設計自動化 67－7
（1993．6．25）

三段ＮＡＮＤゲート回路の一設計法

宮腰 隆　大澤一人　松田秀雄　畠山豊正

富山大学　工学部
〒930 富山市五福3190番地

三段ＮＡＮＤゲート回路の（ゲート数，入力線数）最小化設計の一手法が提案される．初めに，Ｐ許容項からＮ許容項を打ち抜くという考えに基づくＰ－Ｎ項法が述べられる．ついで，多段ＮＡＮＤゲート回路を三段ＮＡＮＤ回路に直し，これを初期回路として，二段ゲートの個数を減らすために，Ｐ許容項の拡大，三段ゲートの個数を減らすために最小被覆表を使って，回路を簡単化するＭＡ３法が述べられる．ＭＡ３法は原理的にＰ－Ｎ項法と同じであるが，多変数の関数に適用される．ＭＡ３法は４変数関数で後藤の方法と比較しゲート数でほぼ同等の回路が得られることが示される．また，９変数までの関数なら全部，10変数の関数については真理値表濃度が0.55までなら計算できることが示される．

An Algorithm for Finding a Minimal Three-Level NAND Network

Takashi Miyagoshi　Kazuto Oosawa　Hideo Matsuda　Toyomasa Hatakeyama

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

A method for the logical design of minimal three-level NAND gate network is proposed. First the P-N term method which is based on the idea cutting out N(egative permissible) terms from a P(ositive permissible) term is explained. Then the MA3 method being improved to apply to more variable functions is described. In the method, a multi-level NAND network is transformed to a three-level NAND network, P-terms are expanded to reduce the number of the second level gate, and a minimum cover table is used to reduce the number of input gates. The MA3 method is able to find the network for the whole of the function up to 9 variables and 10 variable functions which the truth-table-density are less than 0.55.

1．はじめに

ＮＡＮＤゲートのみによる論理回路はＩＣ化に適しており，その設計法の研究は重要である．とくに，三段ＮＡＮＤゲート回路は，否定の入力線が使われないとした場合でも最小限三段あれば，どのような関数も実現できるので，伝播時間が小さいという点で興味がもたれ，古くはMaleyのヒューリスティックな方法[1]や，GimpelのCC-tableによる方法[2]があり，最近では後藤の行列法[3]が発表されている．

我々も1976年頃に[4][5]，Ｐ（許容）項からＮ（許容）項を打ち抜くという考え方に基づくＰ－Ｎ項法を発表している．しかしながら，これらの方法はいずれにしても，計算機メモリや計算時間の制約から４ないし５変数程度の関数しか扱えない．

一方で，近年ＮＡＮＤ一面形のＰＬＤも現れ，より多変数の関数に適用できる設計法も必要となってきている．本稿では，まずＰ－Ｎ項法について詳述する．そして，多変数の関数に適用する場合に予想される難点を挙げて，それを解決する手立てとして，多段ＮＡＮＤゲートから三段に変換し，これを初期回路として簡単化する手法を述べる．これをＭＡ３法と略そう．ＭＡ３法はＰ－Ｎ項法の基本原理を忠実に踏襲している．ＭＡ３法は近似的によい回路を得ることを目的としているが，４変数関数で後藤の方法と比較しても，回路のゲート数はほとんど一致することを示す．また，変数の数がより大きい関数の適用例として，ランダム関数で９変数の全部，および10変数関数の一部まで計算した結果も掲げる．

