設計自動化 67－6
（1993．6．25）

# 多段ＮＡＮＤゲート回路の一設計法

松田秀雄　宮腰　隆　畠山豊正

富山大学　工学部
〒930　富山市五福3190番地

論理関数がマップで与えられているとき，(a)真理値１のセルをできるだけ大きな許容項（複数個）で被覆し，(b)もし，これらの許容項の中で真理値０のセルを含むものがあれば，それを取り除くために０を被覆する許容項を生成する･･･というような手順を繰り返す木形構造の多段ＮＡＮＤゲート回路を得る手法を提案している．近似的によい回路を得る方法なので，少変数の関数では厳密解（厳密に最小回路を得る方法）に劣る．しかし，厳密解を得る手法は，直接多変数関数に適用できず，関数を分割して，扱うことになる．この場合，直接適用できる本方法の方が，近似解といっても，全般をにらんだ戦略なのでより簡単な回路が得られることが示される．

# An Algorithm for Finding a Minimal Multi-Level NAND Network

Hideo Matsuda　Takashi Miyagoshi　Toyomasa Hatakeyama

Faculty of Engineering , Toyama University
Gofuku, Toyama-shi, 930　Japan

A tree type algorithm for deriving a multi-level NAND gate network is described. The method presents an approximate least-cost network. In the case of the function with a few variables, the method is inferior to other methods which find strictly a minimum cost network. But the strict minimum method can not directly applied to the function with many variables because of time and memory limit in a computer execution ; therefore there is no other way but to do that we devide the function into smaller variable functions, apply the method to each of functions and then form networks of the original function ; namely, DS method. However, it is shown that our method derive a smaller cost network than the DS method, because our method can directly apply to the function.

## 1．はじめに

ＮＡＮＤ（あるいはＮＯＲ）ゲートのみによる論理回路の設計は構造の一様性から集積回路化に適しており，非常に重要である．とりわけ，一線入力，すなわち，入力信号線として，$x_i$のみを使い，その否定の信号線$\overline{x}_i$を使わない方式の論理回路は，ＩＣパッケージ上のピン数やＩＣチップ上の接続線の減少をもたらすので，研究の価値がある．

本稿では一線入力（以下略）多段ＮＡＮＤゲート回路の一設計法を提案する．方法は論理設計で用いられるマップの概念を使って説明できる．関数がマップで与えられているとする．（ａ）真理値１（true）のセルをできるだけ大きな許容項（一般に複数個）を生成して，被覆する．（ｂ）もし，これらの許容項の中で真理値０（false）のセルを含むものがあれば，それらの許容項についてだけ０を被覆する許容項を生成する（一般に複数個）．もし，これらの許容項のいずれかが，１のセルを含めば，再び（ａ）のようにして，１のセルを被覆する許容項を生成し･･･というように，すべての許容項が１のセルと０のセルの混在しなくなるまで，（ａ）（ｂ）の手順を交互に繰り返す．この手順の結果，許容項の木が形成され，許容項をＮＡＮＤゲートに置き換えると，（１）初期回路が得られる．続いて，（２）回路の接続関係を使って，簡単化を行い，最後に，（３）回路が正しくｆを実現しているかどうかを検証する．

本方法は近似的によい回路を得る方法である．一方で，厳密に最小ＮＡＮＤゲート回路を得る手法[1],[2]もあるが，これらの方法では，考えられ得るあらゆる回路から，最小回路を見つけるもので，記憶容量や計算時間からの制約から，３ないし４変数（入力）までの関数（または回路）に適用するのが限度である．多変数関数を扱おうとすると，関数を分割して，それぞれに部分回路を求め，これらを合成して元の関数の回路を求める手法しかない．これを分割して厳密解と結びつける手法という意味でＤＳ法と呼ぼう．こうすると部分的に最小回路であっても相互に関連がなく，全体として，よい回路（ゲート数が少なく，同じなら，入力線数が少なく，もしこれも同じなら回路の段数が小さいこと）になっているという保証はない．

これに対し，我々の方法は３ないし４変数の範囲では厳密解に劣るがアルゴリズム的に木形構造をもち，自動的に関数を分割しており，多変数関数にこのまま適用できる．この場合，近似解といっても全般をにらんで（許容項を大きく選んで）ゲート数を少なくしようという戦略なので，部分回路相互に連絡があり，ＤＳ法よりよい解が得られることを実験的に示す．

第２章で用語の説明，第３章で本方法の原理，および厳密解と結びつける手法について記述する．第４章で４変数から８変数の関数について，本方法の方が，ＤＳ法よりよい回路が得られることを示す．また，多変数向きアルゴリズムであることを示すため，12および15変数関数のいくつかについて，計算例を示す．

## 2．基礎的事項

（２値）ｎ変数論理関数　$f(x_1, x_2, \cdots, x_n)$　は仮に変数の数が多くても，概念的にはマップで与えられていると考えることができる．ｎ変数のマップはＭ＝$2^n$個のセルよりなり，入力変数$x_1, x_2, \cdots, x_n$がとる（２進数を１０進数に直した）値によって識別できる．図１は４変数関数のマップの例である．本稿では，関数はセル番号に上付き１を付してそのセルでtrueの値，またセル番号に上付き０を付してそのセルでfalseの値をとるように表す．

