設 計 自 動 化 60－17
（1991．12．13）

疑似許容関数を用いた多段論理最適化

東田 基樹　石川 淳士　平峰 正信　野村 和男　佐藤 晴美　数馬 好和

三菱電機 (株)　カスタム LSI 設計技術開発センター

多段論理式の最適化アルゴリズムとしてトランスダクション法が注目されている。トランスダクション法では、許容関数と呼ばれる回路のドントケアを表した関数を用いて最適化を行なう。従来のトランスダクション法には、最適化能力は弱いが計算時間の短い許容関数CSPFと、計算時間は長いが最適化能力の高い許容関数 MSPF があった。我々は、この２種類の許容関数の長所を合わせ持った許容関数として疑似許容関数を考案した。これを用いた最適化システムを開発し評価を行なった結果、最適化によりリテラル数を平均 28 %削減できた。また、CSPF を用いた最適化に比べても、リテラル数を平均 11%削減することができた。

Multi-level Logic Optimization with Pseudo-Permissible Functions

Motoki Higashida　Junji Ishikawa　Masanobu Hiramine　Kazuo Nomura　Harumi Sato
Yoshikazu Kazuma

Mitsubishi Electoric Corporation,　ASIC Design Engineering Center

A transduction method is one of the algorithm of the strongest multi-level logic optimization. It uses permissible functions which represent Don't Care condition of a circuit. Original transduction alogorithm has two sorts of Permissible Functions: MSPF and CSPF. CSPF has features that computation time is short but optimization capability is weak. MSPF has opposite features, that is, computation time is long but optimization capability is good. We propose new permissible functions, which is called Pseudo-Permissible Functions. It has good features of both CSPF and MSPF ,that is, computaion time is as short as CSPF and optimization capability is as good as MSPF. As experimental results, the optimization with Pseudo-Permissible Functions could reduce literals of initial multi-level logic by 28%, and make a logic with 10% less literals than that with CSPF, on an average.

1.はじめに

我々は論理合成システム SOLDIER の開発を進めている[1]。他の多くの論理合成システムと同様に、本システムでは、Weak-Division アルゴリズム[2]により多段論理を生成している。しかし、Weak-Division アルゴリズムによる論理合成では、特にデータパス系の回路に対する合成回路の品質が悪い。この問題を改善するため、多段論理式中に現れる冗長性を利用した最適化を行なうプログラムの研究開発が行なわれている。このような最適化はブール最適化と呼ばれている。ブール最適化を行なう基本的なアルゴリズムに、カリフォルニア大バークレイ校の MIS2.1[3],2.2[8]、コロラド大ボールダ校の BOLD[4]、イリノイ大のトランスダクション法[5]がある。この中で最も最適化能力が高いと考えられるアルゴリズムがトランスダクション法である[6][9]。我々の開発したプログラムは、このトランスダクション法を改良したプログラムである。

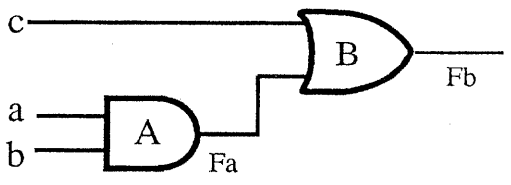
トランスダクションとは、transformation and reduction の略で、回路の変形（transformation）・冗長部分の削除（reduction）を繰り返し行なう多段論理回路の最適化手法である。その最適化処理の概要は、まず各ゲート毎に許容関数と呼ばれる一種のドントケア条件を計算し、その許容関数によって許された範囲で論理の変更（回路の変形・削除）を行ない、回路を最適化するというものである。トランスダクション法では、２種類の許容関数が考案され使用されている。一方は、処理速度は速いが最適化能力が低い、他方は、処理時間はかかるが最適化能力が高いという特徴をもつ[10]。

我々は、この２種類の許容関数の長所を同時に実現できる許容関数として、疑似許容関数を考案した。本プログラムは、この疑似許容関数を用いたトランスダクション法を実現したもので、最適化能力が高く、かつ処理速度も速い。

