5B－10

エラー補償型トランスダクション法による回路の最適化について

澤田 直 *　桑野浩二 +　上林弥彦 +
* 九州大学工学部　+ 京都大学工学部

1 まえがき

近年のVLSI技術の進歩や計算機能力の向上にともない、計算機による論理回路設計は不可欠となっている。それにともない、数々の論理回路最適化手法が考案され、実用化されている。

トランスダクション法[2]は、与えられた回路に対してそれに含まれる冗長性を元に回路の変形削減を行なう手法である。この手法は1970年代に開発されたものであるが、最近になってその性能が評価され、BDD(Binary Decision Diagrams)との結合により論理回路最適化の標準的な手法となりつつある。

トランスダクション法の一手法として、エラー補償法と呼ばれるものがある。この手法も同時期に開発されたものであるが[1]、これまでは文献[4]を除いて、利用した研究発表はほとんどない。これは、文献[4]にもあるように、計算時間が膨大であることが一つの要因であると考えられる。しかしエラー補償による最適化能力が強力である点[1]は無視し得ない。

本稿では、まずエラー補償の概要を示し、その改良法について主に高速化の観点から考える。

2 基本的事項

本章では、エラー補償法についての基本的な概念について述べる。以下では論理回路の構成ゲートとしてNORゲートのみを扱うが、一般化は容易であると考えられる。

2.1 誤りつき許容関数

あるゲートあるいは入力端子、結線で実現される関数を、関数fで置き換えても回路の出力が変化しない場合、fをこのゲート、入力端子、結線に対する許容関数という。また、現実の許容関数(集合)と望ましい(正しい)許容関数(集合)とが異なる場合、前者を誤りつき許容関数(Compatible Sets of Permissible Functions with Errors、以下CSPFE)という。

CSPFEは、0・1・*・0'・1'の5値をとる関数として表される。このうち、0'(0エラー)、1'(1エラー)は現在の関数値はそれぞれ0、1であるが1、0に変わるのが望ましい、という意味を表す。本稿では、このCSPFEを表現するのに共有二分決定グラフ(SBDD)[3]を用いる。

2.2 エラー補償法のアルゴリズム

この節では、基本的なエラー補償法のアルゴリズムについて述べる。

[基本アルゴリズム]

1) 各ゲートの出力関数を計算する。
2) 回路からゲートを一つ削除し、出力ゲートのCSPFEを計算する。
3) 出力ゲートに近いゲートから、以下を繰り返す。
   - 3.1) 各入力結線のうち冗長なものを削除する。
   - 3.2) 新たなエラーを発生させないように、このゲートに接続可能なゲートを回路から探索する。
   - 3.3) このゲートの0エラーを補償する。これは既存の入力結線と3.2)で求めた結線とを置き換えることで実現する。
   - 3.4) このゲートの1エラーを補償する。これは3.2)で求めた結線を新たに接続することで実現する。
   - 3.5) このゲートで補償できたエラーがあれば、これをファンアウト側へ伝える。この時、回路の出力の全エラーが補償されれば5)へ進む。
   - 3.6) 各入力結線にCSPFEを伝搬する。
4) 回路を2)の削除前の状態に戻し、別のゲートの削除を試みるために2)に戻る。削除するゲートがなくなればアルゴリズムを終了する。
5) 回路が改良される限り1)～4)を繰り返す。

3 エラー補償法の改良

本章では、エラー補償法の改良について主に高速化の観点から考える。

3.1 問題点

前述した通り、エラー補償法の計算時間は他のトランスダクション法と比較しても非常に大きい。これは主に繰り返し回数の多さが要因である。ゲート数をnとすると、トランスダクション法の場合計算時間は$O(n^2)$であるのに対して、エラー補償法では$O(n^3)$となるためである。

エラー補償法では以下のような繰り返しが行なわれる。

i) 基本アルゴリズムの2)でのゲートの削除
ii) 3)での各ゲートにおけるエラー補償の実行
iii) 3.2)における接続可能なゲートの探索

高速化のために、これらの繰り返し回数をできる限り減らしていくことが重要である。以下、順に考察していく。

3.2 削除するゲート数の制限(手法1)

ゲートを削除した結果、エラーとなる出力端子の数が多ければ回路の全エラーを補償するのは難しい。エラーとなる要素の数が多い場合も同様である。

Network Optimization by Transduction Method Based on Error Compensation
Sunao SAWADA*, Koji KUWANO+, Yahiko KAMBAYASHI+
*Faculty of Engineering, Kyushu University
+Faculty of Engineering, Kyoto University

よってこれらのゲートを削除の対象から除く方法が考えられる。しかしこの手法は絶対的なものではないので、回路によっては最適化された結果が悪くなることも十分考えられる。

3.3 補償を行なうゲート数の削減 (手法 2)

ゲートの CSPFE が 0' または 1' となるのは、そのゲートの全ての出力結線の CSPFE の該当要素が 0' または 1' の場合である。つまり、出力結線の一部にエラーが存在してもゲートの CSPFE はエラーとならない。これをエラーとすると新たなエラーを生み出す可能性が出てくるためであるが、逆に考えればこの結線のエラーはこのゲートでは補償されることはなく、ファンイン側のゲートでも補償される可能性は少ない。