＜項＞　入力変数の肯定$x_i$(i=1, 2, ･･･, n)およびその否定$\overline{x}_i$をリテラル，重複のないいくつかのリテラルの積を項（キューブ）という．

＜セルと最小項＞　$2^n$個ある各セル（例えば，図１

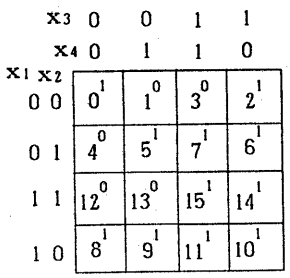



図１　関数の例

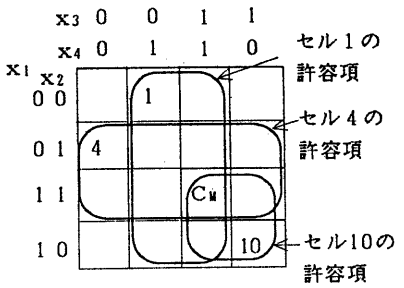



図２　許容項の例

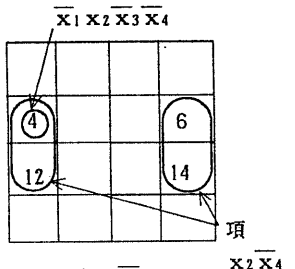



図３　項$x_2\overline{x}_4$の最小の値の最小項

のセル５）のそれぞれにつき，その座標($x_1, x_2, \cdots, x_n$)（セル５の座標は($x_1, x_2, x_3, x_4$)＝(0101)）から，１の成分に対して肯定のリテラル，０の成分に対して否定のリテラルを掛け合わせた項（セル５の場合，$\bar{x}_1 x_2 \bar{x}_3 x_4$）が考えられ，マップ上の唯一つのセルからなる項なので最小項と呼ばれる．変数成分が全部１のセル（最小項）を$C_M$で表す．

＜セルと許容項＞　また，あるセルｊ（j=0,1,$\cdots$, $2^n-1$）の最小項で，否定のリテラルを取り除いてできる項を，そのセルｊの許容項という．４変数の場合で例示すると，セル５の許容項は$x_2x_4$である．４変数ではＭ＝16個のセルがあるから，16個の許容項が考えられる．このうち，セル０は１をとる変数がないのでセル０の許容項はマップ全体を表す項$u_n$とする。セル15の許容項は $x_1x_2x_3x_4$ でセル15そのものである．各許容項はマップ上で表すと，すべてセル15（ｎ変数の場合，セル$2^n-1$，すなわち，セル$C_M$）を含み，セルの座標の１の数をqとすれば$2^{n-q}$個のセルよりなるという性質をもつ．図２にセル１，セル４，セル10の各許容項を示す．

各セル（最小項）のハミング距離を各セルの距離という．特に，セル15（セル$C_M$）とのハミング距離ｄは上記のn-qに等しく，セル15から離れているセル程（含むセルの数で比較して）その許容項は大きい．

＜項の最小の値の最小項＞　例えば，項$x_2\bar{x}_4$において陽に表れていない変数$x_1, x_3$の否定のリテラル$\bar{x}_1, \bar{x}_3$をこの項に掛けた項$\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4$は項$x_2\bar{x}_4$に含まれる最小項のなかで最小の値のものである．これを項$x_2\bar{x}_4$の最小の値の最小項と呼ぶ（図３参照）．

変数の数ｎが多くなると，関数ｆをマップで与えることは事実上困難で，項$t_k$（k=1,2,$\cdots$,m）の和で与えられていると考えた方がよい．

$$f = t_1 + t_2 + \cdots + t_m \qquad (1)$$

＜ディスジョイント・シャープ演算＞　項$t_i$に含まれる最小項の内，項$t_j$にも含まれる最小項を取り除く操作のため，ディスジョイント・シャープ演算が使われ記号$t_i \oplus t_j$で表す．また，関数ｆから項$t_i$に含まれる最小項を取り除く操作は$f \oplus t_i$で行う．ディスジョイント・シャープ演算に関しては文献［３］その他でよく使われているので，ここでは説明を省略する．

＜関数の否定＞　関数ｆがマップで与えられている場合には，ｆの否定$\bar{f}$は１のセルを０に，０のセルを１に変えるだけで得られるが，関数が式（１）のように項の和で与えられ，かつ，変数の数が多い場合には効率のよい否定を求めるアルゴリズムが必要で，ここでは笹尾の方法[4]を我々なりに改良した手法を用いる．同じ関数を実現するＮＡＮＤゲート回路のうち，ゲート数がより少なく，もし同じなら，入力線数がより少なく，これも同じなら，段数がより少ない回路を求めることをここでは簡単化という．

## ３．方　法

### 3.1　初期回路

関数$F = t_1 + t_2 + \cdots t_m$が与えられたとしたら，初期回路を求める手順は，まず，Ｆの否定$\bar{F}$を求めておいて，ｆ＝Ｆとおき，次の手順ＩＮＡＮＤ（ｆ，ｒ，ｅ）を呼べばよい．但し，初め，ｒ＝１，ｅ＝１とする．また，Ｆ*はｅが偶数ならＦ，ｅが奇数なら$\bar{F}$を，それぞれ，表すものとする．Ｆについては，値１（true）をとるセル（または最小項），$\bar{F}$については値０（false）をとるセルについて，それぞれ，注目し，Ｆの場合には値１（true）をとるセルがないとき，$\bar{F}$の場合には値０（false）をとるセルがないとき，それぞれ，関数Ｆおよび$\bar{F}$は$\phi$（空）であるといおう．

