

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

松永 裕介†

†九州大学大学院システム情報科学研究院

〒819-0395 福岡市西区元岡 744

E-mail: †matsunaga@slrc.kyushu-u.ac.jp

あらまし 本稿ではカット列挙を行う効率的なアルゴリズムの提案を行う。他の多くの既存アルゴリズムと異なり、本アルゴリズムはトップダウン型の列挙アルゴリズムに基づいている。カット展開を行う際の終了条件を適切に定めて探索範囲を狭めることでトップダウン型の処理を高速に行っている。実験結果より、本アルゴリズムがほぼ列挙されるカット数に比例した時間で処理を行っていることが示されている。また、実験結果より既存のボトムアップ型アルゴリズムはカット数に比例した時間では処理を行えず、カット列挙の際にファンインのカット数の積に比例した時間を必要としていることも明らかになった。通常、ファンインのカット数の積はオーダーとしてカット数より大きい。そのため、既存のボトムアップアルゴリズムに対して本提案アルゴリズムは計算時間の点で優位であると言える。また、ボトムアップアルゴリズムは列挙したカットを保持しておかなければならないのに対して提案アルゴリズムでは保持する必要がないため、使用メモリ量的にも優位である。

キーワード テクノロジマッピング, FPGA, カット列挙

On Efficient Cut Enumeration in technology mapping for FPGA

Yusuke MATSUNAGA†

† Faculty of Information Science and Electorical Engineering, Graduate School of Kyushu University

744 Motooka, Nishi-ku, Fukuoka, 819-0395 Japan

E-mail: †matsunaga@slrc.kyushu-u.ac.jp

Abstract This paper presents a novel efficient algorithm for cut enumeration. Unlike other existing algorithm, the proposed algorithm is based on the top-down algorithm. Strictly Limiting the search space with the proposed idea of terminate condition of cut expansion makes the top-down algorithm efficient. The experimental results show that the proposed algorithm runs in almost linear time to the number of enumerated cuts. The experimental results also show that the bottom-up algorithm does not run in linear time to the number of enumerated cuts but runs in linear time to the product size of cut merging, which is much larger than the number of enumerated cuts in general.

Key words technology mapping, FPGA, cut enumeration

1. はじめに

2. Introduction

DAG(非巡回有向グラフ)におけるカット列挙は、LUT型FPGA用テクノロジマッピング[1], [2]や書き換え型論理最適化[3]などいくつかの論理合成アルゴリズムで用いられている。カットのサイズが小さい場合(サイズが5~6以下程度)にはカットを全て列挙することは極めて容易であるが、それ以上のサイズの場合(7もしくは8以上)には全てのカットを列挙する

ことが途端に難しくなってくる。そのひとつの理由はカットサイズの増加に対して列挙されるカット数が指数関数的に増加することであるが、より深刻な問題は既存の手法がその(指数関数的に増加する)カット数に対して線形時間でカットの列挙を行えない所にある。Chatterjee, MischenkoとBraytonはテクノロジマッピングの結果の質を損なうことなく考慮すべきカット数を大幅に減らすヒューリスティックを提案している[2]が^(注1),

(注1): 彼らはその文献中でテクノロジマッピングの結果のうち回路の段数のみを報告しており面積については言及していない。回路の段数を最小にするだけで

彼らのヒューリスティックはカット数に対して線形時間でカットの列挙を行うわけではない。これは、彼らのアルゴリズム(だけでなく他の既存アルゴリズムの多くも)がボトムアップ型のアルゴリズムに基づいているからである。ボトムアップ型のアルゴリズムでは、各ノードにおいて、そのファンインにおけるカットの集合の直積を計算している。しかし、一般には(より具体的には、グラフ全体が木構造でない限りは)、最終的に得られるカット数はファンインのカット集合の直積集合のサイズよりもはるかに小さい。これは2つのカットを組み合わせた結果が不適切なカットになったり冗長なカットになる場合が多いからである。いったん、直積集合を生成してから適切なカットを効率的に見つけ出すアルゴリズムは知られているが、その場合も最終的なカット数ではなく、直積集合のサイズに比例した手間を必要とする。この性質がボトムアップ型アルゴリズムの効率に対する大きな欠点となっている。

