社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ＳＬＤＭ－115　（７）
2004／5／28

テクノロジマッピングにおけるDAG被覆アルゴリズムについて

松永　裕介†

† 九州大学システムLSI研究センター
〒816-8580 福岡県春日市春日公園6-1
E-mail: †matsunaga@slrc.kyushu-u.ac.jp

あらまし　テクノロジマッピング問題はDAG被覆問題として定式化することができるが，DAG被覆問題は$\mathcal{NP}$困難であることが知られており，実用規模の回路に対する問題を厳密に解くことは難しい．そのため，多くの実用的なプログラムではDAGを木に分割して各々の木について最適なマッピングを求める木被覆アルゴリズムが用いられている．木被覆アルゴリズムは与えられたグラフ(木)のサイズに比例した時間で最適解を求めることができるが，木に分解することで全体としては最適性が損なわれている．本稿では，DAG被覆アルゴリズムに対する新しいヒューリスティックを提案する．これはDAG中のすべての節点がかならずただ一度だけ被覆される，という条件をDAG被覆に付加したもので，DAG被覆問題の部分問題となっている．一方，木に分解して各々の木に対する被覆を求める場合ももちろん，各節点がかならず一度だけ被覆されるので，提案する問題は従来の木被覆問題を完全に含んでいる．さらに，このすべての節点がかならず一度だけ被覆されるという条件を活かすことでDAGのサイズの二乗に比例した手間で最適解を求めることが可能である．
キーワード　論理合成，テクノロジマッピング，DAG被覆，木被覆，

On a DAG-covering algorithm for technology mapping

Yusuke MATSUNAGA†

† System LSI Research Center, Kyushu University
Kasuga Koen 6-1, Kasuga, Fukuoka, 816-8580 Japan
E-mail: †matsunaga@slrc.kyushu-u.ac.jp

Abstract　Technology mapping can be viewed as DAG-covering problem, which is known an $\mathcal{NP}$-hard. So, it is difficult to solve a practical problem in a reasonable computation time. In the literature, a tree-covering heuristic, which decomposes the entire DAG into a forest of trees, and find the minimum solution for each tree, is proposed. Tree-covering can find the minimum solution in a time propotional to the size of tree, however, the optimality for the entire DAG is lost. This paper presents an novel heuristic for DAG-covering problem, which restricts the problem to find a cover such that each node in a DAG is covered by the solution exactly one times. Obviously, that is a sub-problem of the original DAG-covering. Also, that is a super-problem of tree-covering, because the solution of tree-covering is also an exact one-time cover. It is also shown that the given problem can be solve in a $O(n^2)$ time, where n is the number of node in a DAG.
Key words　logic synthesis, techonology mapping, DAG-covering, tree-covering technology mapping

1. はじめに

テクノロジマッピング問題はDAG被覆問題[1]として定式化することができるが，DAG被覆問題は$\mathcal{NP}$困難であることが知られており，実用規模の回路に対する問題を厳密に解くことは難しい．そのため，多くの実用的なプログラムではDAGを木に分割して各々の木について最適なマッピングを求める木被覆アルゴリズムが用いられている．木被覆アルゴリズムは与えられたグラフ(木)のサイズに比例した時間で最適解を求めることができるが，木に分解することで全体としては最適性が損なわれている．本稿では，DAG被覆アルゴリズムに対する新しいヒューリスティックを提案する．これはDAG中のすべての節点がかならずただ一度だけ被覆される，という条件をDAG被覆に付加したもので，DAG被覆問題の部分問題となってい

る．一方，木に分解して各々の木に対する被覆を求める場合ももちろん，各節点がかならず一度だけ被覆されるので，提案する問題は従来の木被覆問題を完全に含んでいる．さらに，このすべての節点がかならず一度だけ被覆されるという条件を活かすことでDAGのサイズの二乗に比例した手間で最適解を求めることが可能である．

以降，2.章でテクノロジマッピングの従来の技術の概観を行い，5.章で提案するヒューリスティックについて述べる．最後に6.章でまとめと今後の課題について述べる．

2. テクノロジマッピング

テクノロジマッピングとは特定のセルライブラリやFPGAデバイスなどに独立な論理回路を入力として，ネットリストやFPGAの回路を生成することを言う．対象がセルベース設計の場合には，部品としてもちいる