『　　ＩＮＡＮＤ（ｆ，ｒ，ｅ）

［許容項の生成法］によりｍ個の許容項 $P_r, P_{r+1}, P_{r+2}, \cdots P_{r+m-1}$が生成したとする．各許容項$P_k$（k=r,r+1,r+2,$\cdots$,r+m-1)について，以下の操作を行う．

$f_k = P_k \cap F^*$

$f_k$が$\phi$（空）なら　ＲＥＴＵＲＮ

$f_k$が$\phi$でなければ，ｒ＝ｒ＋ｍ，ｅ＝ｅ－１とおいて，

ＩＮＡＮＤ（$f_k$，ｒ，ｅ）を呼ぶ．

但し，

［許容項の生成法］　ｊ＝ｒとおく．

１）$f = t_1 + t_2 + \cdots t_m$　とし，各項$t_i$(i=1,2,$\cdots$,m)の最小の値の最小項$t_{im}$を求める．$t_{im}$のうち$C_M$との距離が最大なもの（複数個あれば，値の小さい方の最小項）を選んで$T_m$とする．

２）セル$T_m$の許容項を生成する．これを$P_j$とする．$f \oplus P_j$の結果の関数をあらためてｆとする．ｆが$\phi$なら許容項の生成はこれで終る．そうでなければ，ｊ＝ｊ＋１と　して，ステップ１）へいく．　』

［許容項と回路との関係］

上の網掛け部分でのＩＮＡＮＤの呼び出しを許容項$P_k$についての呼び出しといおう．出力ＮＡＮＤゲートを$P_0$とする．最初のＩＮＡＮＤの呼び出しで生じた許容項，例えば$P_1, P_2, P_3$は$P_0$への入力ゲートとなり，

仮に$P_1$についての呼び出しで$P_4$, $P_5$が生じたものとすれば，$P_4$, $P_5$が$P_1$への入力ゲートになると考える．但し，各ゲート$P_k$へはその許容項を構成するリテラル，あるいは，変数が入力線として加わる（図４）．これを入力リテラル，または単に，そのゲートへのリテラルといおう．

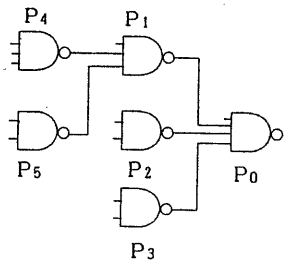



図４　許容項と回路の関係
（変数入力省略）

[例　題]　図１の関数でみる．上付き１のセルでtrueなので，関数は最小項で与えられているとみなす．このような場合，各項$t_i$(i=1,2,･･･,m)の最小の値の最小項$t_{1m}$は$t_1$そのものとなる．最初のＩＮＡＮＤの呼び出しでの[許容項の生成]は次のようになる．図１では$C_M$（セル15）との距離が最大な最小項はセル０である．よってセル０の許容項$u_0$を生成する，$P_1=u_0$（図５（ａ）太線枠内）．$P_1$はマップ全体なので，$f \oplus P_1$は$\phi$となり，許容項は$P_1$ただ一個だけ生成される．そこでＩＮＡＮＤの網掛け部分に進んで，$f_1=P_1 \cap \overline{F}$（いまの場合ｅ＝１なので$F^e=\overline{F}$），$f_1$は（図５（ｂ）上付き０のセルがあるので）$\phi$でない．それで，ｒ＝２，ｅ＝２としてＩＮＡＮＤ（$f_1$，ｒ，ｅ）を呼ぶ．この呼び出しでの[許容項の生成]はセル１が$C_M$から最も距離が離れており，$P_2=x_4$をまず生成し，$f_1 \oplus P_2$の結果，$f_1$のfalseのセルのうちセル$4^0$とセル$12^0$が残り，セル４の許容項$x_2$が$P_3$として得られる．$(f_1 \oplus P_2) \oplus P_3$が$\phi$となり，この呼び出しでの許容項の生成は$P_2$，$P_3$の二つで終わる（図５（ｂ））．$f_2=P_2 \cap F \neq \phi$なので$P_2$についてＩＮＡＮＤの呼び出し（ｒ＝３，ｅ＝３）を行う．図５（ｃ）からわかる通り，このときの呼び出しで，$P_4=x_2x_4$, $P_5=x_1x_4$の二つの許容項が生成される．$P_4 \cap \overline{F} \neq \phi$なので$P_4$についてＩＮＡＮＤの呼び出し（ｒ＝５，ｅ＝４）を行い，$P_6=x_1x_2x_4$が生じ（図５（ｄ）），ここでも$P_6 \cap F \neq \phi$なので，更に$P_6$についてＩＮＡＮＤの呼び出し（ｒ＝６，ｅ＝５）を行い，$P_7=x_1x_2x_3x_4$が生成される(図５（ｅ）)．

