深さ最小かつLUTの信号遷移確率の総和極小なLUT型FPGA向けテクノロジ・マッピング

高田, 大河
九州大学大学院システム情報科学府

松永, 裕介
九州大学大学院システム情報科学研究院

出版情報：DAシンポジウム. 2008, pp.79-84, 2008-08-26
バージョン：
権利関係：

深さ最小かつLUTの信号遷移確率の総和極小なLUT型FPGA向けテクノロジ・マッピング

高田　大河†　　松永　裕介††

† 九州大学 大学院システム情報科学府 情報工学専攻
†† 九州大学 大学院システム情報科学研究院 情報工学部門
〒 819-0395 福岡県福岡市西区元岡 744
E-mail: †{taiga,matsunaga}@c.csce.kyushu-u.ac.jp

あらまし　深さ（最長パスの長さ）制約下で信号遷移確率の総和が小さいネットワークを生成するテクノロジ・マッピング問題はNP困難なクラスと同等に難しいと考えられ，ヒューリスティックな手法で解かざるを得ない．本稿では，深さ最小な初期解からLUTを取り除く反復改善によって局所的な最適解を生成するアルゴリズムを考える．

キーワード　EDA, FPGA, 論理合成，テクノロジ・マッピング，リシンセシス

Area Recovery under Depth Constraint by Cut Substitution for Technology Mapping for LUT-based FPGAs

Taiga TAKATA† and Yusuke MATSUNAGA††

† Graduate School of Information Science and Electrical Engineering, Kyushu University
†† Faculty of Information Science and Electrical Engineering, Kyushu University
744 Motooka, Nishiku, Fukuoka 819-0395 JAPAN
E-mail: †{taiga,matsunaga}@c.csce.kyushu-u.ac.jp

Abstract　The problem to generate a network comprised by LUTs whose total swithcing activity is minimum for technology mapping for LUT-based FPGAs is as difficult as NP-hard class problem, and the problem to generate a network under depth-minimum constraint seems to be difficult. So we must take a heuristic approach to generate a solution for this problem. In this paper, we study the algorithm to generate a local optimum solution by eliminating redundant LUTs while the depth of the network is maintained.

Key words　EDA, FPGA, Logic Synthesis, Technology Mapping, Resynthesis

1. はじめに

LUT(Look-Up Table) 型 FPGA(Field Programmable Gate Array) とは，チップの製造後にチップが実現する機能を変更することができる再構成可能デバイスの1つである．LUTとは決まった入力数以下の任意の論理関数を実現できる素子であり，LUT型FPGAは内部に持つ多数のLUTの論理関数とその接続関係を変更することでチップの機能を変更する．LUT型FPGAの問題点は，性能と消費電力である．

LUT型FPGAにおける回路合成の処理段階の1つに，テクノロジ・マッピングがある．LUT型FPGA向けテクノロジ・マッピングとは，論理関数を表わすネットワーク（サブジェクト・グラフ），LUTの最大入力数を入力として，LUTからなる論理的に等価なネットワークを生成する処理である．

LUT型FPGAの遅延は，LUT内部の遅延と配線部分の遅延からなる．LUT型FPGAにおいては配線部分の遅延が支配的であるため，個々のLUT内部の遅延を一定であると見なす．一方，LUT間の配線部分の遅延は，主にLUT間のバッファの負荷容量を充放電することによって生じる．テクノロジ・マッピングの段階では，各LUT間の配線部分に存在するバッファの数が未決定なため，LUT間の付加容量を正確に見積もることができない．そのため，テクノロジ・マッピング後の処理が理想的であると仮定して，各々のLUT間の配線に対し一定の固有遅延を仮定する．これらに基づき，LUTからなるネットワークの最長パスの長さで遅延を見積もる．LUTからなるネットワークの最長パスの長さのことを，

LUT からなるネットワークの深さと呼ぶ．

LUT 型 FPGA の消費電力は，各 LUT の信号遷移によって生じる消費電力の総和である．ある LUT の信号遷移によって生じる消費電力は，その LUT の信号遷移確率およびその LUT から他の LUT の入力までの経路上に存在する負荷容量の総和によって見積もることができる．LUT の入力における負荷容量と比べて LUT 間の配線上に存在するバッファの負荷容量が大きいが，テクノロジ・マッピングの段階では，各 LUT 間の配線部分に存在するバッファの数が未決定なため，LUT 間の付加容量を正確に見積もることができない．そのため，遅延の場合と同様，各 LUT 間の配線上に存在する付加容量を一定と見なす．これに基づいて，LUT 型 FPGA の消費電力を LUT の信号遷移確率の総和で見積もる．