2．Ｐ－Ｎ項法による三段ＮＡＮＤゲート回路の論理設計

2.1 方法

本稿を通じて，回路入力変数として否定変数を許さ

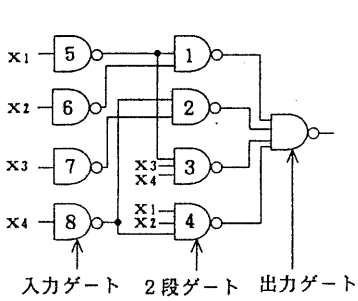

図１　三段ＮＡＮＤゲート回路

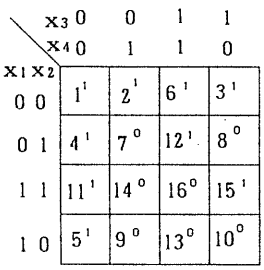

図２　マップのセル番号と関数Ｆ₁の例

ないものとする．また説明の便宜上，４変数の例について話を進める．任意のブール関数は図１のような三段ＮＡＮＤゲート回路で実現できる．右より出力ゲート，二段ゲート，入力ゲートという．このような三段ＮＡＮＤゲート回路では，等価的に出力ゲートはＯＲゲートとして，また二段ゲートはＡＮＤゲートとして働く．従って，ある関数が図２のマップ（カルノー図）（各セルの数字は入力変数の組み合わせ$x_1x_2x_3x_4$と表される二進数字を１の数の少ない順に，また同一の１の数を含むときは数値の小さい順に並べたときの順位番号を表し，各セルの右上付き数字で関数の真理値１(true)，真理値０(false)を表そう）で与えられたとした場合，ＮＡＮＤ回路の最も簡単な設計法は，Quine-McCluskey法で主項のみからなる関数Ｆ₁を実現する最小被覆を求め，これらの主項を入力ゲート，二段ゲートで合成する方法である．本例では$\bar{x}_1\bar{x}_2$, $\bar{x}_3\bar{x}_4$, $\bar{x}_1x_3x_4$, $x_1x_3\bar{x}_4$が主項で図１のように実現できる（ゲート数９，入力線数18）．但し，この方法ではゲート数および入力線数を最小にする方針で設計されていない．Ｐ－Ｎ項法はこの点を考慮したもので，大抵の場合この方法よりゲート数，入力線数の少ない回路が得られる．入力変数x_1, x_2, x_3, x_4およびその否定$\bar{x}_1, \bar{x}_2, \bar{x}_3, \bar{x}_4$をリテラルという．リテラルの論理積項，例えば$x_1\bar{x}_2x_4$を項

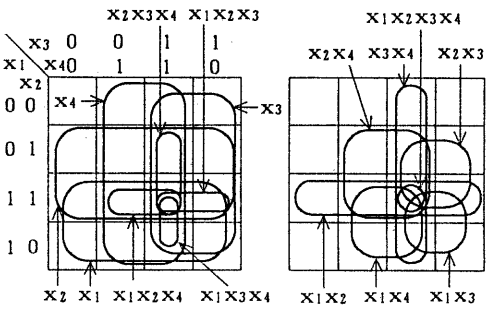

図３　４変数の許容項

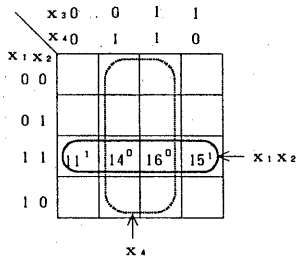

図4　許容項の打ち抜き

という．リテラルを全く含まない1も項である．そこで許容項とは，否定のリテラルを含まない項のことで，4変数の場合，図3で示されているように$x_1, x_2, x_3, x_4, x_1x_2, \cdots, x_1x_2x_3x_4$の15個とf＝1，つまり，マップ全体と一致する項の計16個（一般にn変数では2^n個）あり，これらをマップ上でみるとセル16（座標(1,1,1,1)，これをC_Mと表そう）を共通に含んでいる．これらの許容項だけがP（Positive許容項の略）項およびN（Negative）項になり得る．また，許容項は次のようにマップの各セルと対応づけて定義できる．例えば，図2のセル11の座標は$(x_1, x_2, x_3, x_4)=(1100)$である．このとき，座標が1のリテラルの積項$x_1x_2$をセル11の許容項と呼ぶ．セルは16個あり，1から16まで，それぞれ許容項が対応する．ここでの番号付けの結果，許容項はセル番号の小さい順により大きいか，等しい順になっている（許容項が含むセルの数の大小で比較して）．

上記の例の主項$x_1x_2\bar{x}_4$はマップ上，真理値1のセル11，15を表すが，これを図4のように，許容項x_1x_2を許容項x_4で打ち抜いたもの（他の文献では禁止といっている）と考えることができる．すなわち，x_1x_2の含むセルのうち14，16はx_4にも共通に含まれるので取り除かれ，1のセル11，15だけが残る．この操作をP項x_1x_2をN項x_4で打ち抜くと呼び，記号$P(x_1x_2)N_1(x_4)$で表そう．また，打ち抜きの結果，残った真理値1のセルを$P(x_1x_2)N_1(x_4)$で実現されたセルと呼ぶ．P－N項法はこの打ち抜きの概念に立脚したもので，P項1個につき1つの二段ゲートを要し，N項1つにつき1つの入力ゲートを要する．いまの例ではP項$P(x_1x_2)$に対して図1の4のゲートが対応し，N項に対しては図1の8の入力ゲートが対応し，x_1x_2が直接入力，x_4が入力ゲートを通してゲート4へ入力されている．ところで，代数的に$x_1x_2\bar{x}_4 = x_1x_2(\overline{x_1x_2x_4})$であるから，P項$x_1x_2$をN項$x_1x_2x_4$で打ち抜いても，打ち抜きの効果は変わらない．このような関係にあるN項は互いに互換性があるといおう．次節ではP－N項設計法の例として図2の論理関数F_1を実現してみる．