2. 許容関数

2.1 定義

多段論理回路C中のゲートGにおいて実現されている論理関数を Fg とする。いま、この Fg を他の論理関数 Hg に置き換えても、回路 C のいかなる外部出力の論理関数も変化しないとき、そのような論理関数 Hg をゲート G の許容関数 (Permissible Function:PF) と呼ぶ。図1(a)の回路のゲートA,Bにおいて実現されている論理関数Fa,Fbを図1(b)に示す。ここで、ゲートAの表す論理 Fa を Fa'のように変更しても、外部出力論理（すなわちゲートBの論理Fb）は不変である。従って Fa'はゲートAの許容関数である。許容関数は一般に１つではなく集合となる。例えば、ゲートAの表す論理Faもまたゲートの許容関数である。FaとFa'の違いは、a=1, b=1, c=1のときである。このとき、ゲートの出力が０でも１でも外部出力は変化しないことになる。このような、０でも１でも構わない出力値をドントケアと呼び、＊で



(a)

a b c	Fa Fb	Fa'	PFa
0 0 0	0 0	0	0
0 0 1	0 1	0	*
0 1 0	0 0	0	0
0 1 1	0 1	0	*
1 0 0	0 0	0	0
1 0 1	0 1	0	*
1 1 0	1 0	1	1
1 1 1	1 1	0	*

(b)

図1 許容関数

表す。以後、許容関数の集合をこのドントケア（＊）を用いた不完全記述関数として表すことにする。また、特に断りのない限り許容関数の集合を単に許容関数と呼ぶ。

また、ゲート自体でなくゲート間の接続に対しても、同様にして許容関数が定義される。

トランスダクション法に用いられる許容関数には、2種類のものがある。一つは、自分以外の論理が不変として計算したMSPF(Maximum Set of Permissible Functions)である。これは、各ゲートの許容関数のドントケアとできる部分を全てドントケアとしたものである。MSPFは、次に述べるCSPFに比べ、許容関数の生成に非常に多くの時間を必要とする。もう一つは、他の場所においても論理の変更があるものとして定義されたCSPF(Compatible Set of Permissible Functions)である。CSPFを用いる時、同時に複数箇所の論理の変更を行なっても出力論理関数は不変である。CSPFはMSPFの部分集合であり、トランスダクション法を実行した時の最適化能力が低下する。

2.2 MSPFの計算法

MSPFを求めるアルゴリズムを以下に示す。回路はAND,OR,NOTの論理ゲートから構成されていると仮定する。

[全体のアルゴリズム]

(1) 全てのゲートの論理関数を求める。

(2) 出力ピンのコネクションのMSPFを、その出力論理関数そのものとする。

(3) 出力段のゲートから入力段のゲートへ向かって順次、ゲートのMSPF及びその入力コネクションのMSPFを求める。（ゲート及びコネクションのMSPFの計算法は後述する。）

[ゲートのMSPFの計算法]

(1) ゲートのファンアウトコネクションのMSPFのインターセクションを求め、これをQPFとする。

(2) QPFのドントケアとなる全ての入力パターンについて、それが実際にドントケアであることを確認する。（ドントケアとなる全ての入力パターンについて、ゲートの出力が1でも0でも、出力論理関数が不変であることを確認する）もし、ドントケアでないならば、実際のゲートの論理値に置き換える。

[コネクションのMSPFの計算法]

ゲートGの入力コネクションcのMSPFを求める場合を考える。ゲートGのMSPFをPF(g)とし、ゲートGからコネクションcを取り除いた時のゲートの論理関数をHとする。コネクションcのMSPFは、ゲートの種類に応じて次のように求める。

ORゲートの場合　：関数Hが1となる入力パターンに対する関数値を全てドントケア（＊）に変更したPF(g)

ANDゲートの場合：関数Hが0となる入力パターンに対する関数値を全てドントケア（＊）に変更したPF(g)