ここでようやく，$P_7 \cap F=\phi$で，ＲＥＴＵＲＮ．一つ上の手順（図５（ｄ））に戻って，ここでもＲＥＴＵＲＮ．以下，図５（ｃ）の$P_5 \cap \overline{F}=f_5$についても$P_4 \cap \overline{F}$について行ったと同様の手順を繰り返していき，更に再帰的手順ＩＮＡＮＤがつきるまで行うと，結局，本例題では図６の木で示される形で許容項が生成される．但し，各根にある許容項についての呼び出しで，それぞれの根に直接枝でつながっている許容項が生成される．なお，上で述べた$P_1$から$P_7$までの許容項導出過程が図６の一番左の枝になっている．その他の許容項は$P_8=x_1x_2x_4$, $P_9=x_1x_2x_3x_4$, $P_{10}=x_2x_4$, $P_{11}=x_2x_3$, $P_{12}=x_1x_2x_4$, $P_{13}=x_1x_2x_3x_4$である．

（例題終わり）

先述の[ 許容項と回路との関係 ]，あるいは，図６の木から図７の初期回路が得られる．許容項１個にＮＡＮＤゲート１個が対応するのでゲートを許容項そのもの，あるいはその添字で番号付けして呼ぼう．図７では$P_1$の右先に出力ゲート$P_0$が加わっている．左端の$P_7$や$P_8$のように他のゲートからの出力が接続されておらず，入力リテラル$x_1, x_2, \cdots, x_n$のみが接続されているゲートを入力端ゲートといおう．但し，簡単にするため図７ではリテラル$x_1$, $x_2$などをその添字だけで表している．

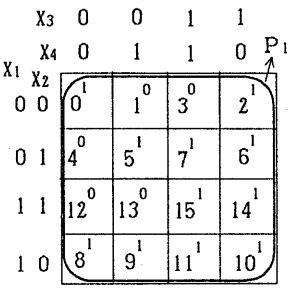
### 3.2　回路の簡単化

ここでは回路接続関係だけを使って簡単化を行う．

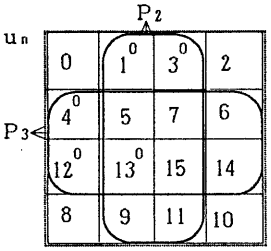
［ゲート数の削減］　入力リテラル（一般に複数個）および入力ゲート（一般に複数個）が同じであるゲートは一つにまとめるというのが削減の基本原理であるが，問題はそれをどのような方法で行うかというところにある．ここでは，まず入力端ゲートについてリテラルが同じものをグループ別に分け，それぞれのグループ内のゲートをそのグループ内で一番小さい番号（各ゲートには対応する許容項の添字で番号付けしている）のゲートで置き換えて接続するよう初期回路を変える．図７の場合には，ゲート７，９，13，11が入力端ゲートで，７，９，13が同じ入力リテラルをもつので，回路上，ゲート９，13を７でおきかえる．

以後，このように回路を変更するごとに，入力リテラルおよび入力ゲートが同じであるゲートが生ずれば，それらをグループ別にし，それぞれのグループ内のゲートをそのグループ内で一番小さい番号のゲートで置き換えて接続するよう回路を変更し，入力リテラルおよび入力ゲートが同じであるゲートが生じなくなるまで続ける．

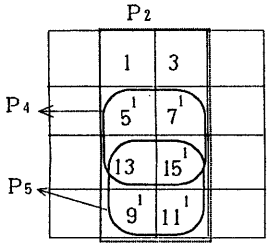
図７の例ではゲート６，８，12の入力リテラルが同じ，$x_1, x_2, x_4$で入力ゲートも７で同じく（ゲート９，13は前の手順で７に変わっている），ゲート８，12を６に置き換えることができる．その結果，ゲート４と10もゲート４一つにまとめれる．しかし，これ以上，ゲートを減らすことはできないので，ゲート数の削減の手順はこれで終わる．図８が図７のように簡単化される．



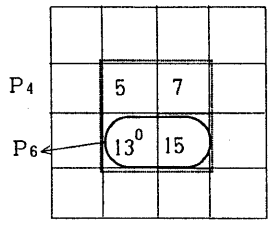

(a) セル0の許容項 $u_n$



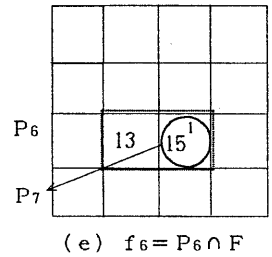

(b) $f_1 = P_1 \cap \overline{F}$




(c) $f_2 = P_2 \cap F$




(d) $f_4 = P_4 \cap \overline{F}$




(e) $f_6 = P_6 \cap F$

図5　手順INANDによる許容項の生成の例

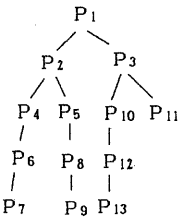



図6　許容項の生成木

［入力線数の削減］　入力線数の削減の基本的考え方は，ゲートIにゲートJが入力ゲートとして接続されているとき，ゲートIの入力リテラル（の全部またはその一部）をゲートJの入力リテラルから削除できるということである．

