三段 NAND ゲート回路の論理設計法について（Ⅰ）
――P－N 項法――

松田　秀雄，宮腰　　隆，畠山　豊正

1．はじめに

ここで発表する，三段 NAND ゲート回路の論理設計法の一つ，P－N 項法は，すでに昭和51，52年度に電気四学会北陸支部連合大会で口頭発表している。当時は半導体技術もＩＣ（集積回路）といわれる時代で，１チップに NAND ゲートを３ないし４個集積したものが出まわっていた。従って，論理設計といっても規模も小さく，入力線数で，３ないし４，ゲート数で10個位のものを想定すればよかった。その後，我々の研究の主眼は NAND ゲート回路からより一般の AND，OR ゲート二段論理回路の方に移り LSI から VLSI へと集積化の規模が発展するのに応じてより大きい回路の論理設計，つまり，多変数論理関数の簡単化に興味を向けてきた。これは，LSI の設計に PLA がよく使われ，PLA は AND ゲート，および，OR ゲートを格子状に多数並べたものを接続して，ユーザが必要とする論理機能を AND，OR ゲート二段論理回路として実現して，使用するからである。ところが，最近，PLA にも NAND ゲートを格子状に多数並べたものが現れ，より規模の大きな NAND ゲート回路の論理設計法が必要となってきた。ひるがえって考えるに，P－N 項法は15年前に発表したものとはいえ，非常に先駆的な方法である。それに，AND，OR ゲート二段論理回路の設計法の研究で培かってきた大規模回路に対処するノウハウが，いまや十分にあり，それと結びつけると，より多変数関数を扱える手法が容易に生み出せる。そのようなわけで，P－N 項法を，ここで論文としてまとめておくことは意義あることと考える。P 許容項から不要なセルを N 許容項で打ち抜くという考え方に基づく P－N 項法について詳述し，計算機プログラムした結果も述べる。

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

NAND ゲート回路の設計については，これまでいくつか発表されているが，発見的方法であったり[1)]，余りにも複雑であったりしている[2)]。ここに提案する P－N 項設計法は，４ないし５変数程度までの論理関数を三段 NAND ゲート回路で実現するのに適した，比較的簡単な手計算向きあるいは計算機向き設計法である。

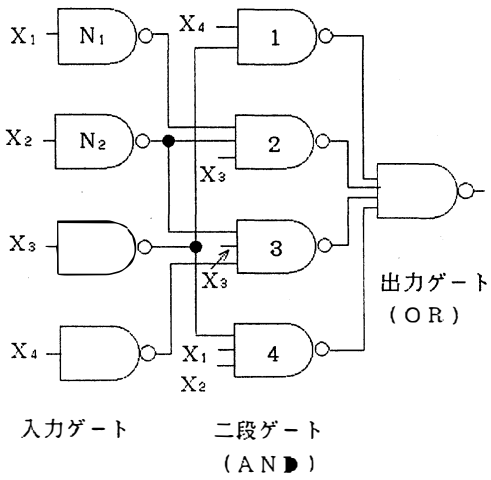


図1　三段 NAND ゲート回路

本論文を通じて，回路入力変数として否定変数を許さないものとする。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ゲートとして働らく。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，網かけのセルで関数が真理値1（true），網かけのないセルで真理値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[3]で主項のみからなる関数 F_1 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 $\bar{X}_3X_4$，$\bar{X}_1\bar{X}_2X_3$，$\bar{X}_2X_3\bar{X}_4$，$X_1X_2\bar{X}_3$ が主項で図1のように実現できる。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。