2.2　［例題］

（ステップ1）　図2の真理値1(true)のセルの最小番号は1である．そこで1のP項$P_1(1)$を取る．これで真理値1の1，2，3，4，5，6，11，12，15のセルが覆われる（図5(b)）．これですべての真理値1のセルが覆われたのでステップ1は終る．いま1つのP項が得られたが，$P_1(1)$は7，8，9，10，13，14，16なる不要な（すなわち，真理値0の）セルを含んでいるので（図2，図5参照）これらのセルを適当なN項を選んで打ち抜く必要がある．これが次のステップである．

（ステップ2）　$P_1(1)$に含まれる不要なセル7，8，9，10，13，14，16のうちで最小番号のセル7の許容項$N_{11}(x_2x_4)$を取る．これによって7，14，16が

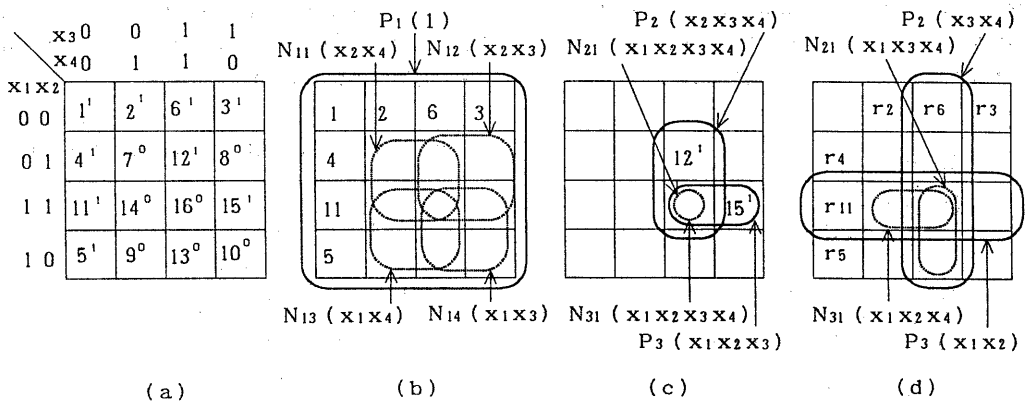

図5　P項から不要なセルをN項で打ち抜く例

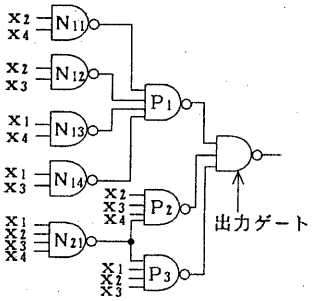

図6　基本P_0-N_0項によるF_1の回路

除かれるが8，9，10，13は残る．よって8のN項$N_{12}(x_2x_3)$を取る．これによっても除かれなかった不要なセル9，10，13についても同様に考えてN項$N_{13}(x_1x_4)$，$N_{14}(x_1x_3)$を取る．$P_1(1)$に含まれた不要なセルがすべて除かれたので次のP項へ移る．ところが，この例ではP項は$P_1(1)$のみであるのでここで終わる．すなわち，P項として$P_1(1)$，N項として$N_{11}(x_2x_4)$，$N_{12}(x_2x_3)$，$N_{13}(x_1x_4)$，$N_{14}(x_1x_3)$が得られる．しかしこの結果図5(b)のように，これらのN項によって$P_1(1)$の項から真理値1のセル12，15も取り除かれてしまう．そこでこれらのセルを実現するため再びP項を選ぶ（ステップ1）へもどる．このとき残された真理値12，15だけの図5(c)のマップで行うと，$P_2(x_2x_3x_4)N_{21}(x_1x_2x_3x_4)$，$P_3(x_1x_2x_3)N_{31}(x_1x_2x_3x_4)$なるP－N項が得られる．これを基本$P_0-N_0$項と呼ぼう．その回路を図6に示す（ゲート数9個，入力線数27）．しかし，セル12，15のP項を選ぶとき，すでに実現されている真理値1のセル2，3，4，5，6，11（但し，$P_1(1)$を導いたセル1は除外）は真理値を0とみても1とみてもよい．これらのセルを復活セル（don't care）と呼び，rとおくと図5(d)となる．この図で$r_6=r_{11}=1$とおいて12，15を実現するP－N項を求めると，$P_2(x_3x_4)N_{21}(x_1x_3x_4)$，$P_3(x_1x_2)N_{31}(x_1x_2x_4)$となる．図(c)で得た基本$P_0-N_0$項に比べ項が大きく入力線数ですぐれている．よって後者のP－N項を採用する．その回路を図7に示す．

