0	33%	74%	173	175	101%	101.9	16.8	9.7	10%	58%
b14	1115	1190	107%	633.1	271.9	142.7	23%	52%	999	1081	108%	9489.1	1444.1	660.0	7%	46%
b14_1	958	1007	105%	513.8	162.1	97.8	19%	60%	906	943	104%	6989.9	758.5	396.5	6%	52%
b15	1578	1675	106%	1717.9	625.4	297.4	17%	48%	1498	1558	104%	40604.3	4160.1	1764.6	4%	42%
b15_1	1603	1802	112%	10403.9	702.5	555.0	5%	79%	1459	1566	107%	134717.9	3404.2	2335.6	2%	69%
b17	5091	5553	109%	12073.4	1653.5	1017.7	8%	62%	4751	5018	106%	179990.2	9649.6	4751.0	3%	49%
b17_1	4949	5559	112%	31814.1	2291.7	1740.5	5%	76%	4552	4980	109%	417510.6	13702.9	5621.9	1%	41%
b20	2151	2352	109%	1445.5	579.9	326.2	23%	56%	1969	2170	110%	20569.8	3002.6	1376.6	7%	46%
b20_1	1894	2008	106%	1387.0	407.3	235.3	17%	58%	1743	1797	103%	20313.5	2044.4	973.4	5%	48%
b21	2253	2467	109%	1339.4	563.9	319.0	24%	57%	2032	2270	112%	18410.0	2894.3	1323.9	7%	46%
b21_1	1966	2098	107%	1592.9	457.2	267.7	17%	59%	1825	1912	105%	23659.5	2233.5	1122.3	5%	50%
b22	3415	3652	107%	2437.9	972.5	553.9	23%	57%	2989	3236	108%	37582.8	5272.4	2621.7	7%	50%
b22_1	2885	3050	106%	2245.4	656.6	382.0	17%	58%	2635	2705	103%	32448.2	3216.4	1577.7	5%	49%
AVERAGE			104%				17%	56%			104%				6%	46%

する実験結果は省略する．“Bu”，“Td”，および “P” の列はそれぞれボトムアップな全列挙のアルゴリズム，トップダウンな全列挙のアルゴリズム，および提案するアルゴリズムを指す．ボトムアップな全列挙のアルゴリズムとトップダウンな全列挙のアルゴリズムは得られるカットの集合が等しいため，被覆アルゴリズムが同じなら生成する LUT ネットワークも同じものである．従ってボトムアップな全列挙のアルゴリズムとトップダウンな全列挙のアルゴリズムの面積については，まとめて “All” の列で示されている．各アルゴリズムに対して同じサブジェクト・グラフおよび K を与えて得られた LUT ネットワークの深さは等しかった．提案手法はボトムアップな全列挙手法と比較して，$K = 8$ および 9 の場合にそれぞれ平均で 6 倍および 16 倍の早さでカットを列挙した．また，提案手法はトップダウンな全列挙手法と比較して，$K = 8$ および 9 のいずれもおよそ 2 倍の早さでカットを列挙した．提案手法で列挙したカットを用いて生成した LUT ネットワークの面積は，全てのカットを用いて生成した LUT ネットワークの面積よりもわずかに 4% ほど大きかった．トップダウンな全列挙手法は，実行時間の点でボトムアップな全列挙手法よりも優れている．しかし，サイズが大きいカットを列挙する必要があるアプリケーションに対して全列挙手法はあまり実用的ではないと考えられる．もし設計者がわずかな面積の増加を許容できるのであれば，提案手法を用いてネットワークの最小深さを保ったまま限定的な列挙を行うことによって，全列挙手法よりもさらに高速にカットを列挙することができる．

6. ま と め

本稿は，列挙されたカットを用いて深さ最小な被覆を行った場合の LUT ネットワークの最小深さを保証しつつ，限られたカットのみを列挙することによって高速にカット列挙を行うアルゴリズムを提案した．ラベル・カットのみからなるネットワークが深さ最小であることを証明し，各ノードにラベル・カットを少なくとも 1 つ含むカットの集合を用いて深さ最小な被覆を行って得られるネットワークが深さ最小であることを示した．提案手法は，カットの展開と FlowMap のラベル付けを組み合わせることによって，各ノードのラベル・カットを効率よく列挙する．また提案手法は，列挙するカットを限定することによる面積の増大を抑えるため，面積の最小化に有効と思われるカットを列挙する．実験の結果，提案手法はボトムアップな全列挙手法と比べ，K が 8 および 9 の場合にそれぞれ 6 倍および 16 倍の早さでカットを列挙した．また，提案手法はトップダウンな全列挙手法と比べ，K が 8 および 9 のどちらの場合にもおよそ 2 倍の早さでカットを列挙した．提案手法と全列挙手法のどちらもネットワークの最小深さは保証されるため，生成したネットワークの深さは各手法において等しかった．提案手法により生成したネットワークの面積は，全てのカットを用いて生成したものよりもわずかに 4% ほど大きかった．もしわずかな面積の増大が許容されるならば，提案手法を用いることで LUT ネットワークの最小深さを保証しつつ全列挙手法より短い実行時間でカットを列挙することが可能である．

謝　　辞

本研究の一部は，JST CREST-DVLSI の支援による．

文　　献

[1] Chatterjee, S., Mishchenko, A. and Brayton, R., “ Factor cuts,” *Proc. ICCAD ‘06*, pp. 143–150 (2006).

[2] Chen, D. and Cong, J., “DAOmap: A depth-optimal area optimization mapping algorithm for FPGA designs,” *Proc. ICCAD ‘04*, pp. 752–759 (2004).

[3] Cong, J. and Ding, Y., “FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs,” *IEEE Trans. CAD*, Vol. 13, pp. 1–12 (1994).

[4] Cong, J. and Ding, Y., “On area/depth trade-off in LUT-based FPGA technology mapping,” *IEEE Trans. on VLSI Systems*, Vol. 2, pp. 213–218 (1994).

[5] Cong, J., Wu, C. and Ding, Y., “Cut ranking and pruning: Enabling a general and efficient FPGA mapping solution,” *Proc. FPGA ‘99*, pp. 29–35 (1999).

[6] Cong, J. and yow Hwang, Y., “Simultaneous Depth and Area Minimization in LUT-Based FPGA Mapping,” *Proc. ACM 3rd Int’l Symp. on FPGA*, pp. 68–74 (1995).

[7] Mishchenko, A., Cho, S., Chatterjee, S. and Brayton, R., “Combinational and sequential mapping with priority cuts,” *Proc. ICCAD ‘07*, pp. 354–361 (2007).

[8] Pan, P. and Lin, C.-C., “A New Retiming-Based Technology Mapping Algorithm for LUT-based FPGAs,” *Proc. FPGA ‘98*, pp. 35–42 (1998).

[9] Takata, T. and Matsunaga, Y., “Area recovery under depth constraint by Cut Substitution for technology mapping for LUT-based FPGAs,” *Proc. ASP-DAC ‘08*, pp. 144–147 (2008).

[10] Teslenko, M. and Dubrova, E., “Hermes: LUT FPGA technology mapping algorithm for area minimization with optimum depth,” *Proc. ICCAD ‘04*, pp. 748–751 (2004).

[11] 松永 裕介，“FPGA 用テクノロジマッピングにおける効率的なカット列挙手法について,” 電子情報通信学会技術研究報告. VLD, VLSI 設計技術， Vol. 106, No. 453, pp. 49–54 (2007).