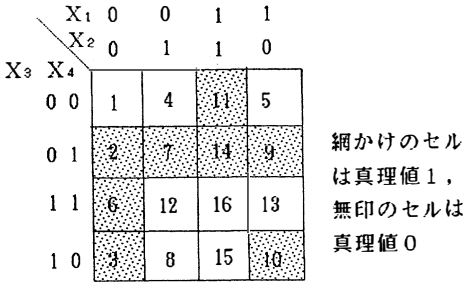
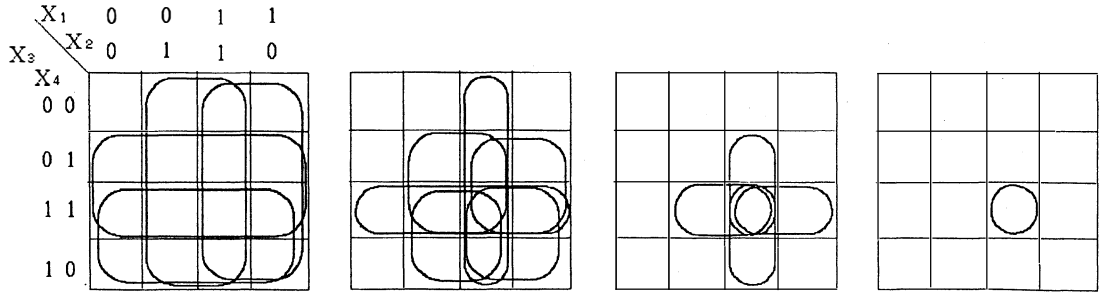


図2　マップのセル番号と
関数 F_1 の例

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，…，$X_1X_2X_3X_4$ 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_n 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6の座標は $X_1X_2X_3X_4=0011$ である。このとき，座標が1のリテラルの積項 X_3X_4 をセル6の許容項と



X1, X2, X3, X4　　X1X2, X1X3, X1X4 X2X3, X2X4, X3X4　　X2X3X4, X1X3X4 X1X2X4, X1X2X3　　X1X2X3X4

(a)　　(b)　　(c)　　(d)

図3　4変数の許容項

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。

上記の例の主項 $\bar{X}_1\bar{X}_2X_3$ はマップ上真理値1のセル3，6を表すが，これを図4のように，許容項 X_3 を許容項 X_1 および許容項 X_2 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_3 の含むセルのうち10，13，15，16は X_1 にも共通に含まれるので取り除かれ，8，12，15，16は X_2 にも含まれるので取り去られ，1のセル3，6だけが残る。この操作をP項 X_3 をN項 X_1 および X_2 で打ち抜くと呼び，記号 $P(X_3)N_1(X_1)N_2(X_2)$ で表そう。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 $P(X_3)$ に対して図1の2のゲートが対応し，N項に対しては図1の N_1，N_2 の入力ゲートが対応し，X_3 が直接入力，X_1，X_2 が入力ゲートを通してゲート2へ入力されている。代数的に $\bar{X}_1\bar{X}_2X_3=(\overline{X_1X_3})\bar{X}_2X_3$ であるから，P項 X_3 をN項 X_1X_3 および X_2 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。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。

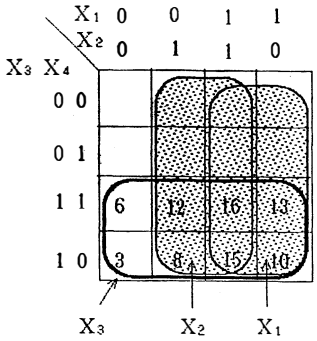


図4　許容項の打ち抜き

2.2 〔例題1〕

（ステップ1） 図2の網かけ（true）のセルで最小番号は2である。そこで2のP項 $P_1(X_4)$ を取る。これで真理値1の2，6，7，9，14のセルが覆われるが，3，10，11は残る（図5(a)）。この中で最小番号セルは3である。従って，3のP項 $P_2(X_3)$ を取る。これで新たに3，10が覆われるが，11は残る。よって11のP項 $P_3(X_1X_2)$ を取る。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3つのP項が得られたが，$P_1(X_4)$ は12，13，16，$P_2(X_3)$ は8，12，13，15，16，$P_3(X_1X_2)$ は15，16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）（Ⅰ） $P_1(X_4)$ に含まれる不要なセル12，13，16のうちで最小の番号セル12の許容項 $N_{11}(X_2X_3X_4)$ を取る。これによって12，16が除かれるが13は残る。よって13のN項 $N_{12}(X_1X_3X_4)$ を取る。$P_1(X_4)$ に含まれた不要なセルがすべて除かれたので次の許容項へ移る。（Ⅱ） $P_2(X_3)$ に