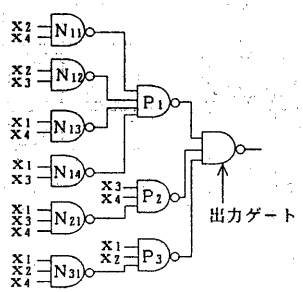

図7　復活セルによるF_1の回路

表1　P－N項最小被覆表

	P_1				P_2	P_3
	N_{11}	N_{12}	N_{13}	N_{14}	N_{21}	N_{31}
X_1					×	
X_2						
X_3						
X_4						×
X_1X_2						
X_1X_3				⊗	×	
X_1X_4			⊗		⊗	⊗
X_2X_3		⊗				
X_2X_4	⊗					×
X_3X_4						
$X_1X_2X_3$						
$X_1X_2X_4$						×
$X_2X_3X_4$						
$X_1X_3X_4$					×	
$X_1X_2X_3X_4$						

次の段階は表1のP－N項最小被覆表によって必要最小限のN項を選ぶことである．表は上欄にP項とそれを打ち抜くために必要なN項をすべて横に並べ，左側の欄に1以外の許容項を図のように大きい順で上から縦に並べる．例えば，$N_{21}(x_1x_2x_4)$は$P_2(x_1x_2)$なのでx_4，x_1x_4，x_2x_4と互換性を持ち，N_{21}列，x_4，x_1x_4，x_2x_4および$x_1x_2x_4$行に×印を書く．他の列についても同様である．できるだけ少ない許容項でN項をすべて被覆し，また，項数が同じなら，入力線数を小さくするため，できるだけ上位の項を優先してとる．いまの例では，必須項x_1x_3，x_1x_4，x_2x_3とx_2x_4の4つだけですべてのN項が被覆される（図7で○印をつけた行がこれらの4つの項で，6列あるN項が全部少なくとも1つの○印をもつ）．結局P－N項の組み合わせとして，$P_1(1)N_{11}(x_2x_4)N_{12}(x_2x_3)N_{13}(x_1x_4)N_{14}(x_1x_3)$，$P_2(x_3x_4)N_{31}(x_1x_4)$，$P_3(x_1x_2)N_{14}(x_1x_4)$が得られ，図8の論理回路が実現できる．

（例題終り）

この回路はゲート数8個，入力線数21で図6に比べ

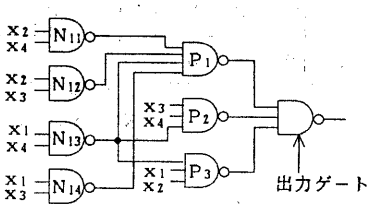

図8　簡単化されたF_1の回路

改良された．（この例では，はじめにQuine-McCluskey法を使って便宜的に設計したＮＡＮＤ回路より入力線数が多くなっているが，ゲート数が減少している）．

2.3　Ｐ－Ｎ項法のアルゴリズム

関数Ｆがマップで与えられているとする．そのとき，Ｐ－Ｎ項法の一般的設計アルゴリズムは次のとおりである．ｆ＝Ｆとおく．

［ラベル１］

（ステップ１）　関数ｆの真理値１のセルを全部覆うように，セル番号の小さい許容項から優先してP_1，P_2，…，P_dと生成する．このとき不要な（真理値０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，次の（ステップ２）へ行く（いま，そのようなＰ項の１つをP_iとする）．Ｐ項が全部，真理値０のセルを含まなければ，Ｐ－Ｎ項の選出の手順は終る．

（ステップ２）　P_iが不要な真理値０のセルを含めばこれらを打ち抜くため，セル番号の小さい許容項から優先してN_{i1}，N_{i2}，…，N_{ij}とＮ項をとっていき，真理値０のセルをすべて取り除く．

［ラベル２］

さて，ここで各Ｐ項に残ったセルを$r_1, r_2, \cdots, r_d$とし，$f_r = r_1+r_2+\cdots+r_d$としよう．f_rがｆの真理値１のセルをすべて含めば（$f = f_r$），Ｐ－Ｎ項選択の手順は終る．さもなくば（ステップ２）でＮ項により強制的に取り去られた真理値１のセル$t_1, t_2, \cdots, t_s$があるはずで，$f_1 = t_1+t_2+\cdots+t_s$としよう．