深さ最小制約下で信号遷移確率の総和最小なネットワークを生成する LUT 型 FPGA 向けテクノロジ・マッピングは，深さ最小制約つき DAG カバリング問題 [1] として定式化できる．LUT 型 FPGA 向けテクノロジ・マッピングを DAG カバリング問題として定式化すると，サブジェクト・グラフをコストの総和が最小な LUT の集合で被覆する問題となる．サブジェクト・グラフがツリーであった場合は，Dynamic Programming により厳密解を保証することができる．具体的には，サブジェクト・グラフの部分グラフに対する最適解とそのコストの総和を用いて，それを包含するようなより大きい部分グラフの最適解を決定できる．しかし一般的なサブジェクト・グラフは Directed Acyclic Graph(DAG) であり，その場合の DAG カバリング問題は NP 困難なクラスに属することが知られている．従って，深さ最小制約つき DAG カバリング問題も難しい問題と考えられ，サイズが大きいサブジェクト・グラフに対して現実的な時間で解を求めるためにはヒューリスティックな手法を用いざるを得ない．DAG カバリング問題の難しさは，解を構築する途中でコストの総和を正確に見積もることが難しい点にある．解を構築する途中でコストの総和を正確に見積もることが難しいため，近年の関連研究においては，適当な初期解を生成した後に初期解の情報を用いた再マッピングを行ってコストの総和を減らそうとする手法が多くみられる [2] [3] [4] [5] [6] [7] [8].

深さ最小制約下で LUT 数最小なネットワークの生成を目的とした再マッピング手法の 1 つに，Cut Resubstitution [5] がある．Cut Resubstitution は，深さ最小な初期解に対して深さが増大しない範囲で LUT を取り除く処理を反復して行う．サブジェクト・グラフにおいて，1 つの LUT が被覆可能な部分グラフをカットと呼ぶ．ある解における LUT L のファンアウトの各 LUT が被覆するカットを，同じ論理を実現し L を入力としない他のカットに置き換える．それによって L の出力数は 0 となるため，ネットワークから L を取り除くことができる．LUT が被覆するカットを置き換えることによる LUT 数の増減は 0 であるため，カットの置き換えにより L を削除することで全体の LUT 数は 1 減る．

本稿では，深さ最小制約下で信号遷移確率の総和が小さいネットワークの生成を目的とした再マッピングへ Cut Resubstitution を応用することを考える．実現する論理が等しい LUT は，LUT の信号遷移確率も等しい．従って，LUT が被覆するカットを，同じ論理を実現するカットに置き換えることによる信号遷移確率の増減は 0 である．LUT L のファンアウトにおける LUT のカットの置き換えにより L を削除した場合，信号遷移確率の総和は L の信号遷移確率だけ減る．ある LUT が被覆するカット c を，同じ論理を実現し LUT L を入力としない他のカット c' に置き換えるためには，c' の入力にあたる各ノードが L 以外の LUT で被覆されている必要がある．そのようなカット c' が存在しない場合に，被覆されていないノードに対してそれを被覆する新たな LUT を生成することでカットの置き換えが可能になる場合がある．目的関数が LUT 数である場合は，新たな LUT を追加して他の LUT を削除したとしてもネットワーク全体の LUT 数の増減は 0 であるため，LUT を追加する処理は考えられていない．しかし目的関数が信号遷移確率である場合は，追加した LUT の信号遷移確率よりも削減した LUT の信号遷移確率が大きければネットワーク全体の信号遷移確率の総和は減少する．本研究では，LUT を追加しない Cut Resubstitution と LUT の追加を許す Cut Resubstitution について比較実験を行った．LUT を追加しない Cut Resubstitution が生成したネットワークに対して，LUT の追加を許す Cut Resubstitution が生成したネットワークにおける信号遷移確率の総和はほぼ同等であった．また，LUT の追加を許す Cut Resubstitution は LUT を追加しない Cut Resubstitution と比較して平均で 7 倍の実行時間を要することを確認した．

本論文は，以下の構成をとる．第 2 章において，問題の定義を行う．第 3 章において，深さ最小制約下で信号遷移確率の総和が小さいネットワークの生成を目的とした Cut Resubstitution の説明を行う．第 4 章において実験結果を示し，第 5 章においてまとめる．