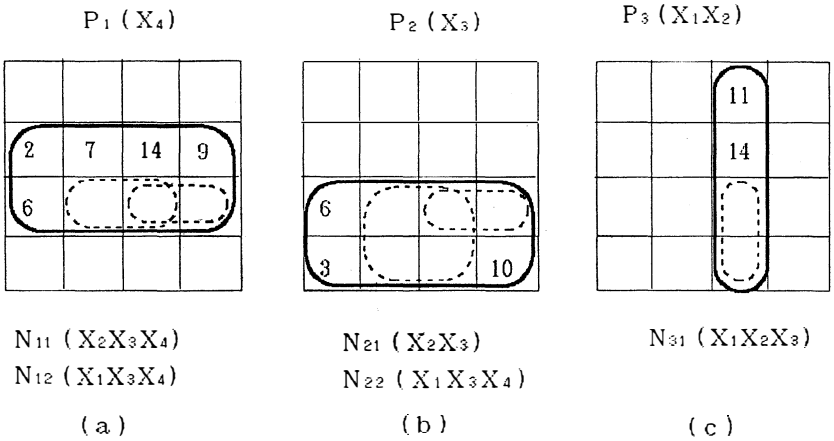


図5　P項から不要なセルをN項で打ち抜く例

含まれる不要なセル8，12，13，15，16についても同様に考えてN項 $N_{21}(X_2X_3)$，$N_{22}(X_1X_3X_4)$ を選ぶ。(Ⅲ) $P_3(X_1X_2)$ の不要なセルについても同様に考えてN項 $N_{31}(X_1X_2X_3)$ を取る。以上ですべてのP項からすべての不要なセルをN項によって取り除いた。しかも各P項に残っているセルは図2の関数 F_1 のすべての真理値1を含んでいるので関数が実現できたことになり，P－N項の選択は終る。

次の段階は図6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順序で縦に並べる。例えば，$N_{31}(X_1X_2X_3)$ は X_2X_3，X_1X_3，X_3 と互換性を持つので，N_{31} 列，X_2X_3，X_1X_3，X_3，および $X_1X_2X_3$ 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，X_2X_3 と $X_1X_3X_4$ の2つですべてのN項が被覆されるのでこれらの行の×印に○印をつける。結局P－N項の組み合わせとして，$P_1(X_4)N_{11}(X_2X_3)N_{12}(X_1X_3X_4)$，$P_2(X_3)N_{21}(X_2X_3)N_{22}(X_1X_3X_4)$，$P_3(X_1X_2)N_{31}(X_2X_3)$ が得られ，図7の論理回路が実現できる。

（例題終り）

この回路はゲート数6個，入力線数17で図1に比べ大幅に改良されたことがわかる。〔例題1〕は比較的簡単にできたが，一般には（ステップ2）で復活セルというものが現れ，N項の選択のさい，多少の判断を要する。次の〔例題2〕，図8でこれを示そう。

	P_1		P_2		P_3
	N_{11}	N_{12}	N_{21}	N_{22}	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					
$X_1X_2X_3$					×
$X_1X_2X_4$					
$X_2X_3X_4$	×				
$X_1X_3X_4$		⊗		⊗	
$X_1X_2X_3X_4$					