一般にゲートは図9のように複雑に接続し合っている．同図でゲート$J_1, J_2, J_3$はゲート$I_1$の入力としてつながっているが，この状況をゲート$J_1, J_2, J_3$はゲート$I_1$の子供であるといおう．また，$J_1$の出力が$I_1, I_2, I_3$の入力としてつながっているが，この状況をゲート$J_1$の親が$I_1, I_2, I_3$であるといおう．このとき，リテラル削減の規則は，ゲート$J_1$の親が$I_1, I_2, I_3$であるとき，親の入力リテラルに共通部分があれば，その全部または，一部を（$J_1$も入力リテラルとして含めば）$J_1$から取り除けるということになる．

この場合も，この規則をどのような順序で適用する

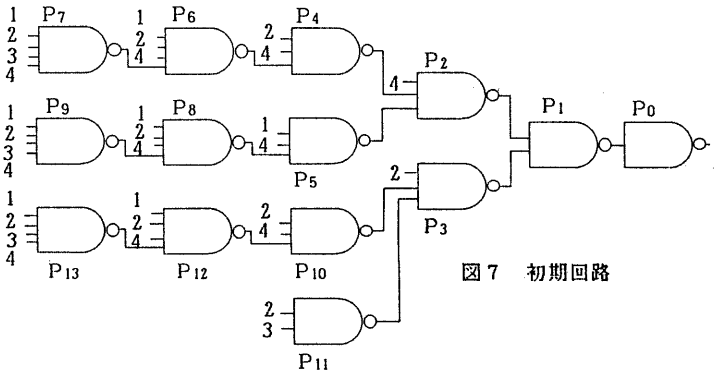



図7　初期回路

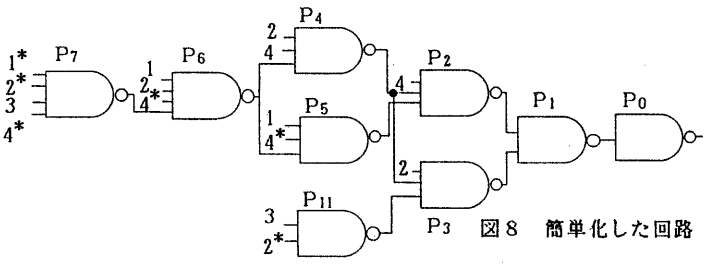



図8　簡単化した回路

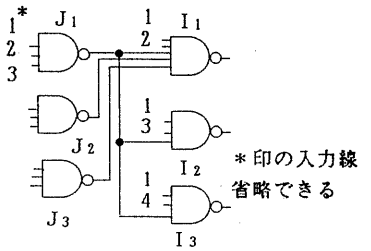



図9　入力線削減規則のための図

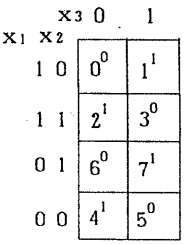



図10　3変数の関数例

かが問題である．ここでは，入力リテラルを減らす次の手順ＤＥＬＩ（$P_i$）を，はじめに，$P_i$を出力ゲート$P_0$とおいて，呼んでいる．

『　　ＤＥＬＩ（$P_i$）

$P_i$が入力端ゲートなら，その親を求め，入力リテラルが削減できる条件を満たすなら，$P_i$のリテラルを減らす．そして，ＲＥＴＵＲＮ．

$P_i$が入力端ゲートでないなら，その子供のリスト｛$P_{i1}, P_{i2}, \cdots, P_{iq}$｝（番号の小さい順になっているとする）を作り，その順にそれぞれ親を求め，入力リテラルが削減できる条件を満たすなら，リストのゲートの入力リテラルを減らす．