NOTゲートの場合：PF(g)の否定

2.3 MSPFの計算例

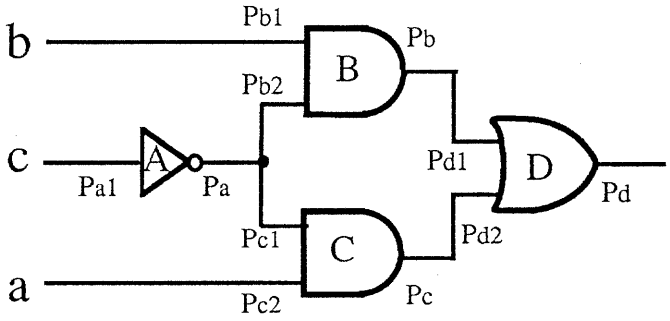
図2(a)の回路に対して、MSPFを計算した結果を図2(b)に示す。ここで、ゲートA～Dの表す論理関数をFa～Fd、MSPFをPa～Pdとする。また、ゲートDの2本のファンインコネクションのMSPFをPd1, Pd2とし、他のコネクションに対しても同様に定める（図2(a)参照）。

問題は、ゲートAのMSPFを求める場合である。このゲートのファンアウトコネクションは2本ある。この2つのコネクションのMSPFであるPb2, Pc1のインターセクションを QPa(=Pb2*Pc1)を図2(b)に示す。関数QPaにはドントケアが3箇所存在する。これらのドントケアの確認を行なう。a,bが共に0の場合の2箇所については、ゲートAの出力に関わらず外部出力は0となりドントケアでよい。次に、(a,b,c) = (1,1,0)の時を考える。外部出力は、ゲートAの出力が1ならば1、0ならば0となる。従って、この時のゲートAのMSPFはドントケアでない。

このように、MSPFを求める時ドントケアを確認する操作が必要である。

3.疑似許容関数をもちいたトランスダクション法

3.1 疑似許容関数



(a)

a b c	F_a	F_b	F_c	F_d	Pd	Pd1 =Pb	Pd2 =Pc	Pb1	Pb2	Pc1	Pc2	QPa	Pa	Pa1
0 0 0	1	0	0	0	0	0	0	0	*	*	0	*	*	*
0 0 1	0	0	0	0	0	0	0	*	*	*	*	*	*	*
0 1 0	1	1	0	1	1	1	*	1	1	*	*	1	1	0
0 1 1	0	0	0	0	0	0	0	*	0	*	*	0	0	1
1 0 0	1	0	1	1	1	*	1	*	*	1	1	1	1	0
1 0 1	0	0	0	0	0	0	0	*	*	0	*	0	0	1
1 1 0	1	1	1	1	1	*	*	*	*	*	*	(*)	1	0
1 1 1	0	0	0	0	0	0	0	*	0	0	*	0	0	1

(b)

図2　疑似許容関数とMSPF

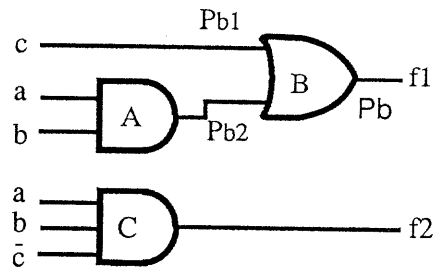
本プログラムでは、許容関数として疑似許容関数と呼ぶ関数を用いている。この関数は、MSPFを求める際のドントケアの確認を行なう前の関数である。すなわち、2.2節に示したゲートのMSPFの計算法において、(2)の処理を省略し、QPFを許容関数としたものである。図2の例における疑似許容関数は、ゲートB～DについてはMSPFと同じPb～Pdとなるが、ゲートAについてはQPaとなる。疑似許容関数は、実際にはドントケアでない入力パターンもドントケアとなっている可能性がある。従って、定義からすると許容関数ではない。そこで、この関数を疑似許容関数と呼んでいる。MSPFは疑似許容関数の部分集合となる。

疑似許容関数はMSPFのもつドントケアを全て含んでいる。従って、この疑似許容関数を用いたトランスダクション法のシステムは、MSPFを用いたものと同等の最適化能力をもつ。一方、疑似許容関数の生成時間はMSPFより大幅に短縮できる。