2. 準　　備

テクノロジ・マッピングにおいて，与えられる入力はサブジェクト・グラフと呼ばれるグラフおよび LUT の最大入力数であり，出力は LUT ネットワークと呼ばれるグラフである．LUT の最大入力数を k と表すと，サブジェクト・グラフと LUT ネットワークは，各ノードが k 入力以下の DAG(Directed Acyclic Graph) である．DAG とは，ループを持たない有向グラフを指す．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しており，外部出力（プライマリ・アウトプット，以降 PO と略す）にあたるノードが指定されたものである．LUT ネットワークの各ノードは k 入力以下の LUT を表している．サブジェクト・グラフや LUT ネットワークにおいて，ノード i がノード j の入力である場合に有向辺 (i, j) が存在する．また本稿においては，サブジェクト・グラフの各ノードに対し，信号遷移確率を表す 0 から 1 までの範囲の実数が与えられるとする．

DAG $N(V,E)$ において，ノード v のファンインとは v の入力辺に接続するノードを指す．v のファンインの集合は $FI(v) = \{u \in V | (u,v) \in E\}$ として定義される．また，v のファンアウトは v の出力辺に接続するノードを指す．v のファンアウトの集合は $FO(v) = \{w \in V | (v,w) \in E\}$ で定義される．$FI(v) = \phi$ であるようなノード v をプライマリ・インプット (以降 PI と略す) と呼ぶ．DAG N の PI の集合を $PI(N)$, PO の集合を $PO(N)$ と表す．ノード v のトランシティブ・ファンイン (以降 TFI と呼ぶ) とは，全ての PI から v までのパスに含まれるノードの集合である．より正確には，v のトランシティブファンイン $TFI(v)$ は以下の式で定義される．

$$TFI(v) = \{v\} \cup \bigcup_{u \in FI(v)} TFI(u)$$

ノード v のトランシティブ･ファンイン･グラフとは，$TFI(v)$ によって導かれる N の 誘導部分グラフのことであり，$TFIG(v)$ と表す．ノード v の段数とは，全ての PI から v までのパスのうち最長なパスの長さである．また，DAG の深さは，DAG における全てのパスのうち最長なパスの長さであり，全てのノードの段数の最大値に等しい．

ノード v を根とするカットとは，式 (1) の条件を満たす $TFI(v)$ の分割のことであり，$(X, \overline{X})$ と表される．$(X, \overline{X})$ が $TFI(v)$ の分割なので，$X \cup \overline{X} = TFI(v), X \cap \overline{X} = \phi$ である．

$$\forall i \in PI(TFI(v)), i \in X, v \in \overline{X} \qquad (1)$$

カット $(X, \overline{X})$ に対し，カット $(X, \overline{X})$ の根を $ROOT((X, \overline{X}))$ で表す．カット $(X, \overline{X})$ に対し，カットの境界部分における X に含まれるノードをカットの葉と呼ぶ．より厳密には，カットの葉の集合 $LEAF((X, \overline{X}))$ を以下のように定義し，$LEAF((X, \overline{X}))$ の要素をカットの葉と呼ぶ．

$$LEAF((X, \overline{X})) = \{i | i \in X, \exists j \in \overline{X}, (i,j) \in E\}$$

v を根とするカット $(X, \overline{X})$ が以下の条件を満たすとき，$(X, \overline{X})$ を v を根とする k フィージブル・カットと呼ぶ．

- $\overline{X}$ の任意のノード i に対し，$\overline{X}$ のノードのみからなる i から v までのパスが少なくとも 1 つ存在する．
- $\overline{X}$ の任意のノード i, X の任意のノード j に対し，辺 (i,j) が存在しない．つまり $\forall i \in \overline{X}, \forall j \in X, (i,j) \notin E$.
- 葉の要素数が k 以下である．つまり $|LEAR((X, \overline{X}))| \leqq k$

ノード v を根とする k フィージブル・カット $(X, \overline{X})$ に対し，$\overline{X}$ から導かれる誘導部分グラフを k フィージブル・コーン（以下 KFC と略す）と呼ぶ．任意の KFC に対応する論理関数は 1 つの k 入力以下 LUT で実現可能である．あるサブジェクト・グラフ $S(V,E)$ における k フィージブル・カットの集合 C' に対し，C' の各要素に対応する KFC をそれぞれ LUT で実現した結果が LUT ネットワークとして実現可能であるためには，C' が以下の 2 つの条件を満たしていなければならない．