3. テクノロジマッピングの入力

3.1 ブーリアンネットワークとサブジェクトグラフ

実現したい回路の仕様もしくは初期回路としてはブーリアンネットワーク[2]が用いられる．ブーリアンネットワークはテクノロジ非依存な論理合成処理で用いられているデータ構造で，多段論理回路の構造を表す．ブーリアンネットワークの各々の節点は任意の論理関数を持てるので，その節点の一つ一つが，セルライブラリ中のセルやFPGAの基本ブロックに対応づけられるとは限らない．逆に言えば，その節点の一つ一つがセルや基本ブロックに対応するようなブーリアンネットワークを作り出すことがテクノロジマッピングの処理そのものと見なすことができる．このように，ブーリアンネットワークの構造をどのように決めるかはテクノロジマッピングの重要なポイントとなっている．そこで，文献[1],[3]等で用いられている手法では，ブーリアンネットワークの各節点を2入力NANDゲートに分解したものを初期入力として用いている(注1)．任意の論理関数はAND-ORの形で表現可能であり，任意のANDゲートおよびORゲートは2入力NANDゲートとインバータに分解可能なので，任意の論理関数は2入力ゲートに分解可能である．このように2入力ゲートに分解されたネットワークをサブジェクトグラフ(subject graph)と呼ぶ．一つのブーリアンネットワークに対して分解の異なるサブジェクトグラフが考えられ，また，その分解の仕方によって得られる最終的な解も異なるが，マッピング前にどのような分解が良いのかを知る術がなく，また，すべての分解を試すことも困難なので，適当なサブジェクトグラフが用いられる．

3.2 セルとパタングラフ

あらかじめ設計済のセルライブラリを用いたセルベース設計の場合には，サブジェクトグラフの各部分を実際のセルで実現することになる．そのため，セルの論理式をサブジェクトグラフと同様に2入力NANDゲートに分解し，グラフにしたものを用意する．これはパタングラフと呼ばれ，サブジェクトグラフへのマッピングに用いられる．つまり，サブジェクトグラフの部分グラフのうちパタングラフと同形のものがあればその部分は該当するセルで実現可能であるということが分かる．以降，このパタングラフと同形の部分グラフのことをマッチ **(match)** と呼ぶ．通常はマッチごとに対応するセルの面積や遅延時間などを反映したコストが割り当てらる．以下でもマッチ m のコストを $COST(m)$ で表すものとする．

3.3 LUT型FPGAのマッピング

LUT型FPGAは一つの基本ブロックで定められた入力数以下の任意の関数を実現可能という特徴を持つ．明らかに効率の悪いやり方としては前述のセルライブラリの場合と同様に，定められた入力以下のすべての関数に対するパタングラフを登録しておいてセルベース設計と同様のマッピングを行うことも考えられるが，通常はサブジェクトグラフから定められた入力数以下の部分グラフを列挙することでマッピングを行っている．LUT型FPGAのマッピングではこの定められた入力数以下の部分グラフがマッチ **(match)** となる．LUT型FPGAの場合もマッチに対してコストが割り当てられるがセルライブラリの場合と異なり面積のコストは一様となる．

4. DAG covering と tree covering

以上のように回路を表すサブジェクトグラフに対するマッチという概念でテクノロジマッピングの問題を定式化してみると，サブジェクトグラフに含まれるすべての節点を被覆するようなマッチの集合を求める問題 (すなわち被覆問題) と見なすことができる．ただし，通常の被覆問題よりも難しいのは，ただ単に被覆すれば良いのではなく，マッチの入力 (すなわちセルや基本ブロックの入力) には他のマッチの出力 (もしくは外部入力) が接続していなければならないという条件が付加されていることである．以上のことを少し厳密に表現すると以下のようになる．

マッピング対象のサブジェクトグラフを $G(V,E)$ とする．ここで，V は節点の集合，E は節点を結ぶ有向辺の集合とする．G 上の部分グラフでセルのパタングラフやLUT型FPGAの基本ブロックに一致するものをマッチと呼ぶ．あるマッチ m に対して，そのマッチに被覆される節点の集合を $COVER(m)$，そのマッチの出力に対応する節点を $ROOT(m)$，そのマッチの入力に対応する節点の集合を $LEAF(m)$ とする．与えられたサブジェクトグラフ上でのすべてのマッチの集合を M_G で表す．このマッチの部分集合 $M \subseteq M_G$ がネットワーク G を被覆するとは，

$$\forall v \in V, \exists m \in M, v \in COVER(m) \quad (1)$$

が成り立つことを言う．マッチ m が解に含まれる時に真になる論理変数を X_m とし，節点 v を被覆するようなマッチの集合を $M_v = \{p_1^v, p_2^v, \ldots\}$ とすると，上の(1)式を満たすマッチの集合を表す論理式は，