また，f_rで各Ｐ項の最小番号のセルを除いてできる関数をf_2とし，あらためて$f_2 = r_1+r_2+\cdots+r_d$とおこう（各Ｐ項が以下の繰り返し手順で再び生成されないためこうする必要がある）．Ｎ項により強制的に取り去られた$t_1+t_2+\cdots+t_s$を実現するため，$f = f_1+f_2$とおいて再びＰ－Ｎ項生成の手順を使うため［ラベル１］へいくが，このとき，はじめに$f_2 = 0$，すなわち，復活セルを全部真理値０のセルとおいて，Ｐ－Ｎ項を求める．これを基本Ｐ－Ｎ項といって，P_0-N_0項と表そう．

次にr_1を真理値１のセルとおき，他の$r_2, r_3, \cdots, r_d$を真理値０のセルとおいて，Ｐ－Ｎ項を求める．このときの項の数Ｇ，リテラルの総数ＩをＰ$_0$－Ｎ$_0$項の場合と比較して，あとのＰ－Ｎ項法の方がＧが小さいか，あるいはＧが等しくてもＩが小さければ，Ｐ→Ｐ$_0$，Ｎ→Ｎ$_0$と置き換える．

続いて，r_1を真理値０，r_2を真理値１のセルと置き換えるなどして，$r_1, r_2, \cdots, r_d$のセルに０と１の真理値のすべての組み合わせを考えて，Ｐ－Ｎ項を生成し，Ｐ$_0$－Ｎ$_0$と比較して最後に残ったＰ$_0$－Ｎ$_0$項を採用する．

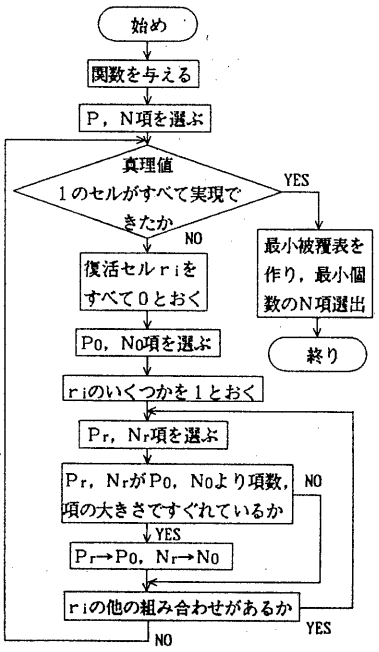

図９　Ｐ－Ｎ項法の流れ図

ここまで行って，まだ実現されていないセルがあれば，［ラベル２］へいって上記の手順を再び繰り返す．すべての真理値１のセルがＰ－Ｎ項のどれかに含まれたとき，手順は終わる．

次にＰ－Ｎ項最小被覆表を用いて，Ｎ項の最小被覆を求める．

以上のアルゴリズムを大まかな流れ図で描くと図９となる．

とくに，上のアルゴリズムで始めから終わりまで全く復活セルを考慮しないで得られる（基本Ｐ－Ｎ項による）ＮＡＮＤ回路を初期回路といおう．

３．多段ＮＡＮＤゲート回路から三段ＮＡＮＤゲート回路への変換

前章のＰ－Ｎ項法は，次のような欠点をもつので多変数の関数に適用できない．①関数が真理値表で与えられているとしているので2^n次元のブールベクトルが必要である．ｎは変数の数とする．②許容項の生成を早めようとすると，全部の許容項を行列の形でもつ必要があり，$2^n \times 2^n$のメモリがいる．③Ｐ－Ｎ項被覆表のためにも，大きなメモリを要し，④復活セルの数がｄ個あれば2^d個だけＰ－Ｎ項の生成の繰り返しが必

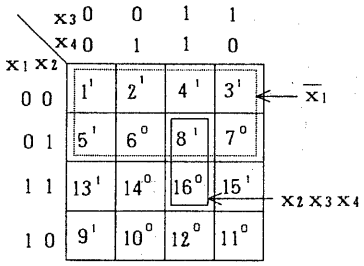

図10　４変数関数の例

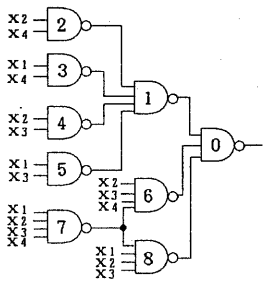

図12　多段回路から変換した三段回路

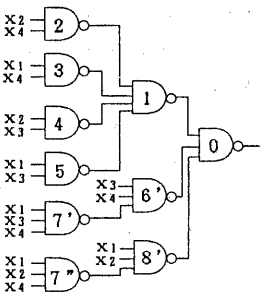

図13　Ｐ項，Ｎ項の拡大

要となる．

本章では前章のＰ－Ｎ項法の原理を継承しながら，これらの欠点を避けて多変数にまで適用できる手法を述べよう．

我々は本稿に先立って，多段ＮＡＮＤゲート回路の一設計法を発表した[6]．これをＭＡ法と略して呼ぼう．ＭＡ法では，関数は項の和の形で与えられる．初期回路は次のようにして得られる．