一方、トップダウン型のアルゴリズムはそのような処理を必要としない。この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スタートして、既存のカットの一部を入力方向に展開することで新しいカットを列挙してゆく。そのため、一見、無駄な処理がないのでボトムアップ型のアルゴリズムよりも効率がよいように思われるが、ひとつだけ大きな欠点を持っている— カットの展開をいつ止めれば良いかわからないのである。これは、サイズの制限を超えたカットの一部を展開することでサイズの制限を満たしたカットを得ることができる場合があるためである。そのため、単純には外部入力に到達するまでカットの展開を行わなければならない、回路規模の増大に対して非効率である。このような理由からトップダウン型のアルゴリズムは過去の文献においてはほとんど使用されていなかった。

本稿では、トップダウン型アルゴリズムにおいてカット展開を行う探索空間をタイトに狭める手法を新たに提案する。このアイデアによってトップダウン型のアルゴリズムはボトムアップ型のアルゴリズムよりもはるかに効率よくカットの列挙を行うことが可能となっている。実験結果によれば、提案手法は列挙されるカット数に対してほぼ線形な時間で処理を行っている。また、実験結果によりボトムアップ型の列挙アルゴリズムの時間複雑度は列挙されるカット数ではなく、ファンインのカット集合の直積集合のサイズに支配されることも示されている。

以降、第2章で基本的な定義や定理を示し、第3章でトップダウンアルゴリズムを効率的にするためのアイデアについて述べる。その後、第4章で実験結果およびその考察を述べ、第5章でまとめを行う。

3. 基本的な定義

サブジェクトグラフとはマッピングや最適化の対象となっている回路の構造を表す DAG(directed acyclic graph: 非巡回有

向グラフ)であり、ノード $v \in V$ および枝 $e \in E$ を持つ。ノードは回路要素に対応し、枝は回路要素間の接続を表している。ノード v のファンインはノード v の入力元となっているノードの集合であり $FI(v)$ で表される。ノード v のファンアウトとはノード v が出力しているノードの集合であり $FO(v)$ で表される。外部入力とはファンインを持たないノードの集合で PI で表される。外部出力とはファンアウトを持たないノードの集合で PO で表される。ノード v の推移的ファンインとは次式で定義されるノードの集合であり、 $TFI(v)$ で表される。nodes defined by the following equation.

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

ノード v の推移的ファンインサブグラフとは $TFI(v)$ から誘導される部分グラフのことで $G(v)$ で表される。

一般には、サブジェクトグラフのノードのファンイン数に対する制約はないが^(注2)、本稿では、説明を簡単にするため各ノードのファンイン数を2と定めている。以下の議論は、ファンイン数が2以上のより一般的な場合に対しても容易に拡張することが可能である。

ノード v の推移的ファンインサブグラフ $G(v)$ に対する極小セパレータをノード v のカットと呼ぶ。つまり、 PI の要素から v へいたる全ての経路が必ずカット c の要素を含んでおり(セパレータ)、また、各々のカットは他のカットを含んでいない(極小性)。ノード v に対して、そのノード自身のみからなる集合 $\{v\}$ もノード v のカットである。カットのサイズとはカットに含まれるノード数である。そのサイズが k 以下のカットを k -フィージブルカットと呼ぶ。ノード v の k -フィージブルカットを $\Phi_k(v)$ で表す。

3.1 ボトムアップ型アルゴリズム

2つのカット集合 A と B に対するマージ操作、 $A \bowtie_k B$ を以下のように定義する[2]。

$$A \bowtie_k B = \{u \cup v \mid u \in A, v \in B, |u \cup v| \leq k\} \quad (1)$$