（1） $\forall c \in C', \forall i \in LEAF(c),$
$(\exists c' \in C', (i = ROOT(c')) \vee (i \in PI))$

（2） $\forall i \in PO(S), (\exists c \in C, i = ROOT(c), i \in PI(S))$

上記の条件を満たす k フィージブル・カットの集合を，フィージブルな k フィージブル・カットの集合であるという．フィージブルな k フィージブル・カットの集合 C および $c \in C$ に対して，カット・ファンイン $CFI(c,C)$ およびカット・ファンアウト $CFO(c,C)$ を以下のように定義する．

$$CFI(c,C) = \{c' | c' \in C, ROOT(c') \in LEAF(c)\}$$
$$CFO(c,C) = \{c' | c' \in C, ROOT(c) \in LEAF(c')\}$$

フィージブルな k フィージブル・カットの集合 C から，各要素に対応する LUT を生成し，c に対応する LUT a と $CFI(c,C)$ の各要素に対応する LUT b 間へ辺 (a,b), $CFO(c,C)$ の各要素に対応する LUT c 間へ辺 (c,a) を生成することでフィージブルな k フィージブル・カットの集合 C に対応する LUT ネットワークを得ることができる．LUT 型 FPGA 向けテクノロジ・マッピングの問題は，与えられたサブジェクト・グラフにおける k フィージブル・カットの集合 C から，フィージブルな C の部分集合を求める問題であると見なすことができる．また，フィージブルな k フィージブル・カットの集合 C および $c \in C$ に対して，c の段数および要求段数を定義する．レベル $LEV(c,C)$ は，C に対応する LUT ネットワークにおける，c に対応する LUT の段数を表しており，以下のように計算できる．

$$LEV(c,C) = \begin{cases} \max_{c' \in CFI(c,C)} LEV(c',C) + 1 & (CFI(c,C) \neq \phi) \\ 1 & (otherwise) \end{cases}$$

サブジェクト・グラフにおけるノード v の信号遷移確率を $ACT(v)$ と表し，v を根とする k フィージブル・カット c のコストを $COST(c) = ACT(v)$ とする．$COST(c)$ は，c に対応する LUT の信号遷移確率を表わしている．このとき，深さ最小制約下で信号遷移確率の総和が最小な LUT ネットワークを求めるテクノロジ･マッピング問題は，次のようになる．サブジェクト・グラフ $S(V,E)$, PO の集合 $PO \in V$, LUT の最大入力数 k, $\forall v \in V$ に対する信号遷移確率 $ACT(v)$ が与えられたときに，$\max_{c \in C'} LEV(c,C')$ が最小であるフィージブルな k フィージブル・カットの集合のうち $\sum_{c \in C'} COST(c)$ が最小であるフィージブルな k フィージブル・カットの集合 C' を求める．

3. 深さ最小かつ信号遷移確率の総和極小解を生成する Cut Resubstitution

3.1 LUT を生成しない Cut Resubstitution

深さ最小かつ信号遷移確率の総和最小なネットワークの生成を目的とした Cut Resubstitution は，深さ最小な初期解に対して k フィージブル・カットを削除する反復処理により局所最適解を生成する手法である．Cut Substitution は具体的には，サブジェクト・グラフ $S(V,E)$，全ての k

フィージブル・カットの集合 C_{all} およびフィージブルな k フィージブル・カットの集合 C_0 を入力として次の処理を反復する．i 番目の反復において，フィージブルな k フィージブル・カットの集合 C_i から $\max_{c\in C_{i+1}} LEV(c, C_{i+1}) = \max_{c\in C_i} LEV(c, C_i)$, $\cup_{c\in C_i} ROOT(c) \subseteqq \cup_{c\in C_{i+1}} ROOT(c)$ の制約を満たし $\sum_{c\in C_{i+1}} COST(c) \prec \sum_{c\in C_{i+1}} COST(c)$ であるフィージブルな k フィージブル・カットの集合 C_{i+1} を求める．

LUT の生成を行わない Cut Resubstitution は，Substitutional Cut Enumeration, Gain Calculation, Cut Elimination の 3 つの処理段階を逐次的に繰り返し行う処理である．