まず木の根（親）となる許容項をP_0として，その子となる許容項を

（ａ）関数ｆの真理値１のセルをできるだけ大きな許容項（一般に複数個：$\{P_i\}$）で被覆するよう生成する．

（ｂ）もし，これらの許容項の中に真理値０のセルを含むものがあれば（それをP_iとする），それらの許容項についてだけ，その許容項に含まれる０を被覆するよう，できるだけ大きな許容項を生成する（一般に複数個：$\{P_i'\}$）．

（$\{P_i'\}$ をP_iの子とする（逆にP_iは $\{P_i'\}$ の親）．但し，P_i,P_i'は一般に複数個）．

もし，これらの許容項のいずれかが（それをP_i'として），真理値１のセルを含めば，再び（ａ）に戻って，１のセルを被覆するよう許容項（ $\{P_i''\}$ ）を生成する．但しこのときは，P_i'に含まれる真理値１のセルでだけ真理値１となる関数をあらためてｆとして用いる．このように，すべての許容項が真理値１のセルと０のセルを混在しなくなるまで（ａ）（ｂ）の手順を繰り返す．この結果，許容項の木が形成され，親の許容項P_iは子の許容項P_i'を含む（P_i'に含まれるセルは全部P_iに含まれる）という性質がある．

許容項の木において，許容項をＮＡＮＤゲートで置き換え，対応するゲートの入力線リテラルとして各許容項のリテラルをそれに加えると，初期回路が得られる．特に許容項P_0は出力ゲートとおく．

なお，上記の許容項の生成法で，同一許容項が何度も表れることがあり，この場合，最初に表れたものを用いると，ゲート数が減る．

図10の関数[*1]（2.2の例と同じ関数）にＭＡ法を適用すると図11のような多段回路が得られる．三段ＮＡＮＤゲート回路では，出力ゲートはＯＲゲート，二段目ゲートはＰ項を形成するＡＮＤゲートとして働き，三段目ゲートは二段目ゲートでできるＰ項に含まれる，不要な真理値０のセルを取り除くために使われる．

一方で，ＭＡ法で得られた多段回路では，二段目（図11のゲート１）四段目（ゲート６，ゲート８）といった偶数段目のゲートは手順（ａ）でできた許容項に対応し，真理値１のセルを実現するために使われており，出力ゲートを除く奇数段目（ゲート２，３，４，５，７）のゲートは手順（ｂ）でできた許容項で，真理値０のセルを除くために使われている．

特に四段目のゲートの実現する真理値１のセルの集合（例えばゲート６）はその親の親である二段目のゲート（ゲート１）の実現する真理値１のセル集合に含まれており，四段目のゲートを出力ゲートに直接接続しても出力ゲートの関数値は不変である．このようにして更に四段目のゲート（元の回路で六段目）が生ずれば，それを出力ゲートに直接つなぐ…を繰り返して多段回路を三段回路に変更することができる．

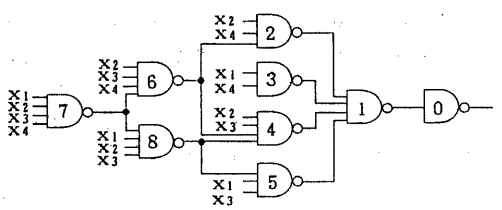

図11　ＭＡ法による多段回路

[*1] 図10のセル番号はＭＡ法に合わせているので，図２のセル番号と異なり，単に２進数(+1)を小さいものから順番に番号付けをしている．ＭＡ法では各セルとC_mのハミング距離を使って大きい許容項から生成するようにしている．

こうしてできた三段回路はＰ－Ｎ項法の初期回路と同じものとなる（図６と図12の回路は同じになっている）．

Ｐ－Ｎ項法ではＰ項，すなわち，二段目のゲートの数を減らすため，復活セルを利用して，初期回路より大きなＰ項に直した．ここでは，Ｐ項の数を減らすため，Ｐ項の拡大という操作を使うが，以下の図12の回路例で説明するとおりＮ項の拡大も伴う．