図6　P－N項最小被覆表

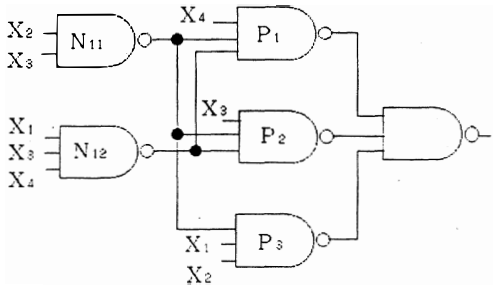


図7　簡単化された F_1 の回路

2.3 〔例題2〕

図8（a）が関数 F_2 の真理値表である。前例にしたがって，（ステップ1），（ステップ2）を進めると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。この結果，これらのN項によって P_1（1）の項から真理値1のセル13，14も取り除かれてしまう（図8（b））。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13，14だけの図8（c）のマップで行うと，$P_2(X_1X_3X_4)N_{21}(X_1X_2X_3X_4)$，$P_3(X_1X_2X_4)N_{31}(X_1X_2X_3X_4)$ なるP－N項が得られる。しかし，セル13，14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と呼び，r とおくと図8（d）となる。この図で $r_6=r_{11}=1$ とおいて13，14を実現するP－N項を求めると，$P_2(X_3X_4)N_{21}(X_2X_3$

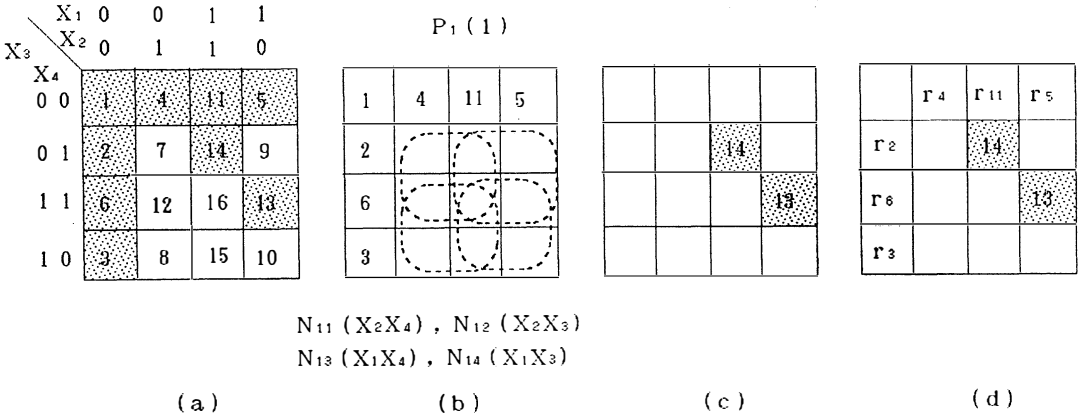


図8　復活セルがある関数（F_2）の例

X_4)，$P_3(X_1X_2)N_{31}(X_1X_2X_3)$ となる。図(c)で得たP－N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。このあとP－N項最小被覆表を作って最小個数のN項を選ぶと次のように，$P_1(1)N_{11}(X_2X_4)N_{12}(X_2X_3)N_{13}(X_1X_4)N_{14}(X_1X_3)$，$P_2(X_3X_4)N_{11}(X_2X_4)$，$P_3(X_1X_2)N_{14}(X_1X_3)$ となる。なお本例の回路図は省略する。　（例題終り）