3.1.1 Substitutional Cut Enumeration

Substitutional Cut Enumeration は C_{all}, C_i を入力とし，C_i の全ての要素に対して置き換え候補となる k フィージブル・カットの集合 $C_{rsb} \in C_{all}$ を列挙する．k フィージブル・カット $c \in C_i$ に対する置き換え候補となる k フィージブル・カット $c' \in C_{all}$ とは，次の条件を満たすものである．

$$\forall i \in LEAF(c'), \exists c'' \in C_i, ROOT(c'') = i$$
$$ROOT(c) = ROOT(c')$$

3.1.2 Gain Calculation

フィージブルな k フィージブル・カットの集合 C_i における要素 c を C_i から取り除いた場合，$CFO(c', C_i) = \{c\}$ であるような c' も同時に C_i から取り除くことができる．Gain Calculation においては，C_i の各要素 c に対し，c を削除することで減る信号遷移確率の予測値（ゲインと呼ぶ）を計算して配列 $gain$ に保持する．その際に，c を削除することで同時に取り除くことができると考えられる k フィージブル・カットを考慮する．

k フィージブル・カット c のゲイン $GAIN(c)$ は，以下のように計算する．

$$GAIN(c) = \sum_{j\in CFI(c,C_i)} GAIN'(j) + COST(c)$$
$$GAIN'(j) = \begin{cases} GAIN(j) & (CFO(j, C_i) = \{c\}) \\ 0 & (otherwise) \end{cases} \quad (2)$$

3.1.3 Cut Elimination

Cut Elimination において，C_i の要素のうちゲインが大きい k フィージブル・カットから順に，削除が可能かどうかを調べる．k フィージブル・カット c が削除可能である条件は，最小深さを D とおくと以下の通りである．

- $CFO(c, C_i)$ の全ての要素 i に対し，c の根を葉に持たず，i と同じ根を持つ C_{rsb} の要素が存在する．すなわち，$\forall j \in CFO(c, C_i), \exists c' \in C_{rsb}, (ROOT(j) = ROOT(c')) \wedge (ROOT(c) \notin LEAF(j))$
- C_i における $CFO(c, C_i)$ の各要素に対し，同じ根を持つ C_{rsb} の要素の組合せ C' を考えた場合に，C_i における $CFO(c, C_i)$ を C' で置き換えた結果の k フィージブル・カットの集合 C'' に対し，$\max_{c\in C''} LEV(c, C'') = D$ となる C' が存在する

k フィージブル・カット c が削除可能であれば，$CFO(c, C_i)$ の k フィージブル・カットを C_{rsb} における他の k フィージブル・カットと置き換えて c を C_i から削除する．もし削除可能な k フィージブル・カットがなかった場合は C_i を出力する．

```
Cut Substitution( S(V,E), C_all, C_0 ){
  C_i := C_0;
  while (1) {
      C'_rsb := SubstCutEnum(C_all, C_i);
      BestResubCalc( S(V,E), C'_rsb, C_i );
      C := CutElim( C_i, C'_rsb );
      if (C = C_i) {
          break;
      }
      C_i := C;
  }
  return C;
}
```

図 1　LUT の生成を許す Cut Reubstitution の疑似コード

3.2 LUT の生成を許す Cut Resubstitution

LUT の生成を許す Cut Resubstitution は，i 番目の反復において，フィージブルな k フィージブル・カットの集合 C_i から $\max_{c\in C_{i+1}} LEV(c, C_{i+1}) = \max_{c\in C_i} LEV(c, C_i)$, $|\cup_{c\in C_i} ROOT(c) - \cup_{c\in C_{i+1}} ROOT(c)| \leqq 1|$ の制約を満たし $\sum_{c\in C_{i+1}} COST(c) \prec \sum_{c\in C_{i+1}} COST(c)$ であるフィージブルな k フィージブル・カットの集合 C_{i+1} を求めるものである．

LUT の生成を許す Cut Resubstitution の疑似コードは，図 1 である．$S(V, E)$ はサブジェクト・グラフ，C_{all} は全ての k フィージブル・カットの集合，C_0 はフィージブルな k フィージブル・カットの集合を表わす．C, C_i, C_{rsb} は k フィージブル・カットの集合を表わす．図 1 における SubstCutEnum, BestResubCalc, CutElim はそれぞれ Substitutional Cut Enumeration, Best Resubstitute Calculation, Cut Elimination の 3 つの処理段階を表わす．