そして，ゲート$P_{i1}, P_{i2}, \cdots, P_{iq}$の順に

ＤＥＬＩ（$P_{ij}$）

を呼ぶ（j=1, 2, ･･･, q）．但し，一度ＤＥＬＩの呼び出しを行った$P_{ij}$については二度以上呼び出す必要はない．』

このことを図８の例で説明する．ＤＥＬＩ（$P_0$）で，出力ゲート$P_0$の子のリスト｛$P_1$｝が作られるが，$P_0$および$P_1$にはリテラルがないので，$P_1$のリテラルは削減できない．続いて，ＤＥＬＩ（$P_1$）では$P_1$の子供のリスト｛$P_2$, $P_3$｝が作られる．$P_2$も$P_3$もそれらの親$P_1$にリテラルがないので，リテラル削減の規則は適用できない．次に(1)，｛$P_2$, $P_3$｝のうち，$P_2$でＤＥＬＩを呼ぶ．$P_2$の子供のリストは｛$P_4$, $P_5$｝である．(2) $P_4$の親は$P_2$, $P_3$であるがこれらの間には共通のリテラルがないので，$P_4$のリテラルは減らない．次に，(3) $P_5$の親を調べると，$P_2$唯一つ，$P_2$のリテラル$x_4$は$P_5$のリテラルにもなっているので，$P_5$のリテラルから$x_4$が省略できる（但し，省略できるリテラルは最終的には消すがリテラル削減の手順がひとわたり終わるまでは活かして使う）．更に，(4)ＤＥＬＩ（$P_4$）で$P_4$の子供のリスト｛$P_6$｝が作られる．$P_6$の親は$P_4$, $P_5$である．$P_4$, $P_5$の間には$x_4$が共通のリテラルとして含まれるので，$P_6$のリテラルから$x_4$が削除できる．更に，(5) ＤＥＬＩ（$P_6$）で子供のリスト｛$P_7$｝ができ，$P_7$の親を，$P_6$リテラルをみると，$P_7$のリテラル$X_1, X_2, X_4$が省略できることがわかる．$P_7$は入力端ゲートなので，ＤＥＬＩはこれ以上深く呼ばれることなく，ＲＥＴＵＲＮして，上の(4)のリスト｛$P_4$, $P_5$｝まで戻る．(4)はリスト中，$P_4$についてまず，ＤＥＬＩ（$P_4$）したもので，今度は$P_5$についてＤＥＬＩ（$P_5$）する．$P_5$の子供のリスト｛$P_6$｝が再び作られるが，今度は調べる必要がない．(1)のリスト｛$P_2$, $P_3$｝の$P_2$の次のゲート$P_3$に戻って，ＤＥＬＩ（$P_3$）を呼ぶ．$P_3$の子供のリスト｛$P_4$, $P_{11}$｝が作られるが，これらについて規則が適用できるかどうかを調べる．$P_4$については(2)以下ですでにみているので，ここでは$P_{11}$だけを調べる．$P_{11}$のリテラル$X_2$が省略できる．$P_{11}$は入力端ゲートなので，ＤＥＬＩはこれでＲＥＴＵＲＮ．しかし，手順はこれですべて尽くされているので，終わる．図９の＊印の入力線が省略できる．

回路の簡単化は上記二つの方法を交互に使って，これ以上簡単化されなくなるまで繰り返し行う．

ゲート数の削減を優先する立場からは［ゲート数の削減］の手順をはじめに適用した方がよいように考えられるが，関数によっては，たまに，［入力線数の削減］の手順から先に適用した方がよい回路が得られということもある．

更に，今回は［入力線数の削減］でリテラルの削減ができれば，削減可能な最大個数のリテラルを取り除くという方法を取ったが，少し少な目に削減した方が全体としてよりよい回路が得られるということもあり，回路の簡単化の問題は単純ではない．

### 3.3　回路の検証

合成された回路が与えられた関数Ｆを正しく実現しているかどうかを検証しなければならない．特に多段ＮＡＮＤゲート回路では出力ゲートがどのような関数を表わしているかは，入力端ゲートから出力ゲートに向かって，順に計算してこなければ，わからない．下記の再帰的手順ＶＥＲＩ($P_i$)はゲート$P_i$の出力を計算するプログラムで，はじめに，$P_i=P_0$とおいて，ＶＥＲＩを呼び出すと，$P_0$の出力関数$F_0$が求まる．Ｆの否定$\overline{F}$と$F_0$との和がトートロジィになるなら，Ｆと$F_0$とは一致するという手法で検証を行っている．式（２）を計算するために，高速な否定を求めるプログラムが要求される．

『　　ＶＥＲＩ($P_i$)

$P_i$の子供のリスト｛$P_{i1}, P_{i2}, \cdots, P_{iq}$｝（番号の小さい順になっているとする）を作る．リストのゲートのすべての出力が算出されておれば，$P_i$の出力；

$$P_i=\overline{P_{i1}\cdot P_{i2}\cdots P_{iq}\cdot x_{j1}\cdot x_{j2}\cdots x_{js}}. \qquad (2)$$

を計算する．但し，$x_{j1}\cdot x_{j2}\cdots x_{js}$は$P_i$の入力リテラルとする．そして，ＲＥＴＵＲＮ．

もし，リストのゲートの中に出力が算出されていないものがあれば，それら｛$P_{i1}', P_{i2}', \cdots, P_{iq}'$｝の中からゲートを一つずつとりだして，それを仮に$P_{ij}'$として，

ＶＥＲＩ($P_{ij}'$)

を呼ぶ. 』

## 4．厳密解と結合した手法

我々の方法では図6の例のように，許容項の木を生成する．手順ＩＮＡＮＤでは，許容項$P_k$が得られると，$f_k=P_k \cap F$ すなわち，$P_k$に含まれる関数Ｆまたはその否定のみを使って，更に，ＩＮＡＮＤの呼び出しで許容項が生成できるものなら，生成し続けるので，「ある節点の許容項はその子として連なる許容項より大きい（セルの集合の包含関係で比較して）」という性質がある．例えば，図6の木の許容項$P_1=u_0$はその子の許容項$P_2=x_4$および$P_3=x_2$より大きい（図5(a),(b)）．更に，$P_2$および$P_3$は，それぞれ，それらの子$P_4$，$P_5$および$P_{10}$，$P_{11}$より大きい．このように，本方法は関数をより小さな関数に分割していく性質をもっており，多変数の関数に適用して，厳密解が用いられるまでに，関数$f_k$が小さくなったところで，厳密解を使うという方法（ＤＳ法）に応用できる．厳密解が適用できる変数の数を$n_0$とする．ＤＳ法とは，我々の方法で，網掛けの部分を次の『』中に変える．

