多段論理合成における二段論理式簡単化の一手法

1J−4

†松永　裕介　　††K. C. Chen　†藤田　昌宏

†(株)富士通研究所　　††FAI

1. はじめに

多段論理回路の簡単化手法のうち，多段回路の構造から生じたドントケア条件を考慮して簡単化を行なうBoolean minimizerと呼ばれるものがいくつか提案されている[1],[2],[3]．このうち，文献［1］の手法は2段論理式簡単化プログラムESPRESSO-II[4]を利用したもので，どのような回路変換を行なうかというヒューリスティックに関しては場当たり的な変換を行なう他の手法に比べて効率的であるが，通常，多段論理回路の内部のドントケアを表現する2段論理式が巨大なものになってしまい，ドントケアそのものやその否定を扱うことが非常に困難になるという問題がある．そこで，本稿では2分決定グラフ（Binary Decision Diagram）[5]を用いて許容関数を表現しつつ，ESPRESSO-IIと同様のヒューリスティックを用いた2段論理式簡単化手法についての提案を行なう．

2. ESPRESSO-IIとMinimum Column Cover Problem

ESPRESSO-II[4]はexpand, irredundant, reduceの3つの主要な処理から構成される．expandは各積項から可能なかぎりリテラルを除く処理を行なう．その際に，なるべく他の積項を被覆するような方向へ積項を拡大させている．irredundantはexpandで得られた積項の集合から冗長なものを取り除く処理を行なう．reduceは前記の2つの処理によって局所最適解に陥るのを防ぐために各積項を可能なかぎり縮小する（リテラルを付け加える）処理を行なう．このうち，reduceは処理する積項の順番を固定すれば，リテラルの追加の順序に関わらず同一の結果が得られるため，許容関数を用いても逐次リテラルを追加することができる[6]．しかし，expandおよびirredundantの場合は逐次的にリテラル／積項の削除を行なうとその処理順序によって解が異なる．また，前もって適切な順序を決めることも難しい．そこで，ESPRESSO-IIではそのような方法はとらずに削除できるリテラル／積項の組を表すテーブルを作り，そのなかで削除するリテラル／積項を決定するというアプローチをとっている．

関数Fのonセット(出力が1となるべき最小項の集合）を被覆する積項の集合を$\{P_i \mid i=1,2,....,n\}$とする．Fのonセットを適当な部分空間に分け，各行がその部分空間に，各列が積項に対応している行列Mを作る．各行列の要素d_{ij}は次のように定められる．

$d_{ij}=1$　第i行に対応する部分空間を第j列に対応する積項が被覆している場合．

$=0$　それ以外．

積項の部分集合Qが関数Fのonセットを被覆するための必要十分条件は，FとP_iから作られる行列Mの各行iに対して，$\Sigma d_{ij} > 0 \{j \mid j \in Q\}$が成り立つことである．そこで，このような行列の各行で1の値をとる列を最低1つは含んでいるような要素数最小の列の集合を求める問題（minimum column cover problem）を解くことで，積項数最小の論理式を求めることができる．以上がirredundantの中心となる処理である．expandの場合，要素数が最小だけでなく他の積項をなるべく被覆するようなリテラルの集合を求めることが目的となっているため，irredundantのように単純にminimum column coverの問題として解くことができないが，どのようなリテラルの集合が同時に削除可能かを判断するために上記と同様の行列を用いている（この場合，offセットの部分空間が行に，リテラルが列に対応する．offセットの部分空間と交わらないリテラルが1となる）．そこで，このような行列（本稿では被覆行列と呼ぶ）を作り出すことができれば，効率良く削除すべきリテラル／積項の選択が行なえることになる．

3. 2分決定グラフを用いた被覆行列の生成

A BDD-based Two Level Minimizer for Multi Level Logic Synthesis

†Yusuke Matsunaga, ††K. C. Chen, and †Masahiro Fujita

†FUJITSU Lab. LTD. ††FUJITSU AMERICA INC.