但し、疑似許容関数は実際にはドントケアでない部分をドントケアにしている可能性があり、これを用いてトランスダクションを行なえば出力論理が変化する可能性がある。これをチェックする処理が余分に必要である。しかし、MSPFを用いたトランスダクション法においても、回路変形の度にMSPFを再生成する必要がある。この時、各ゲートの論理関数と共に出力論理関数が求まるので、出力論理の不変性のチェックはこの時同時に行なえる。従って、処理時間のオーバヘッドはわずかである。

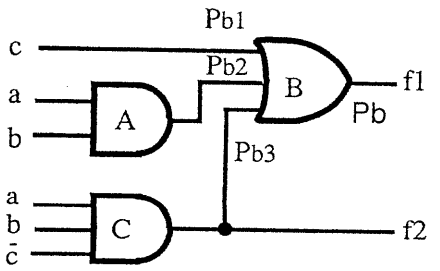
3.2 トランスダクション法

疑似許容関数を用いたトランスダクション法の基本処理は、通常の許容関数を用いたトランスダクション法と同じである。異なっているのは、通常のトランスダクション法の基本処理を行なった後で、回路機能の不変性を確認する処理が必要な点である。トランスダクション法には、いくつかの基本処理が存在する[4]。疑似許容関数を用いたトランスダクション法においても、これらを同様に実行でき



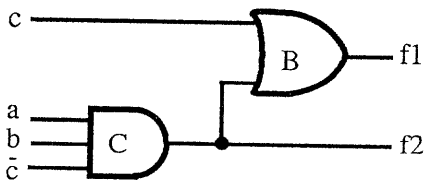
(a) 元の回路

a b c	Fa	Fb	Fc	Pb	Pb1	Pb2
0 0 0	0	0	0	0	0	0
0 0 1	0	1	0	1	1	*
0 1 0	0	0	0	0	0	0
0 1 1	0	1	0	1	1	*
1 0 0	0	0	0	0	0	0
1 0 1	0	1	0	1	1	*
1 1 0	1	1	1	1	*	1
1 1 1	1	1	0	1	*	*



(b) 追加後の回路

a b c	Fa	Fb	Fc	Pb	Pb1	Pb2	Pb3
0 0 0	0	0	0	0	0	0	0
0 0 1	0	1	0	1	1	*	*
0 1 0	0	0	0	0	0	0	0
0 1 1	0	1	0	1	1	*	*
1 0 0	0	0	0	0	0	0	0
1 0 1	0	1	0	1	1	*	*
1 1 0	1	1	1	1	*	*	*
1 1 1	1	1	0	1	*	*	*



(c) 追加・削除後の回路

図 3　疑似許容関数を用いたトランスダクション法

る。ここでは、コネクションの追加・削除 (connectable/disconnectable) を例にして、疑似許容関数を用いたトランスダクション法について説明する。

[基本処理]

ゲートに新たな接続を追加することにより回路の構造を変化させる（コネクションの追加）。これにより、元のコネクションに冗長が生じることがある。これを削除（コネクションの削除）することにより回路の最適化を行なう。コネクションの追加／削除が回路の出力論理に影響を与えるかどうかは疑似許容関数を用いて判断する。つまり、そのゲートに新たな接続を追加したときのゲートの論理関数が疑似許容関数に含まれているときには追加可能である。次に、追加処理後のゲートのファンインコネク

ションの疑似許容関数を再計算する。疑似許容関数が全く0(1)を含まない時、そのコネクションは削除できる。どのように追加するか、どのように削除するか、により最適化の結果が大きく変化する。このための戦略がいくつか提案されている[5]。

[例題]

図3(a)の回路を考える。回路図の隣に疑似許容関数も示してある。ゲートBにゲートCからのコネクションを追加した時のゲートBの論理Fb'は疑似許容関数Pbに含まれる。この時、ゲートBにゲートCからのコネクションを追加しても出力論理f1は不変である。(この例ではFb'=Fbであるが、Fb'⊆Pbであれば Fb'≠Fbでもよい。) 追加を行なった後の回路が図3(b)である。次に、このゲートBのコネクションの疑似許容関数を再計算する。このとき、Pb2が1を含まない。従って、このコネクションは削除できる。コネクションの削除に伴ってゲートAを削除し、最終的に、図3(c)のような回路に変換される。

[アルゴリズム]