(注1)：[3]では3入力NAND，4入力NANDなども用いている．

$$C_1 = \prod_{v \in V} (X_{p_1^v} + X_{p_2^v} + \cdots) \quad (2)$$

で表される．この論理式を充足する (否定のリテラル $\overline{X_i}$ がないのでこの論理式は必ず充足可能である．) 変数の値の組合せのなかで，もっとも 1 となる変数の少ない組合せを求める問題 (もしくは各々の変数ごとに設定された重みの和を最小にするような組合せを求める問題) は一般に最小被覆問題 (minimum covering problem) と呼ばれる．例えば，積和形論理式の簡単化問題もこの最小被覆問題で表すことができる．この問題は $\mathcal{NP}$ 問題であるが，比較的性能のよい近似手法が知られている (例えば文献 [4])．

ところが，前述のようにテクノロジマッピングの場合にはこの被覆条件だけでは十分ではない．マッチの結果として得られたネットリストが実現可能 (feasible) でなければならないので，あるマッチ m が解に選ばれるのならば，その入力となる節点 $u \in LEAF(m)$ を根とするような別のマッチも選ばれなければならない，という条件がさらに必要となる．

具体的には，

$$\forall m \in M, \forall u \in LEAF(m), \quad \exists l \in M, ROOT(l) = u \quad (3)$$

となる．この条件は，節点 u を根に持つマッチの集合を $\{q_1^u, q_2^u, \ldots\}$ とすると，

$$C_2 = \prod_{m \in M_G} \left(\prod_{u \in LEAF(m)} (\overline{X_m} + X_{q_1^u} + X_{q_2^u} + \cdots) \right) \quad (4)$$

という論理式で表される．以上をまとめると，

$$C = C_1 \wedge C_2 \quad (5)$$

と言う条件式を得ることができる．このように，与えられたサブジェクトグラフをマッチで被覆する問題を DAG 被覆と呼ぶ．ここで，DAG とは directed acyclic graph の略で閉路の無い有向グラフを指す．DAG 被覆は，この条件式 (5) を充足するような値の組み合わせのうち，もっともコストの低いものを探せば良いのであるが，問題は 2 項目の C_2 である．ここには各和項中に 1 つの否定のリテラルが含まれているので，変数 X_i の値を $0 \rightarrow 1$ と変化させることによって C_2 の値は $0 \rightarrow 1$ と $1 \rightarrow 0$ のどちらにも変化する可能性がある．このような論理関数を binate 関数と呼ぶ．逆にいかなる変数の値の変化に対して単調変化しかしない論理関数を unate 関数と呼ぶ．この関数の呼び名にちなんでこの問題は binate covering 問題とも呼ばれる．積和形論理式の簡単化に現われる最小被覆問題は unate covering 問題となる．

この binate covering 問題は古くは'50 年代に不完全指定順序機械の簡単化問題を解くために研究されており，問題のサイズを小さくする簡約化アルゴリズムなどは知られているが，厳密解を求めることができるのは数十変数程度である．この DAG 被覆の場合には変数の数は $|M_G|$ となり(注2)，低く見積もってもネットワークのサイズに比例するものと考えられるので，数千〜数万ゲート規模の回路のマッピング問題をこの binate covering 問題として解くことは現実的には不可能である．また，binate covering 問題はそのまま 0-1 整数計画問題に変換することが可能であるが，それにしても数千〜数万変数の 0-1 整数計画問題を効率よく解くことは難しい．

そこで考え出されたのが，サブジェクトグラフをそのファンアウト数が 1 の部分グラフ (グラフ理論でいうところの tree: 木) に分解し，その各々の部分グラフに対して最適なマッチの組合せを求める，という手法である [3]．

対象が DAG から木に変わると動的計画法によってグラフのサイズに対して線形時間で最適解を求めるアルゴリズムが存在する．これは，ある節点 v を根とする部分木に対する最適解は，その部分木に含まれる節点 (を根とする部分木) に対する最適解から計算できることによる．つまり，入力の節点から始めて自分を根とする部分木に対する最適解を計算してゆくことで，最終的に与えられた木全体の最適解が求められるというものである．