[定理1] ノード v の k -フィージブルカットは以下のように計算される[2]。

$$\Phi_k(v) = \begin{cases} \{\{v\}\} & v \in PI \\ \{\{v\}\} \cup (\Phi_k(u_1) \bowtie_k \Phi_k(u_2)) & \text{otherwise} \end{cases} \quad (2)$$

ただし、 u_1 と u_2 は v のファンインとする。

ボトムアップ型カット列挙アルゴリズムはこの定理1の式を直接用いることで容易に構築できる[1], [2]。ただし、問題はこのマージ操作 $\Phi_k(u_1) \bowtie_k \Phi_k(u_2)$ に存在する。今、 $M = |\Phi_k(u_1)|$ 、 $N = |\Phi_k(u_2)|$ と定めるものとする。マージ操作の時間複雑度は $O(M \times N)$ である^(注3)。もしも結果として列挙されるカット $\Phi_k(v)$ のサイズも $O(M \times N)$ であるならこの計算複雑度は妥

あればごく一部のカットのみを調べるだけでよいが面積を小さくするためには多くのカットを調べる必要がある。そのため、この論文の結果だけからでは彼らのヒューリスティックが優れているかどうかは実際にはわからない。

(注2): もちろん k 入力 LUT にマッピングするためにはファンイン数が k 以下という制約を満たしていないとマッピングできない。

(注3): カットサイズ k は定数とみなせるのでカットをひとつ扱う手間は定数と考える。

当なものと言えるが、多くの場合、そうではない。2つのカット集合の直積集合は多くの無駄なカット（サイズが k 以上のカット、もしくは極小でないカット）を含んでおり、特にグラフが再収斂構造を持っているときにその傾向は顕著に現れる。

図1のノード a における3-フィージブルカットを列挙するためには、そのファンインである b および c の3-フィージブルカットの集合 $\Phi_3(b) = \{\{b\}, \{d, e\}, \{d, h, i\}, \{g, h, e\}, \{g, h, i\}\}$ と $\Phi_3(c) = \{\{c\}, \{e, f\}, \{h, i, f\}, \{e, i, j\}, \{h, i, j\}\}$ の直積集合を計算する必要がある。この場合、双方とも要素数が5であるから計25組の組合せが考えられるが、その結果として得られる3-フィージブルカットの集合は、 $\Phi_3(a) = \{\{b, c\}, \{b, e, f\}, \{c, d, e\}, \{d, e, f\}\}$ の計4つのみである。

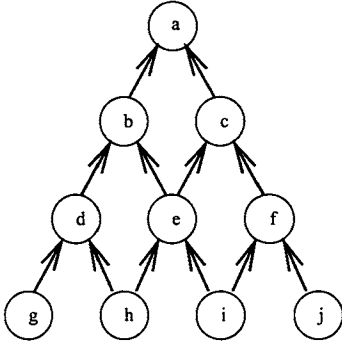


図1 カット列挙対象のグラフ

3.2 トップダウン型アルゴリズム

カットの展開とは、カット c 中のノード v をそのファンインである u_1 と u_2 に置き換えることで、新たなカット c' を作り出す処理のことである。自分自身からなるカット $\{v\}$ 以外のノード v の全てのカットは他のカットからカットの展開を行うことで作り出すことができる。そこで、 $\{\{v\}\}$ を最初のカットとして、既存のカットに可能なカットの展開をすべて適用して新たなカットを作り出す処理を変化がなくなるまで繰り返せば全てのカットを列挙することができる。

図2に図1のノード a に対するカットをトップダウンに列挙する様子を示す。各矢印の横にカットの展開を行ったノード名を記している。実は、 $\{b, c\}$ に対して前に b の展開を行い、次に c の展開を行った結果と、前に c の展開を行い、次に b の展開を行った結果はともに $\{d, e, f\}$ であり同一となる。そこで、図2ではカットの要素として前に現れるノードから前に展開するようにしている。その結果、図2のような木が得られる。最終的に18個のカットが列挙される^(注4)が、その中で3-フィージブルカットは上記の4つのみである。