前述のように，多段回路内部のドントケアを2段の積和形論理式で表現すると通常は莫大な積項数を必要とし，メモリ量・計算量の点で問題となる．特に，expand処理ではonセットとdcセットからその否定であるoffセットを計算するため，巨大なdcセットは全体の効率に非常に悪影響を及ぼす．そこで，これらのonセット，offセット，dcセットを許容関数[3]として2分決定グラフで表現し，この2分決定グラフから被覆行列を生成する手法を提案する．

図1に処理の概要を示す．関数cover_matは引数として，被覆すべきonセットgおよび，各積項の論理関数の配列f[]を取り，結果として被覆行列を返す．処理は，一般のapply演算と同様に各グラフの中で最もインデックスの小さい節点（top vertexと呼ぶ）を選び，その変数を0／1に固定した部分グラフを引数としてcover_matを再帰的に呼びだすことで全ての入力空間を探索する．最悪の場合，最小項に到達するまで変数の値を固定する必要があるが，一般には様々な打ち切り条件が働くため，もっと早い時期に探索が打ち切られる．まず，gが0になってしまった場合，そのような入力空間はもはや被覆する必要はないのでそれ以後の探索が打ち切られる．また，f[]のうち，0でないグラフが全て同一のグラフになった場合には，gがどのような論理であろうと（0ではない），そのようなグラフに対応する列が1となっている行を被覆行列に追加する．それ以降の探索は必要ない．gが1になった場合には，関数in_scopeが呼ばれる．以後の探索ではgは1であることが明かなので，もはや引数としては渡されない．もし，f[]の中に1となるものが現われた場合には，そのような積項に対応する列が1となっている行を被覆行列に追加する．

4．おわりに

本稿では，2分決定グラフを用いた被覆行列の生成手法についての提案を行なった．この手法により，2分決定グラフで表現された論理関数，および許容関数を用いて多段論理回路の簡単化を行なう際に，minimum column cover を使ったより効率的な回路変換が可能になるものと思われる．

参考文献

［1］ K. A. Bartlett, R. K. Brayton, G. D. Hachtel, R. M. Jacoby, R. Rudel, A. Sangiovanni-Vincentelli, A. Wang, "Mulitilevel Logic Minimization Using Implicit Don't Cares", IEEE Trans. on CAD, June, 1988, pp. 723-740.

［2］ D. Bostick, G. D. Hachtel, R. Jacoby, M. R. Lightner, P. Moceyunas, C. R. Morrison, D. Ravenscroft, "The Boulder Optimal Logic Design System", IEEE International Conference on Computer Aided Design, November 1987, pp. 62-65.

［3］ S. Muroga, Y. Kambayashi, H. C. Lai, J. N. Culliney, "The Transduction Method - Design of Logic Networks based on Permissible Functions", to apper IEEE Trans. on Computers, in 1989.

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

［5］ R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Trans. on Computers, August, 1986, pp.677-691.

［6］ 松永 裕介，藤田 昌宏，”トランスダクション法の評価と改良”，情処研報 Vol. 89, No.108,89-DA-50-8, 1989, pp.55-62.


```
cover_mat(g, f[], n) {
/* g は被覆されるべきonセットを表すBDD */
/* f[] は各積項の論理を表すBDDの配列 */
/* n は積項数 */
    if (g == 0) return;
    if (全ての0でないf[i]が等しい) {
        そのようなf[i]に対応する列に1を
        持つ行を結果の被覆行列に加える;
        return;
    }
    if (g == 1) {
        in_scope(f[], n);
    } else {
        v ← top vertex of g and f[i];
        cover_mat(gv = 0, fv = 0[], n);
        cover_mat(gv = 1, fv = 1[], n);
    }
}

in_scope(f[], n) {
    if (f[i] == 1であるようなf[i]がある) {
        そのようなf[i]に対応する列に1を
        持つ行を結果の被覆行列に加える;
        return;
    }
    v ← top vertex of f[];
    in_scope(fv = 0[], n);
    in_scope(fv = 1[], n);
}
```

図1：2分決定グラフを用いた被覆行列生成処理