3.2.1 Substitutional Cut Enumeration

Substitutional Cut Enumeration は C_{all}, C_i を入力とし，C_i の全ての要素に対して置き換え候補となる k フィージブル・カットの集合 $C_{rsb} \in C_{all}$ を列挙する．ここで列挙する k フィージブル・カットの集合は 3.1.1 節で列挙した C_{rsb} に加え，葉のノードのうち 1 つのみが C_i の要素の根となっていない k フィージブル・カットを含むものである．すなわち，C_i における各 k フィージブル・カット c に対し，3.1.1 における条件を満たす k フィージブル・カットに加えて，以下の条件を満たす k フィージブル・カット $c' \in C_{all}$ を C_{rsb} として保持する．

$$ROOT(c) = ROOT(c')$$
$$\exists n \in LEAF(c'), (\forall \imath \in LEAF(c') - \{n\},$$
$$\exists c'' \in C_i, ROOT(c'') = i) \wedge (\forall j \in C_i, n \neq ROOT(j))$$

3.2.2 Best Resubtitutie Calculation

Best Resubstitute Calculation は，全ての $c \in C_i$ に対して，c が削除可能かどうかを調べ，削除可能な k フィージブル・カットのうち実際に削除した場合に最もゲインが大きいと予測されるものを保持する．c が削除可能であるためにはまず，$CFO(c, C_i)$ の各要素 i に対し，c の根を葉に持たず i と同じ根を持つ置き換え候補の k フィージブル・カットが存在しなければならない．すなわち $\forall j \in CFO(c, C_i), \exists c' \in C_{rsb}, (ROOT(j) = ROOT(c')) \wedge (ROOT(c) \notin LEAF(c'))$ が成り立たなければならない．置き換え候補の k フィージブル・カット $a \in C_{rsb}$ の $LEAF(a)$ の要素 n のうち，$\forall j \in C_i, n \neq ROOT(j)$ を満たすノードの集合を $NEW(a)$ で表す．$NEW(a)$ の要素数は 0 または 1 である．C_i における $CFO(c, C_i)$ の各要素に対して，同じ根を持つ C_{rsb} の要素の組合せ C' を考える．C_i における $CFO(c, C_i)$ を C' で置き換えた結果の集合 C'' に対して，C'' をフィージブルにするために加えなければならない k フィージブル・カットが高々1 つであるための条件として，$|\cup_{j \in C'} NEW(j)| \leqq 1$ がある．$|\cup_{j \in C'} NEW(j)| = 0$ である C' において，C_i における $CFO(c, C_i)$ を C' で置き換えた結果の k フィージブル・カットの集合 C'' に対し，$\max_{j \in C''} LEV(j, C'') = D$ であれば c は削除可能である．また，$|\cup_{j \in C'} NEW(j)| = 1$ である C' において，$\cup_{j \in C'} NEW(j) = \{w\}$ とする．C_{all} の要素 i のうち $(ROOT(i) = w) \wedge (ROOT(c) \notin LEAF(i)) \wedge (\forall i \in LEAF(i), \exists c' \in C_i, ROOT(c) \in LEAF(i)$ である k フィージブル・カットの集合を C_w とする．また，C_i のうち，ノード v を根とする k フィージブル・カットを $COVER(v, C_i)$ と表す．C_w の要素 i のうち，$max_{j \in LEAF(i)} LEV(COVER(j, C_i)) + 1$ の値が最小の k フィージブル・カット c_w を保持する．c_w は，C' に追加することで C' をフィージブルにする k フィージブル・カットのうち，w を被覆する最小段数の k フィージブル・カットである．C_i における $CFO(c, C_i)$ を C' で置き換え，c_w を加えて得られる k フィージブル・カットの集合 C'' に対し，$\max_{j \in C''} LEV(j, C'') = D$ であれば c は削除可能である．

削除可能な $c \in C_i$, c を削除するための置き換え候補 C'，新たに追加する k フィージブル・カット c_w を保持し，c と c_w の組に対するゲインを計算する．(c, c_w) に対するゲイン $GAIN(c)$ は，以下の式で求める．以下の式中の $GAIN'(j)$ は，式 (2) で求める．

$$GAIN(c, c_w) = \begin{cases} \sum_{j \in CFI(c, C_i)} GAIN'(j) + COST(c) \\ \qquad (c_w = NULL) \\ \sum_{j \in CFI(c, C_i)} GAIN'(j) + COST(c) \\ \qquad -COST(c_w) \\ \qquad (otherwise) \end{cases}$$