この木被覆はその木に対するマッピングとしては常に最適解を生成し，実用的にも高速なアルゴリズムであるが，回路全体を木状に分割することによって，いくつかの最適性は損なわれてしまう．具体的には，

- 分割された回路の境界を跨いだセル割り当てができない．
- 境界での位相が合わないと位相合わせのためのインバータが挿入されることになり全体では最適とならない．

という問題がある (図 1)．

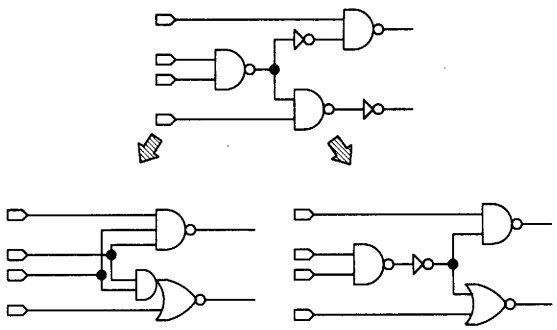

図 1　ファンアウトを跨いだセル割り当て

この図の例に対する適切な解は (セルライブラリや FPGA デバイスにもよるが)，左下の回路例のように 3 入力 NAND と AND-OR-INV を 1 つづつ用いるものであるが，もしも，もととなったブーリアンネットワークを木状に分割してしまうとこのような結果を得ることはできない．さらにその分割点をどちらの極性で実現するかによって得られる回路は異なったものとなる．もしも面積を小さくしたいのであれば，右下の回路のよ

(注2)：集合 S に対して，その要素数を $|S|$ で表す．

うにもとのネットワークとは反対の極性で実現した方が良い解を得ることができる．しかし，一般にはファンアウトを跨いだセル割り当ての有効な手法は提案されていないので，多くの場合はサブジェクトグラフを木状に分割してから木被覆を行っている．

5. DAG タイリングアルゴリズム

Kao と Lai は FPGA 用テクノロジマッピングの DAG 被覆に対するヒューリスティックを提案している [5]．これはゲートの重複を行わない (前述の用語でいうとサブジェクトグラフ上の節点を複数回，被覆しない) という制約をつけた上で面積 (基本ブロック数) が最小の解を求める，というものであるが，実際には厳密アルゴリズムではなく簡単な近似アルゴリズムとなっている．本稿ではこの制約を持つ DAG 被覆問題 (ここでは **DAG タイリング (DAG-tiling)** と呼ぶことにする) の性質について解析を行い，多項式時間で面積最小解を求める厳密アルゴリズムを提案する．

5.1 DAG タイリング問題

DAG タイリング問題とは，以下の式を満たすようなマッチの集合 $M \subseteqq M_G$ を求める問題ということができる．

$$\forall v \in V, \sum_{m \in M} COVERED(v,m) = 1 \qquad (6)$$

ここで $COVERED(v,m)$ は次式のように定義される関数である．

$$COVERED(v,m) = \begin{cases} 1 & \text{if } v \in COVER(m) \\ 0 & \text{otherwise} \end{cases} \qquad (7)$$

つまり，サブジェクトグラフ中の節点 v を被覆するマッチの個数は常に 1 である．ということになる．

もちろん，どのような解が有効なわけではなく，例えば面積 (基本ブロック数) などの尺度で最適な解を求めることが実用的には意味がある．

一般の DAG 被覆問題では被覆条件とは別に実現可能条件が必要であったが DAG タイリング問題では不要となる．なぜならすべての節点をかならず一度ずつ被覆する，という条件が実現可能条件を兼ねているからである．

また，DAG を木に分割し，各々の木に対して木被覆を求めると，明らかにすべての節点がただ一度ずつ被覆されることになるので，木被覆の解も DAG タイリング問題の充足解であることが分かる．

このように DAG タイリング問題は DAG 被覆問題の部分問題であり，かつ，自身の部分問題として木被覆問題を含む丁度，中間の問題であることが分かる．

5.2 面積最小解を求める DAG タイリングアルゴリズム

では，この DAG タイリング問題を効率的に解くためにはどのような工夫が必要であろうか？ Kao と Lai はこの問題に対するヒューリスティックをいくつか提案している [5] が，外部入力から近い部分では比較的うまく働いているものの，総じて段数の深い回路に対してはヒューリスティックが有効ではないように見える．実際には，面積最小の DAG タイリングは多項式時間で厳密解を求めることが可能であるのでこのようなヒューリスティックは不要である．DAG タイリングでは以下のように，木被覆と同様に動的計画法の最適性の原理を用いることが出来る．