これがトップダウン型カット列挙アルゴリズムの非常に単純化した方法である。しかし、この単純なアルゴリズムでは k -フィージブルカットのみを求めることはできない — このアルゴリズムではサイズの制約のない全てのカットの列挙を行って

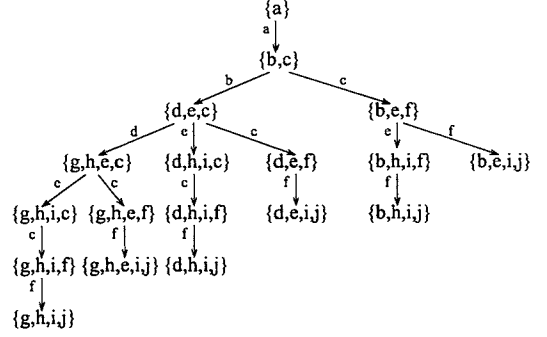


図2 トップダウン型のカット列挙

しまう。さらに、サブジェクトグラフが再収斂の構造を持っている場合には、 k -フィージブルカットが非 k -フィージブルカット（サイズが k 以上のカット）からカット展開により得られる可能性があるため、トップダウン型のカット列挙アルゴリズムでは、現在のカットサイズが k を超えたカットに対しても処理を止めることができない。このような理由により、トップダウン型アルゴリズムに基づいた効率のよいカット列挙アルゴリズムは今まで知られていなかった。

4. 提案アルゴリズム

ボトムアップ型アルゴリズムにとって、非効率性の源はマージ操作にあり、ボトムアップ型の処理を行う限り不可避なものである。ファクターカット (factor cuts) [2] の概念はこの問題に対する直接の解決にはなっておらず、単に、扱うカットの数を減らすことでマージ操作の非効率性を隠蔽しているに過ぎない。

一方、もしも適切なカット展開の終了条件が与えられればトップダウン型アルゴリズムをより効率的にできる可能性は残されている。実際、そのような終了条件は存在する。基本となるアイデアは定理1の中に示されている。

$X = \{C_1, C_2, \dots, C_n\}$ をノードの集合の集合とする。また、 $\Psi(X) = \bigcup_{i=1}^n C_i$ とする。この記法を用いると、 $\Psi(\Phi_k(v))$ はノード v の k -フィージブルカットに含まれているノードの集合を表すことになる。定理1から以下の補題が導かれる。

[補題 1] 外部入力でないノード v に対して以下の式が成り立つ。

$$\Psi(\Phi_k(v)) \subseteq \{v\} \cup \Psi(\Phi_k(u_1)) \cup \Psi(\Phi_k(u_2)) \quad (3)$$

ただし、 u_1 と u_2 は v のファンインである。

証明:

$$\Psi(\Phi_k(v)) = \Psi(\{\{v\}\} \cup (\Phi_k(u_1) \bowtie_k \Phi_k(u_2))) \quad (4)$$

$$= \{v\} \cup \Psi(\Phi_k(u_1) \bowtie_k \Phi_k(u_2)) \quad (5)$$

$$\Psi(A \bowtie_k B) \subseteq \Psi(A \cup B) = \Psi(A) \cup \Psi(B) \quad (6)$$

式(5)の右辺の後半部分を式(6)で置き換えることにより、以下の式を得る。

$$\Psi(\Phi_k(v)) \subseteq \{v\} \cup \Psi(\Phi_k(u_1)) \cup \Psi(\Phi_k(u_2)) \quad (7)$$

(注4)：実際には $\{g, h, e, i, j\}$ は極小セパレータではないのでカットではない。

□

補題 1 から以下の定理を得る。

[定理 2] ノード v の k -フィージブルカットをトップダウン型アルゴリズムで列挙する際には、 $\{v\} \cup \Psi(\Phi_k(u_1)) \cup \Psi(\Phi_k(u_2))$ に含まれるノードよりも外部入力寄りのノードに対するノード展開を行う必要はない。