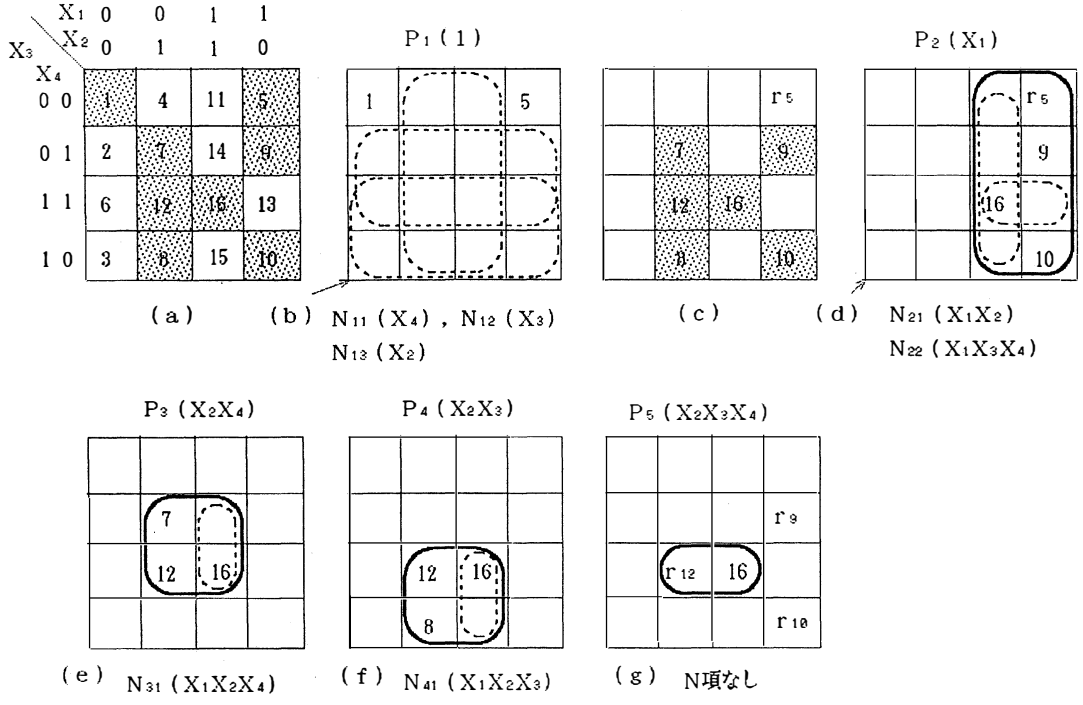


図9　P－N項法の例

2.4　P－N項法の計算機による設計法

本節では，P－N項法を用いて，関数の真理値表をデータとして計算機に与えたとき，最適な論理回路を出力として打ち出す自動設計プログラムの一つについて述べる。

図9（a）のような関数の真理値表が与えられたとする。そのとき，P－N項設計法のアルゴリズムをより一般的に述べると，次のとおりである。

（ステップ1）　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_m と生成する（例では図(b)の P_1(1)のみ）。このとき不要なセルを含めば，次の（ステップ2）へ行く。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 ($i=1,2,\cdots,m$) 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（図(b)では N_{11}，N_{12}，N_{13}）。

さて，ここで各P項に残ったセルが与えられた関数値1のセルをすべて含めば，許容項選択の手順は終る。さもなければ（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のセルがあるはずで，これを実現するため再度（ステップ1）へもどる。この際，復活セルがあっても，まず，これを無視して（ステップ1）（ステップ2）を実行し，基本P－N項を選ぶ。これを P_0-N_0 項と表そう。次に復活セル r_1，r_2，…，r_k を考慮して，0，1の種々の組み合わせについて，各々（ステップ1）（ステップ2）を繰り返し，P－N項を得る。これを P_r-N_r 項と表そう。基本 P_0-N_0 項と比較評価して良い方をとる。手計算ではマップ上で視覚判断する（例では図(c)で $r_5=1$ とおくと，図(d)，(e)，(f)のP－N項が選ばれ，$r_5=0$ とした基本P－N項よりP項の個数が1個減るので，こちらを採用する）。

ここまで行って，まだ実現されないセルがあれば，上記の手順を再び繰り返す。そしてすべての真理値1のセルがP－N項のどれかに含まれたとき，手順は終る（例ではもう一度行って，図(g)ですべてのセルが覆われる）。

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かくと図10となる。

計算機設計のプログラムではマップはセルの数（n 変数なら 2^n 個）に等しい次元のブールベクトルで表現する。図9(a)の真理値表なら，TRF＝(1000101111010001）となる。各セルの真理値0および1をセル番号の小さい順に左から右へ並べている。この方法によると，許容項 X_2X_4（図9(e)参照）は7，12，14，16のセルでのみ1となるので，左から7番目，12番目，14番目，16番目を1とした（0000001000010101）というベクトルで表現できるが，これはセル7