まず，サブジェクトグラフ中の節点のうち，ある節点 v へ到達可能な節点の集合を v の推移的ファンイン **(transitive fanin)** と呼び，$TFI(v)$ で表すことにする (v も $TFI(v)$ に含まれるものとする)．以下の式 (8) を満たすようなマッチの集合 M_v を v に対する DAG タイリング被覆と定義する．

$$\forall u \in TFI(v), \sum_{m \in M_v} COVERED(v,m) = 1 \qquad (8)$$

つまり，M_v とは節点 v の推移的ファンインに対する DAG タイリング被覆解である．

次に節点 v の面積コスト $AREA(v)$ を次のように定義する．

$$AREA(v) = \sum_{m \in M_v} COST(m) \qquad (9)$$

節点 v において面積コスト $AREA(V)$ が最小となるような v の DAG タイリング被覆を v の**最適 DAG タイリング被覆**と呼び $\hat{M}_v$ で表すことにする．

また，サブジェクトグラフ全体の面積コストも同様に定義することができる．サブジェクトグラフ全体の面積コストを最小にするマッチの集合を同様に最適 DAG タイリング被覆と呼び $\hat{M}$ で表すことにする．

DAG タイリング被覆 M に対して，各々のマッチ $m \in M$ の出力の節点の集合を関節点と呼び $ROOTS(M)$ と表す．つまり，

$$ROOTS(M) = \{v | \forall m \in M, v = ROOT(m)\}$$

である．

このとき，以下の定理が成り立つ．

[定理 1] 適 DAG タイリング被覆 $\hat{M}$ はその関節点 $v \in ROOTS(\hat{M})$ における最適 DAG タイリング被覆 $\hat{M}_v$ から構成される．

証明: もしもある関節点 v におけるマッチが最適 DAG タイリング被覆解に含まれていない場合には，その節点における解を最適 DAG タイリング被覆解に取り替えることで全体の面積コストを下げることが可能である(注3)．これはもともと全体の被覆解が最適であることに矛盾する．よって，すべての関節点における被覆解は最適である． □

この定理により，木被覆の場合と同様に，サブジェクトグラフの各々の節点における最適 DAG タイリング被覆を求めて行けば全体の最適 DAG タイリング被覆が求められることが分かる．面積最小の最適 DAG タイリング被覆を求めるアルゴリズムは以下の様になる (図 2)．

(注3)：一般の DAG 被覆では $TFI(v)$ に含まれない部分に影響をおよぼすことがあるため，$TFI(v)$ の部分が最適でも全体のコストが上昇する場合がありうる．

```
DAG_tiling(Graph G) {
    G の各節点を入力から出力方向へトポロジカル順で整列
    for G の各節点 v に対して {
        C_v ← +∞;
        for v を根 (出力) とするマッチ m に対して {
            c ← m の面積;
            for m の各入力に対応する節点 u に対して
                c ← c + C_u;
            C_v ← min(C_v, c);
        }
    }
}
```

図 2　DAG タイリングのアルゴリズム

実はこれはほとんど木被覆アルゴリズムと同一であるが，DAG の場合には異なる入力の推移的ファンインが重複している場合があるため，面積の計算だけは特別に考慮する必要がある．