定理 2 に基づいて以下の新しいトップダウン型アルゴリズムを作ることができる (3)。

```
cut_enumeration(SubjectGraph G, int k) {
    ( $v_1, v_2, \dots, v_v$ )  $\leftarrow$  topological_sort( $G$ )
    foreach  $v_i$  {
         $u_1, u_2 \leftarrow FI(v_i)$ 
         $\{v_i\}$  と  $\Psi(\Phi_k(u_1)) \cup \Psi(\Phi_k(u_2))$  の間に存在するノードに
        印を付ける。
        印の付いたノードに対してのみカット展開を行い、
        カットを列挙する。
    }
}
```

図 3 提案するトップダウン型アルゴリズム

マージ操作と異なり、 $\{v\} \cup \Psi(\Phi_k(u_1)) \cup \Psi(\Phi_k(u_2))$ の時間複雑度は $O(m+n)$ である。ここで、 $m = |\Psi(\Phi_k(u_1))|$ 、 $n = |\Psi(\Phi_k(u_2))|$ とする。カット展開で作られるカットの数を E とすると、トップダウンアルゴリズムの時間複雑度は $O(E)$ となる。カット展開で作られたカットの全てが k -フィージブルカットではないが、提案しているカット展開の終了条件は極めてタイトであると考えられるので、提案アルゴリズムの時間複雑度は概ね列挙されるカット数に比例すると言える。

5. 実験結果

提案アルゴリズムの性能評価のためにいくつかの実験を行った。表 1 に列挙されたカット数、提案するトップダウンアルゴリズムの計算時間 (TD) とボトムアップ型アルゴリズムの計算時間 (BU) を示す。

実験は Pentium IV (クロック周波数 3.4GHz, 1GB メモリ) の計算機で行った。提案アルゴリズムは既存のボトムアップアルゴリズムに比べて非常に高速に処理を行っている^(注5)。9-フィージブルカットの全列挙を行った結果としては知られているものの中でもっとも高速であると思われる。文献 [2] でも 9-フィージブルカットの列挙を行っているが、その実験結果の表記は誤解を与えやすいものとなっている。彼らの実験では全ての 9-フィージブルカットの列挙は行っておらず、各ノードにつきたったの 2000 カットのみを列挙しているだけである (たしかに脚注で “underestimate” とは書いてある)。例えば、文献 [2] では C1355 は 433,995 個の 9-フィージブルカットを持つと報告しているが、実際には C1355 は 8,352,187 個もの 9-フィ

ジブルカットを持っている^(注6)。そのため、文献 [2] で示されている全列挙の場合の計算時間と比較することは意味を持たない。文献 [2] の partial factor の計算は非常に高速である。これはその名の通り、一部の非常に少ないカットのみを列挙しているからである。深さ最小のみを目的としたマッピングの場合には、partial factor は非常にうまく働いているように見えるが、面積の最少化を行う場合にこの partial factor が有効化は疑問の余地が残る。深さだけを小さくする場合には各ノードに対して、その最小深さを得るただひとつのカットが列挙できればよいのに対して、面積を考慮にいたれたマッピングではその最小深さを得る複数の (通常は多数の) カットの列挙を行う必要があるからである [4]。

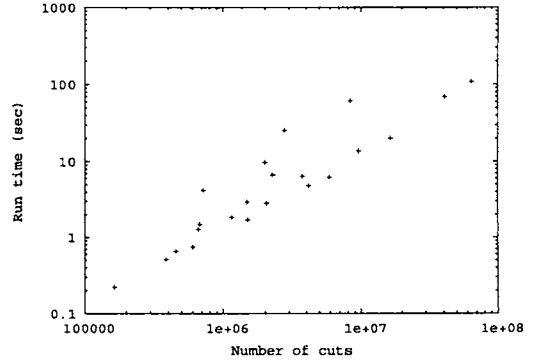


図 4 Run time (TD) vs. number of cuts

図 4 は提案アルゴリズムの計算時間とカット数の関係を両対数軸で示したものである。ほとんどの点が対角線上に並んでおり、このプロットの傾きから得られた計算複雑度は $O(N^{1.01})$ となっている。ここで、 N は列挙されたカットの数である。