『 $f_k=P_k \cap F$

$f_k$（あるいは$P_k$）が$n_0$変数以下の関数なら，厳密解で部分回路を選び，そのゲート数がmなら，

$r=r+m$ として，ＲＥＴＵＲＮ．

$f_k$が$n_0$変数より大きい関数ならば，$r=r+m$，$e=e+1$とおいて，

ＩＮＡＮＤ($f_k$, r, e)を呼ぶ. 』

## 5．結　果

Hellermanは，すでに，３変数関数について，最小ＮＡＮＤゲート回路をカタログとして与えている[1]．また，後藤は最近４変数関数程度なら，最小回路に近い回路が得られる方法[2]を発表している．例えば，３変数関数Ｐ同値類代表関数68個について，Hellermanのカタログと照らし合わせて32個の回路が一致し，最小回路にならなかったものについても，最小になるよう縮約の方法を吟味している．我々は，後藤の方法にこの縮約の方法と，更に，独自の改善の手法も組み込んで，上記，３変数68個の関数中58個が最小回路と一致する自作プログラムを持っている．本章ではこの我々の作った後藤の方法を単に後藤の方法と表し，本稿で提案の手法を本方法と呼ぶことにする．

３変数関数のマップのセル番号は図10に示したとおりで，この順序で真理値を右より左に並べた２進数を10進数に換算した大きさで関数に番号をふろう．例えば，関数 00000000 はNo.1，関数 11111111 No.256，図10の関数 10010110 はNo.150の関数ということにする．表１はこのように表した13個の関数につき，後藤の方法と本方法をHellermanの最小回路と比較したものである．但し，Ｇはゲート数，Ｉは入力線数，ＳはＮＡＮＤゲート回路の段数である．後藤の方法では11個の関数で一致し，本方法では全部の関数で，Ｇ，Ｉ，Ｓのいずれかが多くなっている．

次に，変数の大きいところで，本方法と厳密解と結びつけた方法との比較を行う．厳密解として，Hellermanのカタログを使かった方法をＤＳ法１($n_0$)，後藤の方法を用いた手法をＤＳ法２($n_0$)と称することにする．但し，$n_0$は厳密解を結びつける変数の数である．

表２は４変数から８変数の関数について，きめられた真理値表濃度になるように，trueの値をとる最小項を乱数を使って発生させた50個の関数のＧ，Ｉ，Ｓおよび計算時間（CPU TIME）の各方法の平均値である．

本方法の３章で述べた［許容項の生成法］では同じ（リテラル）許容項が何度も番号を変えて表れる．一度表れた許容項は以後何度表れようと，初めの番号のものを用いれば，初期回路が簡単になる．これを同じ許容項を生成しない方法と呼ぼう．表２の本方法およびＤＳ法２は同じ許容項を生成しない方法になっている．ＤＳ法１（$n_0$=3）では同じ許容項を生成しない方法にすると，変数の数が６，７および８の関数で50個中１個だけ誤りのある回路を生じたので，同じ許容項の生成を許す方法となっている．したがって，初期回路のゲート数が多くなり，簡単化した回路のゲート数も，ＤＳ法２（$n_0$=3）より多く，特に，８変数では記憶量不足で計算不能となった．

表１　３変数関数での比較

| 関数番号 | Hellermanの最小回路 G | I | S | 後藤法 G | I | S | 本方法 G | I | S |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 6 | 11 | 3 | | | | 6 | 12 | 5 |
| 30 | 5 | 8 | 4 | | | | 6 | 10 | 5 |
| 50 | 5 | 7 | 4 | | | | 5 | 8 | 5 |
| 70 | 5 | 7 | 4 | | | | 5 | 8 | 5 |
| 90 | 6 | 11 | 4 | 7 | 14 | 4 | 7 | 15 | 5 |
| 110 | 7 | 14 | 4 | 7 | 16 | 4 | 7 | 17 | 5 |
| 130 | 6 | 11 | 3 | | | | 6 | 13 | 4 |
| 150 | 5 | 10 | 3 | | | | 5 | 11 | 4 |
| 170 | 7 | 12 | 4 | 7 | 13 | 3 | 7 | 15 | 4 |
| 190 | 6 | 11 | 4 | | | | 7 | 12 | 4 |
| 210 | 5 | 8 | 3 | | | | 6 | 10 | 4 |
| 230 | 6 | 10 | 4 | | | | 6 | 11 | 4 |
| 250 | 6 | 10 | 3 | | | | 6 | 12 | 4 |
| 平均 | 5.8 | 10.0 | 3.6 | 5.8 | 10.5 | 3.5 | 6.1 | 11.1 | 4.5 |

空欄は最小回路に一致

本方法の方がＤＳ法１，ＤＳ法２より，ゲート数Ｇおよび入力線数Ｉの小さい回路が得られる．

本方法は近似的によい回路を求める手法であるため，表１のように，変数の数が小さい関数では厳密解よりゲート数の多い回路となるが，表２のように，変数の数が多い関数になると，厳密解と結びつけた手法よりゲート数が相当少ない回路が得られる．