(1) 疑似許容関数を作る。

(2) 回路中の各ゲートGに対して、以下の処理を行なう。

(2-1) ゲートGの疑似許容関数に含まれる論理を持つコネクションを全て、ゲートGに追加する。但し、追加によって回路中にループができるコネクションは除く。

(2-2) ゲートGの疑似許容関数を作り直し、元のコネクション、追加のコネクションの順に冗長なコネクションを削除する。

(2-3) 出力論理が不変であることを確認する。もし出力論理が変化するなら、コネクションの追加前の状態に戻し、(2)の処理を他のゲートに対して続ける。

(2-4) コネクションの追加・削除によって、回路の総コネクション数が増大するなら、コネクションの追加前の状態に戻し、(2)の処理を他のゲートに対して続ける。

(2-5) コネクションの追加・削除後の回路に対して、疑似許容関数を再生成する。

(3) もし、(2)の処理で回路の総コネクション数が減少したなら、再度はじめから(2)の処理を行なう。

4.評価

以上のようなアルゴリズムに基づいてプログラムを作成し、評価を行なった。評価データには、論理合成のワークショップ[7]の回路を11種類、社内で開発された実際の回路の論理を3種類用いた。

4.1 リテラル数による評価

論理合成回路の評価は、実際のチップ上にしめる回路面積で評価できれば最善である。しかし、本プログラムのような抽象ゲートレベルの最適化プログラムでは、回路面積を直接評価することは困難である。抽象ゲートレベルでは、多段論理式のリテラル数が回路面積のよい近似となることが知られている[2]。リテラル数による、評価結果を表1に示す。入力はWeak-Divisionアルゴリズムにより生成した多段論理式である。表中Input,Outputは、それぞれ入力と最適化後のリテラル数を表す。また、Out/Inは、最適化によるリテラルの削減率を示す。最適化により、最大83%、最小5%、平均28%のリテラルを削減できた。

4.2 計算時間

トランスダクションの実行にかかった計算時間（ユーザタイム）を、表1のCPU timeの欄に示す。使用計算機はSparcServer/490、単位は秒である。また、処理を終えるまでに疑似許容関数を計算した回数(最適化のループ回数)をLoopの欄に、出力論理の変化するような回路変形を行なった回数をIncorrectの欄に示す。

計算時間は疑似許容関数の生成回数と回路規模に大きく依存する。とくにALU4は、疑似許容関数の生成回数が非常に多く、従って多くの計算時間がかかった。疑似許容関数の生成回数は、コネクションの削減率と大きな相関があり、生成回数が多いほど削減率も高くなる。

表1　実験結果

データ名	疑似許容関数を用いたトランスダクション法							CSPFを用いたトランスダクション法	
	リテラル数			実行ログ					
	Input	Output	Out/In	CPU time	Loop	Incorrect	I.C./Loop	CSPF	Out/Cspf
5xp1	102	89	0.87	8	39	0	0.00	89	1.00
sao2	153	130	0.85	83	76	0	0.00	139	0.94
vg2	87	82	0.94	11	20	0	0.00	82	1.00
bw	211	172	0.82	54	87	0	0.00	188	0.91
duke2	379	319	0.84	536	128	0	0.00	330	0.97
misex1	66	53	0.80	3	35	0	0.00	52	1.02
misex2	108	103	0.95	27	23	0	0.00	107	0.96
misex3	1003	498	0.50	6244	525	0	0.00	469	1.06
misex3c	511	375	0.73	1339	163	0	0.00	421	0.89
alu4	1061	182	0.17	7348	533	0	0.00	448	0.41
clip	161	91	0.57	35	94	0	0.00	127	0.72
melco1	714	409	0.57	1047	394	71	0.18	454	0.90
melco2	612	476	0.78	2123	322	90	0.28	520	0.92
melco3	149	92	0.62	28	79	0	0.00	103	0.89
平均			0.72				0.03		0.90