3.2.3 Cut Elimination

ゲインが最大の $c \in C_i$ および c を削除するための置き換え候補 C'，新たに追加する k フィージブル・カットの組合せを用いて，k フィージブル・カットの置き換えを実行する．C_i における $CFO(c, C_i)$ を C_i から取り除き，C' を C_i へ加える．さらに c_w を C_i へ追加して c を C_i から削除し，その結果生成される k フィージブル・カットの集合を出力する．もし削除可能な k フィージブル・カットがない場合は処理を終了する．

表 1　ITC'99 ベンチマークセットにおける実験結果

	WD	WD & CS		WD & CS+	
	sw	sw(vs WD)	time	sw(vs WD)	time
b05	82.70	80%	50.26	80%	51.77
b12	100.90	94%	0.28	94%	0.37
b14	822.22	88%	6.33	88%	21.04
b14_1	627.35	88%	3.63	88%	12.23
b15	708.09	82%	6.06	84%	27.84
b15_1	541.95	88%	9.13	88%	41.50
b17	1776.61	81%	58.46	83%	754.81
b17_1	1503.99	86%	79.16	87%	736.90
b20	1728.12	87%	20.52	87%	95.40
b20_1	1279.56	87%	11.84	87%	48.72
b21	1812.66	88%	21.13	88%	93.93
b21_1	1307.57	88%	12.11	88%	52.90
b22	2512.57	87%	43.31	87%	208.08
b22_1	1939.61	87%	23.45	87%	110.84

4. 実　　験

従来の Cut Resubstitution, および新たな LUT の生成を許す Cut Resubstitution のアルゴリズムを，九州大学の論理合成システム Magus にプログラムとして組み込み，両者の解および実行時間を比較する実験を行った．両者は，深さ最小制約下で信号遷移確率の総和最小なネットワークの生成を目的とした LUT 型 FPGA 向けテクノロジ・マッピングのプログラム WeightedDmap の後処理として組み込まれた．WeightedDmap は，深さ制約つき DAG カバリング問題に対するヒューリスティックなアルゴリズムの実装であり，文献 [9] における近似アルゴリズムの応用である．用いた言語は C++である．用いたマシンの CPU は IntelXeon 3.0GHz, メインメモリは 16GB である．また，本実験における LUT の入力数制約は 5 とした．

表 1 に実験結果を示す．表 1 は ITC'99 ベンチマークセットの各ベンチマークに対し，WeightedDmap，WeightedDmap と既存の Cut Resubstitution の組合せ，WeightedDmap と新たな LUT の生成を許す Cut Resubstitution の組合せが生成した LUT ネットワークの信号遷移確率の総和と実行時間の比較である．ITC'99 ベンチマークセットのうち規模が小さいベンチマークは省略した．"WD" は WeightedDmap, "WD&CS" は WeightedDmap と従来の Cut Resubstitution の組合せ，"WD&CS+" は WeightedDmap と新たな LUT の生成を許す Cut Resubstitution の組合せを表わす．"sw" の列は生成した LUT ネットワークの信号遷移確率の総和，"sw(vs WD)" は生成した LUT ネットワークの信号遷移確率の総和を WeightedDmap が生成した信号遷移確率の総和

で割ったもの，“time” は秒単位の実行時間である．ベンチマークが同じであれば LUT ネットワークの深さはいずれも等しいため，LUT ネットワークの深さは省略した．実験の結果，従来の Cut Resubstitution と新たな LUT を追加する Cut Resubstitution の信号遷移確率の総和はほとんど同等であった．両者とも，WeightedDmap の解の信号遷移確率を平均で 17%削減した．LUT の生成を許す Cut Resubstitution は，LUT の生成を許さない Cut Resubstitution の 7 倍ほどの実行時間を要した．

従来の Cut Resubstitution よりも広い解空間を探索するにも関わらず，新しい LUT の生成を許す Cut Resubstitution の解の信号遷移確率の総和は従来の Cut Resubstitution の解の信号遷移確率の総和とほぼ同等であった．これは，新しい LUT の生成を許す Cut Resubstitution において，実際に新しい LUT を生成して LUT を削除する処理がほとんど行われなかったためである．新しい LUT を生成して信号遷移確率が大きい LUT を削除する処理がほとんど行われなかった原因としては，以下の 2 つが考えられる．

- LUT を生成して他の LUT を削減することで信号遷移確率の総和が下がる場合がほとんど存在しない．
- LUT を削減した場合に得られる信号遷移確率の減少量の見積もり（ゲイン）が正確でない．