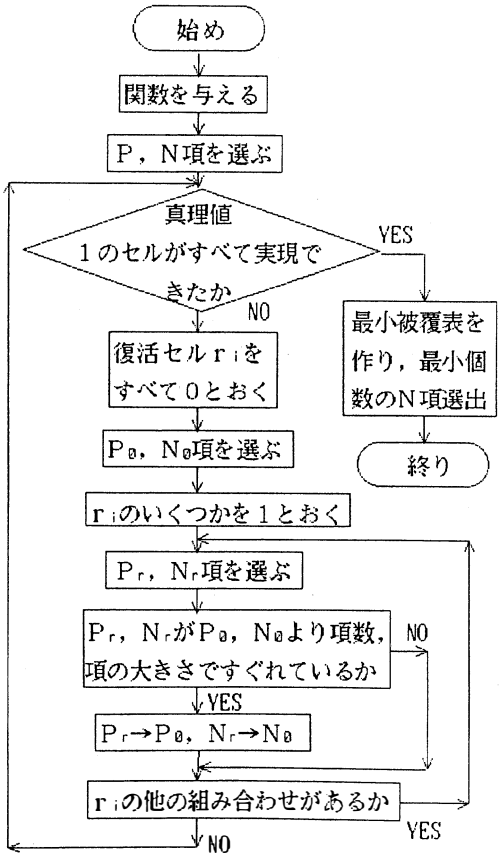


図10　P－N項法の流れ図

の許容項であるから，これを行列C(16, 16)の7行目におく。他の15個の許容項も同じ要領でベクトルで表し並べたのが図11で，許容項行列と呼ぶ。計算機設計では，これがP－N項の打ち抜きに本質的な役割をする。P項 $P_3(X_2X_4)$ は計算機ではP(3)＝7としておく。X_2X_4 についての情報が欲しければ行列Cと結びつけてC(P(3), K)＝C(7, K), (K＝1,2,…,16)によっていつでも得られる。N項も同様に表す。例えば，$N_{21}(X_1X_2)$ (図9(d))ならN(2, 1)＝11とする。P項の数は4変数の場合で高々8＝2^{4-1}(一般に2^{n-1})個であり，各P項1個につき必要とするN項の数は高々4＝2^{4-2}（一般に2^{n-2}）個である。従って，これらの項にはDIMENSION P(8), N(8, 4)なる配列を用意すれば十分である。

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	0	1	0	0	0	1	1	0	1	0	0	1	1	1	0	1
3	0	0	1	0	0	1	0	1	0	1	0	1	1	0	1	1
4	0	0	0	1	0	0	1	1	0	0	1	1	0	1	1	1
5	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1	1
6	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	1
7	0	0	0	0	0	0	1	0	0	0	0	1	0	1	0	1
8	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1
9	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	1
10	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1
11	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1
12	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
13	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

図11　許容項行列 C(16, 16)