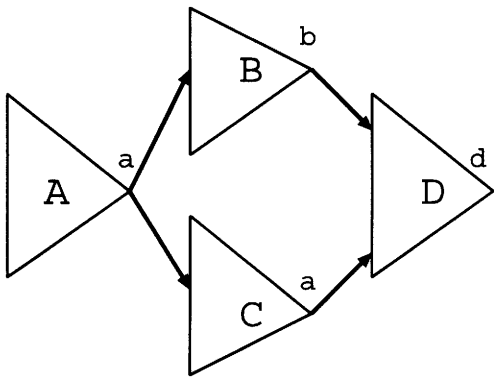

図 3　面積コストの計算

図 3 の例において，節点 d における面積を計算するものとする．もちろん，答は $A+B+C+D$ であるが，d の入力である b と c から各自の面積コスト $AREA(b)$ と $AREA(c)$ はそれぞれ $A+B$ と $A+C$ となるため，単純に $AREA(b)$ と $AREA(c)$ を足し合わせることはできない．そこで，面積コストはスカラ値ではなく，各々のマッチにおける出力の節点をキーとして，その面積を値とした 2 つ組 $(v, area)$ の集合として保持するようにする．図 3 の例においては $AREA(b) = \{(a, A), (b, B)\}$, $AREA(c) = \{(a, A), (c, C)\}$ となる．そして，d においてその入力の面積を足し合わせるときには実際には集合和操作を行う．そうすることによって $AREA(d) = \{(a, A), (b, B), (c, C), d, D)\}$ を得ることができる．この集合の形の面積コストから実際のスカラ値を得るためには，節点を表すキーを無視して，すべての値を足し合わせれば良い．

図 2 より明らかなように，この DAG タイリングアルゴリズムは各節点につき一回ずつ処理を行う．サブジェクトグラフの節点数を n とすると外側のループは n 回，回ることになる．また，各節点においてはマッチの数だけループを回るが，マッチの数はセルライブラリや FPGA の仕様に基づいて定まる数であり，サブジェクトグラフを問題の入力と見なした場合には定数と見なすことが可能である．また，前述のように面積コストを表すデータ構造として各節点とその面積の組の集合の形をとっているために，面積コストの計算に $O(n)$ を必要とする．そのため，アルゴリズム全体の計算複雑度は $O(n^2)$ となる．

6. おわりに

本稿では，テクノロジマッピングで用いられる DAG 被覆問題に対するヒューリスティックとして DAG タイリング問題の厳密アルゴリズムを提案した．このアルゴリズムは，木状に分割した木被覆アルゴリズムと同様に部分回路の最適解が回路全体の最適解を構成するという性質を利用することで，多項式時間で厳密最小解を求めることができる．ただし，面積コストの計算が木被覆アルゴリズムより複雑なため計算複雑度は $O(n^2)$ となっている．ここで n は回路の規模 (ゲート数) である．

本稿では面積最小解を求めるアルゴリズムを示したが，遅延コストが簡単に計算できるモデルの場合には遅延制約を満たす中での面積最小解を求めるアルゴリズムを開発することも容易であると思われる．テクノロジマッピングの問題はこの被覆問題だけでなく，もととなるサブジェクトグラフの生成／2 入力ゲートへの分解，およびマッチングのすべてが合わさったはじめて完成するものであるため，他の部分問題も含めて総合的に性能のよいテクノロジマッパの開発が今後の課題である．

謝辞　本研究の一部は日本学術振興会科学研究費基盤研究 (B)(2)「論理関数処理を用いた FPGA テクノロジマッパの開発」(課題番号 15300019) による．

文　献

[1] R. Rudell: ""Logic Synthesis for VLSI Design"", PhD thesis, U. C. Berkeley (1989).

[2] R. K. Brayton, R. L. Rudell, A. Sangiovanni-Vincentelli and A. R. Wang: ""MIS: A Multiple-Level Logic Optimization System"", IEEE Trans. Comput.-Aided Design Integrated Circuits, **CAD-6**, 6, pp. 1062–1081 (1987).

[3] K. Keutzer: ""DAGON: Technology Binding and Local Optimization by DAG Matching"", Proceedings of 24th Design Automation Conference, pp. 341–347 (1987).

[4] R. K. Brayton, C. McMullen, G. D. Hachtel and A. L. Sangiovanni-Vincentelli: ""Logic Minimization Algorithms for VLSI Synthesis"", Kluwer Academic Publishers (1984).

[5] C.-C. Kao and Y.-T. Lai: "Area-minimal algorithm for lut-based fpga technology mapping with duplication-free restriction", ASP-DAC 2004, pp. 719–724 (2004).

[6] H. Savoj, H. Touati and R. K. Brayton: ""Improved Scripts in MIS-II for Logic Minimization of Combinational Circuits"", International Workshop on Logic Synthesis (1991).

[7] R. J. Francis, J. Rose and Z. Vranesic: "Chortle-crf: Fast technology mapping for lookup table-based fpgas", the 28th ACM/IEEE Design Automation Conference, pp. 613–619 (1991).

[8] R. Murgai, N. Shenoy, R. K. Brayton and A. Sangiovanni-Vincentelli: "Improved logic synthesis algorithms for table look up architectures", the International Conference on Computer-Aided Design, pp. 564–567 (1991).