ゲインを計算する際に，削減する LUT l のファンインのうちファンアウトが l のみの LUT i は l が削除された場合ともに削除できると考えて，l のゲインとして l の信号遷移確率に i のゲインを加えている．しかし実際には，l のファンアウトの LUT が置き換わった後に i を新たなファンインとしている場合がある．上記のような場合，l のゲインには i のゲインの量だけ誤差が生じている．上記のような誤差を含まないゲインは本稿において用いたゲインより正確であるため，そのようなゲインを用いて Cut Resubstitution を行うことで良い解を生成できる可能性がある．

LUT の生成を許す Cut Resubstitution は，LUT の生成を許さない Cut Resubstitution の 7 倍ほど実行時間を要している．LUT の生成を許す Cut Resubstitution の実行時間には，改善の余地が存在する．LUT の生成を許す Cut Resubstitution は，LUT の削除を一度行うごとに置き換え候補の k フィージブル・カットの列挙をやり直す．1 度の LUT の削除に伴うネットワークの変化は小さいので，LUT を削除するたびに k フィージブル・カットの置き換え候補を列挙することは無駄が大きいと考えられる．実際には，LUT を削除するたびに置き換え候補の k フィージブル・カットに関して差分だけを更新することでより実行時間を短く抑えられる可能性がある．

5. 終わりに

本稿では，深さ最小制約下で LUT の信号遷移確率の総和が小さいネットワークを生成する LUT 型 FPGA 向けテクノロジ・マッピング問題の解法として，Cut Resubstitution を応用することを検討した．k フィージブル・カットの置き換えにより信号遷移確率の総和を LUT の生成を許し探索する探索空間を広げることで, さらなる信号遷移確率の総和の削減をねらう処理を検討した．実験の結果，LUT を生成しない Cut Resubstitution による解に対し，LUT の生成を許した Cut Resubstitution による解における信号遷移確率の総和の改善は見られなかった．また，新たな LUT の生成を許して探索空間を広げたことにより，従来の Cut Resubstitution と比較して平均で 7 倍の実行時間を要した．解の質が同等かつ実行時間が短いので，LUT の生成を行わない Cut Resubstitution が LUT の生成を行う Cut Resubstitution よりも優位なアルゴリズムであることが確認できた．

本稿で述べた手法はどちらも，初期解の信号遷移確率をおよそ 17%削減できている．今後の課題として，本稿で述べた手法が生成したネットワークと既存手法が生成したネットワークの質を比較することが挙げられる．今後は，本手法および既存手法が生成したネットワークに対してパッキングおよび配置配線を行い，配置配線結果の遅延および消費電力見積もり値を確認する予定である．

謝　　辞

本論文の一部は，平成 20 年度科学研究費補助金 (基盤研究 (B))(課題番号:20300020) による．

文　　献

[1] Richard L.Rudell, “Logic synthesis for VLSI design,” Ph.D.thesis, University of California, Berkeley, 1989.

[2] J.Cong, Y.Ding, “On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping,” IEEE Transactions on Very Large Scale Integration Systems, June, 1994.

[3] D.Chen, J.Cong, “DAOmap: A Depth-optimal Area Optimization Mapping Algorithm for FPGA Designs,” IEEE International Conference on Computer Aided Design, 2004.

[4] Maxim Teslenko, Elena Dubrova, “Hermes: LUT FPGA Technology Mapping Algorithm for Area Minimization with Optimal Depth,” IEEE International Conference on Computer Aided Design, 2004.

[5] Taiga Takata, Yusuke Matsunaga, “Area Recovery under Depth Constraint by Cut Substitution for Technology Mapping for LUT-Based FPGAs,” Asia and South Pacific Design Automation Conference, Feb, 2008h.

[6] Andrew Ling, Deshanand P. Singh, Stephen D. Brown, “FPGA Technology MApping: A Study of Optimality,” Design Automation Conference, 2005.

[7] Sean Safarpour, Andreas Veneris, Gregg Baeckler, Richard Yuan, “Efficient SAT-based Boolean Matching for FPGA Technology Mapping,” Design Automation Conference, 2006.

[8] Yu Hu, Victor Shih, Rupak Majumdar, Lei He, “FPGA Area Reduction by Multi-Output Function Based Sequential Resynthesis,” Design Automation Conference, 2008.

[9] 松永 裕介, “DAG カバリング問題の下限とそれを用いた厳密アルゴリズムについて”, 信学技報, 第 VLD2007-8 巻, 2007.