図12は出力の一部を示す。(1)のTRFが与えられた関数（図9(a)参照），(2)のPは第一回目の許容項の生成で得られたP項でP(1)＝1，P(2)＝…＝P(8)＝0（0は許容項なしの意味），続くNから右が4つづつ＊印でN項が仕切られ，N(1, 1)＝2，N(1, 2)＝3，N(1, 3)＝4，N(1, 4)＝0，N(2, 1)＝0，N(2, 2)＝0…と並ぶ（図では途中で打ち切ってある)。(3)，(4)のTF3はこのP－N項によって実現されたセル，TF2は除かれた真理値1のセルを表す（ここまでは図9(b)に相当)。(5)はこのTF2の真理値1のセル（図9(c)参照）を実現するための第二回目のP－N項選択で得られた基本P項，N項，すなわちP_0－N_0項を示す。(6)のTF4はこのとき5に復活セルがあることを示している。TF2にさらにr_5＝1としたのが図中（7）のTFEである（図9(c)でr_5＝1としたもの)。このTFEに対してP－N項の選択が行われたのが（8）のPでこの部分をもう少し説明してみる。すでに，P(1)＝1が求まっているのでTFE(図9(c)参照)では，まず2番目のP項を求めることになるが真理値1の最小のセル5により，P(2)＝5とおく，次にP(2)を反転（否定ベクトル）させるため（1－C(P(2), K))＝(1111011100010000）を求め，これとTFEとをかけてTLとおくと7，8，12セルだけが1で他が0のベクトルとなる。これはTFEの真理値1のセル7，8，12がまだ覆われずに残っていることを意味する。最小のセル7により第3のP項としてP(3)＝7とおく。続いて（1－C(P(3), K)）とTLとをかけ，結果をまたTLとおく。今度はTLは8セルだけが1となっている。第4のP項はP(4)＝8ととる（このあたり，図9(d), (e), (f)に対応)。(8)のP5，7，8はこのように求めたものである。不要なセルを取り除くためのN項の選択も同様な考え方で行っている。(8)のP－N項と（5）の基本P_0－N_0項と比較評価され，これが有利と判定されて入れ替わる。(1)～(8)のパターンがCOUNT2以後もう一度繰り返されて（9）ですべてのP－N項が求まる。(10)はP（5）がN項をもたないことを表示している（項はキューブ，CUBEともいわれる)。

```
    INDEX  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
(1) TRF    1  0  0  0  1  0  1  1  1  1  0  1  0  0  0  1
    COUNT 1****************************************
    P  1  0  0  0  0  0  0  0   N  2  3  4  0* 0  0  0  0*
(2) P  1  0  0  0  0  0  0  0   N  2  3  4  0* 0  0  0  0*
(3) TF2    0  0  0  0  0  0  1  1  1  1  0  1  0  0  0  1
(4) TF3    1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
    PFSUM  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
(5)    7  8  9 10  0  0  0  0   N 14  0  0  0*15  0  0  0*13 14  0  0*13 15
    KO=    0    TFE   0  0  0  0  0  0  1  1  1  1  0  1  0  0  0  1
(6) TF4    0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
    KO=    1 (7)TFE   0  0  0  0  1  0  1  1  1  1  0  1  0  0  0  1
(8) P  5  7  8  0  0  0  0  0   N 11 13  0  0*14  0  0  0*15  0  0  0*
    COUNT 2** ************************************
    P  5  7  8  0  0  0  0  0   N 11 13  0  0*14  0  0  0*
    P     5  7  8  0  0  0  0   N  2  3  4  0*11 13  0  0*
       1
                 .
                 .   (この間省略)
                 .
    COUNT 3****************************************
    P 12  0  0  0  0  0  0  0   N  0  0  0  0* 0  0  0  0* 0  0  0  0* 0  0
(9) P  1  5  7  8 12  0  0  0   N  2  3  4  0*11 13  0  0*14  0  0  0*15  0

(10)PCUB(5) HAS NO NCUBE
```

図12　出力(１)

次に計算機は自動的に図13のＰ－Ｎ項最小被覆表を作り，最小被覆を求めてくれる（図13の(11) NEED１）。同図(12)は計算機が最適として選んだＰ－Ｎ項で，Ｐ項を二段目ゲート，Ｎ項を入力ゲートとして作図すれば，論理回路が得られる。

計算（CPU）時間は復活セルの多いもので数10秒，通常の関数では１秒未満であった。但し，FACOM 230-45Sによるデータである。現在の富山大学情報処理センターのIBM 3081-KX４を使えば，これらの計算時間の1/30ぐらいで求まると思われる。