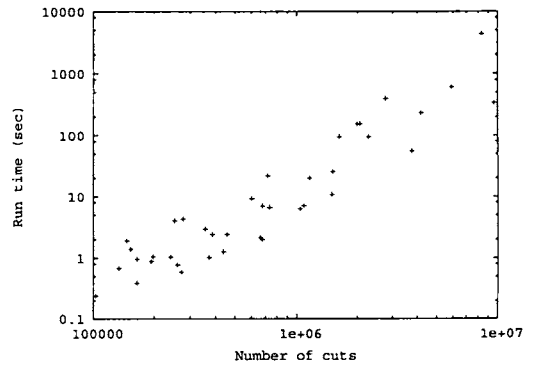


図 5 Run time (BU) vs. number of cuts

図 5 はボトムアップ型アルゴリズムの計算時間とカット数の

(注5) : ‘des’に関しては、ボトムアップアルゴリズムではメモリーオーバーフローのため 9-フィージブルカットを求めることができなかった。

(注6) : 本実験で用いられたサブジェクトグラフと文献 [2] で用いられたものとは 2 項分解のやりかたが少し異なる可能性があるが、カット数のオーダーまでは変わらない。

表 1 提案アルゴリズム (TD) とボトムアップアルゴリズム (BU) の比較

Name	$k = 7$			$k = 8$			$k = 9$		
	Number of cuts	Run time (sec)		Number of cuts	Run time (sec)		Number of cuts	Run time (sec)	
C1355	482777	1.56	9.11	1999127	9.69	151.19	8352187	61.41	4344.56
C1908	131625	0.18	0.39	385827	0.51	2.39	1162247	1.84	19.96
C2670	184839	0.24	0.89	603524	0.74	9.21	2058923	2.81	154.22
C3540	217634	0.39	0.82	678834	1.48	7.00	2274044	6.67	94.21
C499	193465	0.64	1.72	721099	4.18	21.74	2768527	25.51	394.79
C5315	422278	0.54	1.98	1513503	1.70	25.48	5905219	6.17	604.86
C6288	2361841	2.85	23.20	9584837	13.62	336.89	40847099	69.34	1636.76
C7552	1117237	1.29	12.64	4166612	4.79	231.85	16403104	20.07	2757.65
des	862298	1.39	2.29	3743929	6.37	54.98	64237236	108.43	NA
rot	64659	0.12	0.10	164623	0.22	0.39	456229	0.65	2.39
too_large	663105	0.76	2.15	663105	1.27	2.15	1499694	2.92	10.72

関係を両対数軸で示したものである。相関は図 4 よりも低く、少しばらついているように見える。プロットの傾きから得られた計算複雑度は $O(N^{1.8})$ となっている。この結果から提案アルゴリズムとボトムアップアルゴリズムの速度差が定数倍以上あることがわかる。より多くのカットを存在する場合にはカットサイズ以上のより大きな計算時間の差が付くことになる。

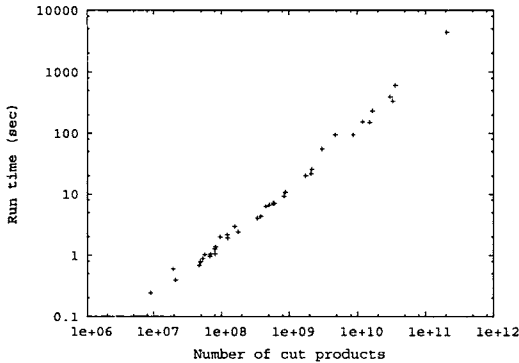


図 6 Run time (BU) vs. number of cut products

図 6 はボトムアップ型アルゴリズムの計算時間とカットマージにおける直積集合のサイズの関係を両対数軸で示したものである。カットマージにおける直積集合のサイズとは以下の式で与えられる。

$$\sum_v |\Phi_k(u_1)| \times |\Phi_k(u_2)|$$

where $\{u_1, u_2\} = FI(v)$

ほぼ全ての点が対角線上に並んでおり、その傾きは約 1 である。以上のことから、ボトムアップ型アルゴリズムの計算複雑度がカットマージにおける直積集合のサイズに比例しており、列挙された最終的なカットの数には依存していないことがわかる。さらに、列挙されたカットの数はカットマージにおける直積集合のサイズよりはるかに小さい。