よって、これらのゲートをエラー補償実行の対象から除くこととする。こうすると、これらのゲートの他のエラー要素の補償が行なわれなくなるが、回路全体の観点から見れば影響は少ないと考えられる。

同様の議論から、ゲートの CSPFE のエラーの要素がない場合もファンイン側のゲートを補償実行の対象から除くことができる。

4 実験結果

基本アルゴリズムによるエラー補償法と前述した手法を合わせたエラー補償法による論理回路最適化プログラムを C 言語を用いて作成し、SPARC STATION 10 上で実験を行なった。

初期回路としては、LGSynth'91 多段ベンチマーク回路をファンイン 4 までの NOR ゲートにマッピングしたものを用いた。回路は主に 200 ゲートまでのものを選んだ。それ以上の回路では実行可能な BDD ノード数 (最大 100 万ノード) を超えて実行が止まる、あるいは実行時間が膨大になるためである。SBDD 処理は、NTT の湊真一氏による SBDD パッケージを使用している。

表 1 に基本アルゴリズムと手法 2 による回路の最適化結果を示す。この表の回路を含む 16 の回路でトランスダクション法の Connectable/Disconnectable [2] による最適化の結果と比較すると、基本アルゴリズムの方がゲート数が平均 4.5% 多く結線数は逆に 1.9% 少ない。出力端子数が 8 以上の回路の場合基本アルゴリズムの方がゲート数で 11.1% 結線数で 4.0% それぞれ多いが、それ以外の回路ではほぼ同じか多少基本アルゴリズムの結果が良かった。

表 2 に手法 1 を用いた場合の結果を示す。なお、表中 * が付いているものは基本アルゴリズムの結果と異なる回路が得られたものを表し、— は実験を行なっていないことを表す。エラーとなる出力端子の数が 1 個の場合に限り補償を行なう (手法 1-1) 場合、複数の出力端子を持つ回路で実験すると、半数は基本アルゴリズムの場合と同じ回路が求められ、計算時間は 30% から最大 94%(cordic) 削減されたが、残りの回路では異なる回路が得られ計算時間が逆に増大する場合 (lal) もあった。エラーとなる出力端子の個数の上限を 2 とする (手法 1-2) と、計算時間の削減効果は減少する。

また手法 2 を用いると、基本アルゴリズムの結果と比較してゲート数で平均 3.8%、結線数で 1.8% 増加するのに対し、計算時間は 61% 減少した。

表 1: 基本アルゴリズムと手法 2 の実験結果

回路	I/O	初期回路	適用結果	手法 2
9symml	9/1	168/402	148/386/1925	167/406/903.4
cm150a	21/1	70/131	48/103/123.8	49/101/87.2
cordic	23/2	105/197	69/124/119.2	69/124/69.2
f51m	8/8	143/277	93/187/328.5	101/198/76.8
i1	25/16	46/85	44/81/3.7	44/81/2.3
lal	26/19	148/301	105/206/611.1	105/203/133.7
pcle	19/9	86/141	86/141/78.3	86/141/18.9
pcler8	27/17	103/174	103/174/166.8	103/174/43.2
pm1	16/13	53/107	49/98/11.2	52/104/3.6
sct	19/15	117/243	80/163/405.8	84/168/122.2
z4ml	7/4	70/130	39/73/10.2	44/79/6.5

(ゲート数 / 結線数 / 計算時間 (秒))

表 2: 手法 1 の実験結果

回路	手法 1-1	手法 1-2
cordic	69/124/7.5	—
f51m	100/197/211.0*	93/187/302.8
i1	44/81/2.7	44/81/3.3
lal	107/202/706.8*	106/209/516.4*
pcle	86/141/42.4	86/141/47.7
pcler8	103/174/85.0	103/174/93.7
pm1	49/98/6.4	49/98/8.4
sct	84/165/218.5*	84/165/293.2*
z4ml	35/66/10.0*	39/73/8.6

(ゲート数 / 結線数 / 計算時間 (秒))

5 あとがき

本稿では、トランスダクション法の一手法であるエラー補償法についてその高速化の手法を含めて述べてきた。手法 2 の利用により、解の品質をあまり落すことなくある程度の高速化を行なうことができた。また手法 1 についても計算時間短縮の有力な方法であることも実証できた。今後はより一層の最適化、高速化を目指すとともに、大規模な回路への適用についても考慮していきたい。

謝辞

SBDD パッケージの使用を快諾していただいた京都大学工学部情報工学教室矢島研究室の皆様に感謝致します。

参考文献

[1] Y.Kambayashi, et al., "NOR Network Transduction Based on Error-Compensation(Principles of NOR Network Transduction Programs NETTRA-E1,NETTRA-E2 and NETTRA-E3)",Report No. UIUCDCS-R-75-737,Dept. of Comp.Sci. Univ.of Illinois, June 1975.

[2] S.Muroga, et al., "The Transduction Method – Design of Logic Networks Based on Permissible Functions",IEEE Trans. on Comp.,Vol.38,No.10, 1989.

[3] S.Minato, et al., "Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation", In Proc. 27th Design Automat. Conf., 1990.

[4] 藤本徹哉、神戸尚志, "許容関数集合に基づく組合せ論理回路の遅延最適化", 情処研報 Vol.90,No.100,90-DA-55, Dec. 1990.