疑似許容関数を用いたトランスダクション法では、出力論理が変化する恐れがある。この例題では、2例について出力論理が変化した。平均すると3%の回路変形で出力論理が変化する。出力論理が変化することは通常の許容関数を用いたトランスダクション法ではあり得ず、疑似許容関数を用いたトランスダクション法ならではのオーバヘッドである。しかし、この実験結果から分かるように、論理が変化することは稀であり、このための計算時間のオーバーヘッドは問題にならない。また、出力論理の変化は、2つのデータに集中して起こっている。これは、同じ回路変形をループの別の箇所で何度も繰り返して行なっているためである。履歴を残す等の工夫により、出力論理の変化するような回路変形の多くを排除できると考えている。

4.3 CSPFを用いたトランスダクション法との比較

多くのトランスダクション法を実現したプログラムは、許容関数としてCSPFを用いている。従って、ここでは疑似許容関数を用いたトランスダクション法とCSPFを用いたトランスダクション法を比較する。表1のCSPFの欄にCSPFを用いたトランスダクション法による最適化を行なった時のリテラル数を示す。トランスダクション法の基本処理は、疑似許容関数を用いた時と同じものを用いている。また、CSPFを求める時のゲートのコネクションの優先順位は、文献[11]に合わせた。

表中のOut/Cspfの欄に、疑似許容関数を用いた時とCSPFを用いた時の最適化結果のリテラル数の比を示す。疑似許容関数を用いたトランスダクション法は、CSPFを用いたものに比べて平均10%リテラルの少ない回路を生成している。

5. まとめ

多段論理式の最適化アルゴリズムとして有名なトランスダクション法に改良を加え、プログラムを開発した。従来のトランスダクション法には、最適化能力は弱いが計算時間の短い許容関数と、計算時間は長いが最適化能力の高い許容関数があった。我々は、この2種類の許容関数の長所を合わせ持った許容関数として、疑似許容関数を考案した。これを用いたプログラムを開発し、評価をおこなった結

果、合成回路のリテラル数を約27％削減することができた。

参考文献

[1] 平峰、石川、野村、佐藤、数馬、"多段組合せ論理合成システム SOLDIER"、情処研報 90-DA-55, p.p.41-48, (Dec. 1990).

[2] R.K.Brayton, R.Rudell, A.Sangiovanni-Vincentelli and A.R.Wang, "MIS:A Multiple-Level Logic Optimization System," IEEE trans. on CAD, Vol. CAD-6, No.6, pp.1062-1081, (Nov.1987).

[3] K.A.Bartlet, R.K.Brayton, G.D.Hachtel, R.M.Jacoby, R.Rudell, A.L.Sangiovanni-Vincentelli, and A.Wang, "Multilevel Logic Minimization Using Implicit Don't Cares," IEEE Trans. on CAD, Vol.7, pp.723-740,(1988)

[4] D.Bostick, G.D.Hachtel, R.Jacoby, M.R.Lightner, P.Moceyunas, C.R.Morrison and D.Ravenscroft, "The Boulder Optimal Logic Design System", ICCD'87, pp.67-65, (1987).

[5] S.Muroga, Y.Kambayashi, H.C.Lai, and J.N.Culliney, "The Transduction Method -Design of Logic Networks based on Permissible Functions", IEEE Trans. on Computer, vol.C-38, pp.1404-1424,(Oct.1989).

[6] H.Sato, Y.Yasue, Y.Matsunage, M.Fujita, "Boolean Resubstitution with Permissible functions and binary decision diagrams," DAC'90, pp.284-pp.289, (1990).

[7] R.Lisanke, "Logic Synthesis and Optimization Benchmarks", Technical Report , Micro-electronics Center of North Carolina,(Dec.1988).

[8] H.Savoj and R.K.Brayton, "The Use of Observability and External Don't Cares for the Simplification of Multi-Level Networks," Proc.DAC, pp.297-301, (1990).

[9] 松永、藤田、"順序付き２分決定グラフと許容関数を用いた多段論理回路簡単化手法、"信学論(A),Vol.J74-A,No.2,pp.196-205,(1991).

[10] J.C.Limqueco and S.Muroga, "Logic Optimization of MOS Networks," Proc.DAC.,pp.464-469,(1991).

[11] 藤田、"トランスダクション法に基づく多段論理回路簡単化機能をもつ論理合成システムとその評価", 情処論文誌, Vol.30,No.5,pp.613-623, (May 1989).