図12の二段目のゲート１，６，８に対応する許容項P_1, P_6, P_8を拡大できないか調べる．まず，P_1は拡大できない．次に，P_6について見る．許容項$P_6 = x_2x_3x_4$を含む許容項としては，$1, x_2, x_3, x_4, x_2x_3, x_2x_4, x_3x_4, x_2x_3x_4$がある．これらは許容項$P_6 = x_2x_3x_4$で陽に表れない変数$x_1$の否定$\bar{x}_1$がマップ上で占める領域内のセルの許容項と一致する（図10点線枠内）．既に生成しているＰ項や真理値０のセルの許容項を除けばセル{2, 3, 4, 5}の許容項がP_6の拡大の対象となる．このとき，

① $P_6N_7 = P(x_2x_3x_4)N(x_1x_2x_3x_4)$で実現された真理値１のセルが最低限保持されること．

② 入力している三段目のゲート，すなわちＮ項の数が増えないこと．という方針で行う．

セル２で許容項x_4を生成すると，中に真理値０のセルがあるので取り除くようx_3x_4，x_1x_4を生成するが，ここで実現されていたセル８が取り除かれてしまう．セル３，５のＰ項で考えても同様な理由でP_6の代わりにできない．しかし，セル４の許容項$P(x_3x_4)$から真理値０のセルを取り除くべく$N(x_1x_3x_4)$で打ち抜いてもセル８は残っており，上記の条件が満たされ，元の許容項$P_6 = x_2x_3x_4, N_7 = x_1x_2x_3x_4$がそれぞれ$P_6' = x_3x_4, N_7' = x_1x_3x_4$と拡大される．これで入力線数が削減され，場合によっては，この操作で拡大した項に含まれる小さな許容項が除去できることもある．同様にゲート８の許容項$P_8 = x_1x_2x_3$も，$P_8' = x_1x_2$に拡大でき，打ち抜くＮ項は$N_7 = x_1x_2x_3x_4$から$N_7'' = x_1x_2x_4$へ拡大する．

図13は図12を拡大した結果の回路である．Ｐ－Ｎ項法と同様に，ここでＮ項を減らすためＰ－Ｎ項被覆表を使う．

表２　必要な項のみ生成したＰ－Ｎ項最小被覆表

	1				6'	8'
	2	3	4	5	7'	7"
X_1					×	
X_4						×
X_1X_3				⊗	×	
X_1X_4		⊗			⊗	×
X_2X_3			⊗			
X_2X_4	⊗					⊗
$X_1X_2X_4$						×
$X_1X_3X_4$					×	

最上欄の番号は図13のゲート番号で，
一行目は二段ゲート（Ｐ項に相当），
二行目はそれらを打ち抜く入力ゲート（Ｎ項に相当）．

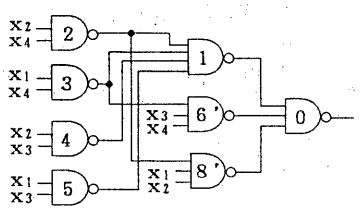

図14　簡単化した三段回路

Ｐ－Ｎ項法の場合にはＰ－Ｎ項最小被覆表を作るためにすべての許容項をあらかじめ生成した．ここでは，メモリを節減するため，Ｐ－Ｎ項を見れば互換性のあるＮ項がすべて導かれるので，必要に応じて互換性のある項を生成する方策をとっている．いまの場合は，表２のようなＰ－Ｎ項最小被覆表となる．

こうして，表２を使って新たに選ばれたＮ項を，先に拡大したＰ項に接続し直す．こうして簡単化された（ゲート数，入力線数が少なくなった）回路が図14である．図14では，ゲート数８，入力線数21とＰ－Ｎ項法で得られた結果と同じになった（回路構成が異なるのは，単にＰ－Ｎ項最小被覆表におけるＮ項の選び方の違いによるものである）．

４．結　果

本章では，ＭＡ３法をFORTRANによりプログラム化し実行した結果について述べる．後藤は，最近一線入力３段ＮＡＮＤゲート回路を得る方法[3]を発表している．このたび，このプログラムを計算時間などを比較するために我々が独自にFORTRANで作成したので４変数関数で比較する．後藤の論文にならって，既に最小回路の分かっている４変数関数100個で，後藤の方法とＭＡ３法で得られた回路とを比較してみると，入力線数が１～３本多くなったのが18個，そのうち，ゲート数も１個多くなったのが４個であった．他の82関数は，すべて一致した．それら100個の関数から任意に10個抽出して示したのが表３である．表中，最小回路とは明らかに分かっている最小回路のことであり，Ｇはゲート数，Ｉは入力線数，Ｓは段数，Ｔは計算時間（ミリ秒）をそれぞれ表す．また，関数は，４変数のセルを図10のように1, 2, …, 16と昇順に番号付けし，４個ごとのセルの真理値（０または１）を16進数に変換してある．表中，空白のところは最小回路と一致したことを示す．