6. おわりに

本稿では、トップダウン型のアルゴリズムに基づいた効率のよいカット列挙アルゴリズムの提案を行った。本アルゴリズムではカットの展開の終了条件を適切に設定することで、トップダウン型アルゴリズムの探索範囲をコンパクトに保っている。実験結果による評価から、本アルゴリズムが列挙されるカット数にほぼ比例する手間でカットの列挙を行えることが示されている。

また、ボトムアップ型アルゴリズムの性質についての考察を行った。すなわち、ボトムアップ型アルゴリズムの計算時間はカットマージにおける直積集合のサイズに強く依存しており、それが非効率性の原因となっていることを明らかにした。あるアプリケーションにおいてはファクターカット [2] のようにごく一部のカットを列挙するだけのヒューリスティックがうまく働く場合もあるが、全てもしくは多数のカットを列挙しなければならないアプリケーションにとってもボトムアップ型アルゴリズムは現実的な解決策にはならないものと考えられる。

研究課題としてこのカット列挙アルゴリズムを用いた LUT 型 FPGA 用テクノロジーマップの開発が挙げられる。また、ファクターカットのような列挙するカットを削減するヒューリスティックとトップダウン型アルゴリズムの組合せも興味深い課題であると思われる。

謝辞 本研究の一部は文部科学省「知的クラスタ創成事業」補助金による。

文 献

- [1] J. Cong and Y. Ding: "FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA design", IEEE Transactions on Computer-Aided Design, 13, 1, pp. 1-12 (1994).
- [2] S. Chatterjee, A. Mischenko and R. Brayton: "Factor cuts", Proceedings of IEEE International Conference on Computer Aided Design, pp. 143-149 (2006).
- [3] A. Mischenko, S. Chatterjee and R. Brayton: "DAG-aware AIG Re-writing: A fresh look at combinational logic synthesis", Proceedings of 43rd IEEE/ACM Design Automation Conference, pp. 532-536 (2006).
- [4] J. Cong, C. Wu and Y. Ding: "Cut Ranking and Pruning:

The first part of the paper describes the architecture of the mapping solution. The second part describes the implementation of the mapping solution. The third part describes the results of the implementation. The fourth part describes the conclusions of the paper.

The first part of the paper describes the architecture of the mapping solution. The second part describes the implementation of the mapping solution. The third part describes the results of the implementation. The fourth part describes the conclusions of the paper.

The first part of the paper describes the architecture of the mapping solution. The second part describes the implementation of the mapping solution. The third part describes the results of the implementation. The fourth part describes the conclusions of the paper.

The first part of the paper describes the architecture of the mapping solution. The second part describes the implementation of the mapping solution. The third part describes the results of the implementation. The fourth part describes the conclusions of the paper.

The first part of the paper describes the architecture of the mapping solution. The second part describes the implementation of the mapping solution. The third part describes the results of the implementation. The fourth part describes the conclusions of the paper.

The first part of the paper describes the architecture of the mapping solution. The second part describes the implementation of the mapping solution. The third part describes the results of the implementation. The fourth part describes the conclusions of the paper.

The first part of the paper describes the architecture of the mapping solution. The second part describes the implementation of the mapping solution. The third part describes the results of the implementation. The fourth part describes the conclusions of the paper.

The first part of the paper describes the architecture of the mapping solution. The second part describes the implementation of the mapping solution. The third part describes the results of the implementation. The fourth part describes the conclusions of the paper.