３．ま　と　め

三段NANDゲート回路の設計法として，Ｐ－Ｎ項法について述べた。この方法では，n変数関数がマップで真理値として，与えられていると仮定している。マップには2^n個のセルがあり，これらに我々独自の方法で番号付けを行う。すると，許容項がセル番号の順序で大きなものから発生できるようになる。このため，Ｐ－許容項を生成して，その中の不要なセルをＮ－許容項を生成して取り除くという基本操作の繰

```
PCUBE          1  1  1  2  2  3  4
NCUBE          1  2  3  1  2  1  1
( 1) 1         0  0  0  0  0  0  0
( 2) X4        1  0  0  0  0  0  0
( 3) X3        0  1  0  0  0  0  0
( 4) X2        0  0  1  1  0  0  0
( 5) X1        0  0  0  0  0  1  1
( 6) X3X4      0  0  0  0  1  0  0
( 7) X2X4      0  0  0  0  0  0  0
( 8) X2X3      0  0  0  0  0  0  0
( 9) X1X4      0  0  0  0  0  1  0
(10) X1X3      0  0  0  0  0  0  1
(11) X1X2      0  0  0  1  0  1  1
(12) X2X3X4    0  0  0  0  0  0  0
(13) X1X3X4    0  0  0  0  1  0  0
(14) X1X2X4    0  0  0  0  0  1  0
(15) X1X2X3    0  0  0  0  0  0  1
(16) X1X2X3X4  0  0  0  0  0  0  0

(11) NEED1     2  3  4  4  6  5  5

                  1( X4)' ( X3)' ( X2)'
                 X1( X2)' ( X3X4)'
(12)          X2X4( X1)'
              X2X3( X1)'
            X2X3X4

各行最初はＰ項，( )' がＮ項
```

図13　出力(２)

り返しである本方法は計算機で容易に実行可能となる。

プログラム化にあたっては，関数を（サイズ 2^n の）ブールベクトルで表したこと，許容項を配列を使って生成する仕組み，アルゴリズムの進行にともなう各ステップでのデータの変化の模様を計算機出力を用いて，詳しく述べた。

本方法では 2^n の大きさのブールベクトルを用いるので，余り大きな回路には適用できない。

多変数向きの手法については別途報告する。

参考文献

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

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

3） E. J. McCluskey: Minimization of Boolean functions, Bell Syst. Tech. J., **35,** 5, pp. 1417-1444 (Nov. 1956).

昭和51年度，52年度電気四学会北陸支部連合大会で一部発表

The Logical Design of Minimal Three-level NAND Circuits (Ⅰ)
—— P-N cube method ——

Hideo MATSUDA, Takashi MIYAGOSHI, Toyomasa HATAKEYAMA

In this paper, we propose the P-N cube method for logical design of three-level NAND circuits. there are 2^n permissible cubes on the map for n-variable functions. We number all the permissible cubes in an unique manner that the permissible cube may be generated in order of its size. This strategy for generating the cubes make P-N cube method very efficient, because the following procedure is used repeatedly in the method; first some P-(permissible) cubes are selected to cover all 1'-minterms on the map, and then 0'-minterms get mixed with those 1'-minterms are deleted by using some N-(permissible) cubes from each P-cube. In the program, the function is represented as a 2^n-dimensional Boolean vector and permissible cubes are generated with a particular array of $2^n \times 2^n$ size. Also how to change the shape of data at each step of progress of the algorithm is explained by the output results of a computer.

〔英文和訳〕

三段 NAND ゲート回路の論理設計法について（Ⅰ）
—— P－N 項法 ——

松田　秀雄，宮腰　　隆，畠山　豊正

本論文で，我々は三段 NAND ゲート回路の論理設計法として，P－N項法を提案している。n 変数の関数のマップには 2^n 個の許容項がある。すべての許容項に許容項が大きさの順序で生成出来るよう我々独自の方法で番号付けを行なっている。項生成のこの方法によって，P－N項法は大変効率的となる。というのは，最初にマップ上のすべての１の最小項がカバーされるように，いくつかのP（許容）項が選ばれ，それから１の最小項と混ざって入った０の最小項がいくつかのN（許容）項を使って，各P項から取り除かれるという手順が本方法で繰り返されるからである。プログラムにおいて，関数は 2^n 次元ブールベクトルとして表され，許容項は大きさ $2^n \times 2^n$ のある特別な配列でもって生成される。また，アルゴリズムの進行の各スッテプで，データがどのように変わるかも計算結果を使って説明される。