ＭＡ３法は，多変数関数に適用することを考えているので，近似最小回路を目指しており，小変数で若干

表３　４変数関数10個の比較結果

関数	最小回路 G I S	後藤の方法 G I S T	ＭＡ３法 G I S T
0 0 B F	5 10 3	123	30
B 3 3 3	6 9 3	137	37
F B F 7	6 11 3	152	6 12 3 66
F D F 7	6 11 3	151	6 12 3 66
7 0 3 F	6 12 3	128	63
4 5 D D	6 13 3	126	69
7 3 3 B	6 13 3	130	6 14 3 65
0 8 1 D	6 14 3	144	54
B 5 5 7	7 16 5	9 19 3 134	9 19 3 64
B 7 7 F	8 19 5	9 18 3 132	9 18 3 81

Gはゲート数，Iは入力線数，Sは段数，Tは計算時間(ミリ秒)．

後藤の方法より悪くなるのはやむを得ない．

表４は，変数の数が４～10までのＭＡ３法で求めた回路結果である．変数の数４～９までの各数値は，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個の関数の平均値である．変数の数10だけは，真理値表濃度0.05から0.55まで実行可能であったが，0.65以上は初期回路を記憶する配列でメモリ不足が生じ結果が得られなかった（使用計算機はIBM 3081-KX4，富山大学情報処理センター）ため，一部の結果から求めた平均値である．

許容項は，一般にｎ変数で２^{n}個存在するが，ＭＡ３法では，２^{n}個のうち（表２のように）必要な許容項のみ生成することで，最小被覆表に要するメモリの節約を図ろうとしている．しかし，

縮小率＝生成した許容項数／全許容項数

と定義して，変数の数４～９までの様子を示すと表５となる．変数の数が７，８，９と大きくなるにつれて，ほとんどすべての許容項の生成が必要となってきて，意図した効果が上がっていないことが分かる．この場合，互換性のある項でも余り役に立たない不要な項を早く判定して，捨てるという方法を取り入れることも

表４　ランダム関数による結果

変数の数	ゲート数 G	入力線数 I	計算時間 T(ミリ秒)
4	5.78	13.54	25
5	10.12	30.76	75
6	16.42	61.60	303
7	29.54	137.60	1658
8	49.44	281.14	9779
9	86.36	572.80	63246
10	151.83	1247.41	388816

各数値は，真理値表濃度0.05から0.1刻みで0.95まで各５個ずつ計50個の関数の平均値．但し，10変数は0.05から0.55まで各２個ずつ計12個の関数の平均値．

表５　最小被覆表縮小率

変数の数	4	5	6	7	8	9
縮小率	0.55	0.64	0.80	0.89	0.93	0.96

変数の数４～８は30個の関数の平均値，９は15個の関数の平均値．

考えねばならない．

５．おわりに

本稿では，三段ＮＡＮＤゲート回路の一設計法として，４，５変数程度の関数なら手計算でもできるＰ－Ｎ項法について述べた．これを多変数向きにするため，多段ＮＡＮＤゲート回路から三段ＮＡＮＤゲート回路を得た後，二段目ゲートを拡大，三段目ゲートを表処理することで，ゲート数や入力線数を削減するＭＡ３法について述べた．

本方法で，さらに多変数関数に適用するため，項の拡大に論理関数簡単化に使われている項の拡大手法を使うこと，表処理を避け，回路の接続関係だけを使ってゲート数等を削減する手法を考えている．

参考文献

[１] Maley, G.A. and Earle, J.: The Logic Design of Transistor Digital Computers, John Wiley, 1963.

[２] Gimpel, J.F.: The Minimization of TANT Networks, IEEE Trans. Electron. Comput., Vol. EC-16, No. 1, pp. 18-38, 1967.

[３] 後藤：一線入力３段ＮＡＮＤゲート回路の行列法による最小化手法，情報処理学会論文誌，Vol. 32, No. 9, pp. 1156-1171, 1991.

[４] 松田：三段ＮＡＮＤゲート回路網の計算機による設計法，昭和51年度電気四学会北陸支部連合大会講演論文集，Ｂ-41，1976.

[５] 松田：三段ＮＡＮＤゲート回路網の論理設計法，昭和51年度電気四学会北陸支部連合大会講演論文集，Ｂ-42，1976.

[６] 松田，他：多段ＮＡＮＤゲート回路の一設計法，情報処理学会研究報告，ＤＡ67-6，June, 1993.