表３は12変数と15変数の関数について，それぞれ，２例ずつの計算結果である．極端に真理値濃度が小さい場合と大きい場合で，かつ，変数の数が多いので，厳密解が適用される前に，本方法の部分で回路が求まってしまうのが，本方法とＤＳ法２に差がない理由と考えられる．ＤＳ法１もごくまれにしか起こらない誤りを恐れなければ，同じ許容項を生成しない方法でやればよく，この場合，初期回路のゲート数が減って，ＤＳ法２の結果と余り変わらなくなる．ここでは関数は次のようにして発生している．ｎ変数の項でｉ番目リテラルが$x_i$または$\overline{x}_i$である確率を$d_1$，陽に表れない確率を$d_2$（$d_1+d_2=1$）としたとき，（$d_1$,$d_2$）を成分濃度と呼ぼう．この場合，（$d_1$,$d_2$）＝（1,0）とすれば，最小項が発生できるし，（$d_1$,$d_2$）＝（0,1）とすれば，マップ全体を表す項$u_n$が生成でき，（$d_1$,$d_2$）の値を適当に組み合わせて各種の項が発生できる．表中の関数の成分濃度はこれを表し，そのような項をいくつ発生したかを示すのが項数である．なお，真理値表濃度は関数を分離加法形に直す手持ちのプログラム[3]によって，計算している．使用計算機はＩＢＭ 3081-KX4（富山大学情報処理センター：ユーザ使用領域８Ｍバイト）である．

## ６．まとめ

本方法は近似的によいＮＡＮＤゲート回路を得る手法であるため，３ないし４変数関数では厳密解よりゲート数などが多くなることがある．しかし，変数の数が多くなると，近似解といっても，全般をにらんだ戦略なので，厳密解と結びつけた手法より，ゲート数の少ない回路が得られることを示した．プログラムは大ざっぱなもので，各変数に割当可能な記憶量などを検討して，今後，更に，多変数関数向きに改善したい．

## 参考文献


[１] Hellerman,L.:A Catalog of Three-Variable Or-Inverter and And Inverter Logical Circuits ,IEEE Trans.Electron.Comput.,Vol.EC-12,No.3,pp.198-223(1963).

[２] 後藤：禁止用ループの使用による一線入力ＮＡＮＤゲート回路の生成法の一手法，情報処理学会論文誌，Vol.30,No.5,PP.624-631(1989).

[３] 松田，宮腰：論理式を分離加法形式で表現する一手法，情報処理学会論文誌，Vol.33,No.4,PP.560-569(1992).

[４] Sasao,T.:An Algorithm to Derive the Complement of a Binary Function with Multiple-valued Inputs,IEEE.Trans.Comput.,Vol.C-34,No.2,pp.131-140(1985).


表２　厳密解と結びつけた方法との比較

| 変数の数 | ＤＳ法１（$n_0$=3） G | I | S | TIME | ＤＳ法２（$n_0$=3） G | I | S | TIME | ＤＳ法２（$n_0$=4） G | I | S | TIME | 本方法 G | I | S | TIME |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 7.1 | 16.1 | 3.7 | 0.03 | 7.6 | 16.6 | 3.6 | 0.10 | 7.6 | 16.5 | 3.6 | 0.10 | 6.4 | 15.3 | 3.8 | 0.02 |
| 5 | 17.1 | 46.4 | 4.9 | 0.08 | 18.4 | 47.4 | 4.9 | 0.25 | 17.5 | 46.5 | 4.5 | 0.91 | 14.1 | 41.4 | 5.1 | 0.07 |
| 6 | 41.1 | 126.2 | 6.0 | 0.32 | 39.6 | 117.4 | 6.0 | 0.61 | 40.5 | 121.1 | 5.7 | 2.26 | 28.7 | 98.1 | 6.1 | 0.22 |
| 7 | 114.4 | 390.2 | 7.1 | 2.18 | 95.1 | 318.3 | 7.1 | 1.92 | 110.9 | 368.2 | 6.9 | 8.01 | 64.6 | 260.9 | 7.2 | 0.90 |
| 8 | — | — | — | — | 197.7 | 765.9 | 8.0 | 5.67 | 264.8 | 765.9 | 8.0 | 23.68 | 137.3 | 637.2 | 8.1 | 3.50 |

各変数とも真理値表濃度を0.05から0.95まで0.1刻みに変えて，各５個ずつ，乱数を使って発生させた50個の関数の平均数
Ｇ：ゲート数，Ｉ：入力線数，Ｓ：段数，TIME：SEC

表３　変数の大きい関数での計算例

| 変数の数 | 成分濃度（$d_1$:$d_2$） | 項数 | 真理値表濃度 | ＤＳ法１（$n_0$=3） G | I | S | TIME | ＤＳ法２（$n_0$=4） G | I | S | TIME | 本方法 G | I | S | TIME |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 0.985 :0.015 | 65 | 0.0023 | 253 | 1342 | 7 | 18 | 253 | 1342 | 7 | 18 | 252 | 1334 | 7 | 19 |
| 12 | 0.240 :0.760 | 8 | 0.8887 | 23 | 92 | 8 | 1 | 23 | 92 | 8 | 1 | 20 | 84 | 6 | 1 |
| 15 | 0.985 :0.015 | 65 | 0.0023 | ＊1— | — | — | — | 143 | 1150 | 7 | 22 | 143 | 1150 | 7 | 23 |
| 15 | 0.985 :0.015 | 8 | 0.8887 | ＊2— | — | — | — | 67 | 252 | 10 | 6 | 67 | 252 | 10 | 6 |

＊1：初期回路のゲート数2590と多すぎて求まらず．＊2：初期回路のゲート数1308と多すぎて求まらず．TIME:SEC