論理関数の主項を計算機で高速に導出する手法

— 分割法について —

松田秀雄，　宮腰　隆

緒　　言

論理回路の自動設計では与えられた論理関数の主項(Prime Impicant)を高速に導出することがきわめて重要である。部分マップ法はそのための有効な手法であるが，ここでは，関数の真理値が与えられると，そのマップを分割して，より小さな変数のマップを作り，その関数の主項から，元の関数の主項を導出する，分割法を提案し，その有効性を明らかにする。すなわち，分割法は基本部分マップ法に比べ，計算時間を約40%程度までに短縮し，かつ，必要とする占有記憶量も相当量節約できるので，より多変数の関数の論理設計が可能となる。

1．理　　論

1.1　諸　定　義

1.1.1　部分マップ　　マップ上のセルは2進座標で表わされるが，これを1の数の少ない順に，同一数なら，2進数として小さい順に番号付けを行なう。図1は3変数のマップの例であるが，そのセル番号は図2のように2進ベクトルを並べた順序になっている。

セルjの座標で0をとる変数の否定リテラルの積項で表わされるマップ上の部分を部分マップjと呼びS_jと記す。3変数のマップとして，例えば図1の(b)を用いれば，各部分マップS_jは図3のように二重わく線で囲んだ部分で表わされる。すなわち，部分マップS_8はセル8の座標が（1,1,1 ）で否定リ

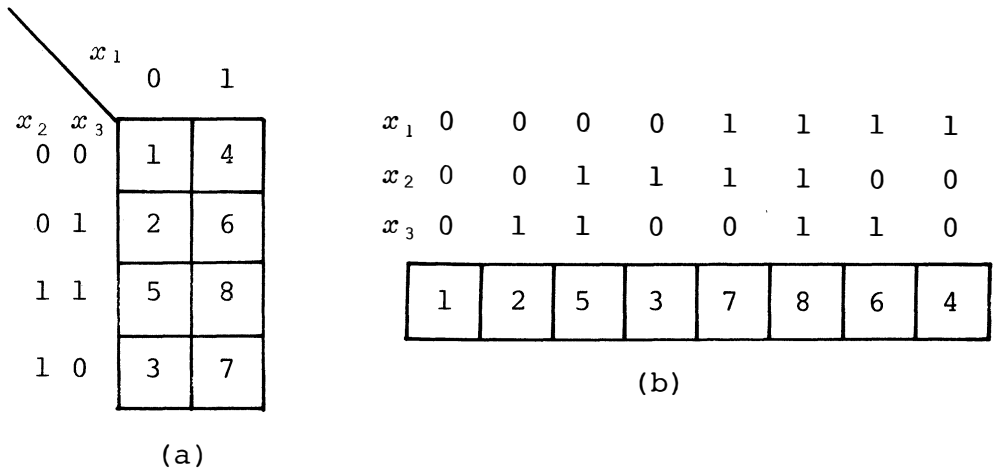

図1　3変数のマップの例

セル番号	2進ベクトル x_1	x_2	x_3
1	0	0	0
2	0	0	1
3	0	1	0
4	1	0	0
5	0	1	1
6	1	0	1
7	1	1	0
8	1	1	1

図2　ベクトルの並べ方

テラルがないので，このときだけ特別に，S_8＝マップ全体とする。S_7はセル7の座標が（1,1,0）であるから，$S_7=x'_5$，と表わされるマップ上の部分，又，部分マップS_6はセル6の座標が（1,0,1）であるから，$S_6=x'_2$，…，S_4はセル4の座標が（1,0,0）であるから，$S_4=x_2x_3$と表わされる部分，以下，同様に見ていくと，最後に部分マップS_1はセル1の座標が(0,0,0)であるから，$S_1=x_1x_2x_3$と表わされる部分，つまりセル1のみからなる。このように一般に部分マップはセル番号jが大きい程大きく（但し，セルの数で），いまの場合

$$S_8 \geq S_7 \geq S_6 \geq \cdots S_1$$

の関係がある。

1.1.2　許容キューブ　　部分マップjにおいて，セルiの座標で1をとる変数の肯定リテラルの積項で表わされるS_j上のキューブを許容キューブi,jと呼び$P_{i,j}$と記す。例えば上の部分マップS_8で考えてみると，セル1の許容キューブ$P_{1,8}$はセル1の

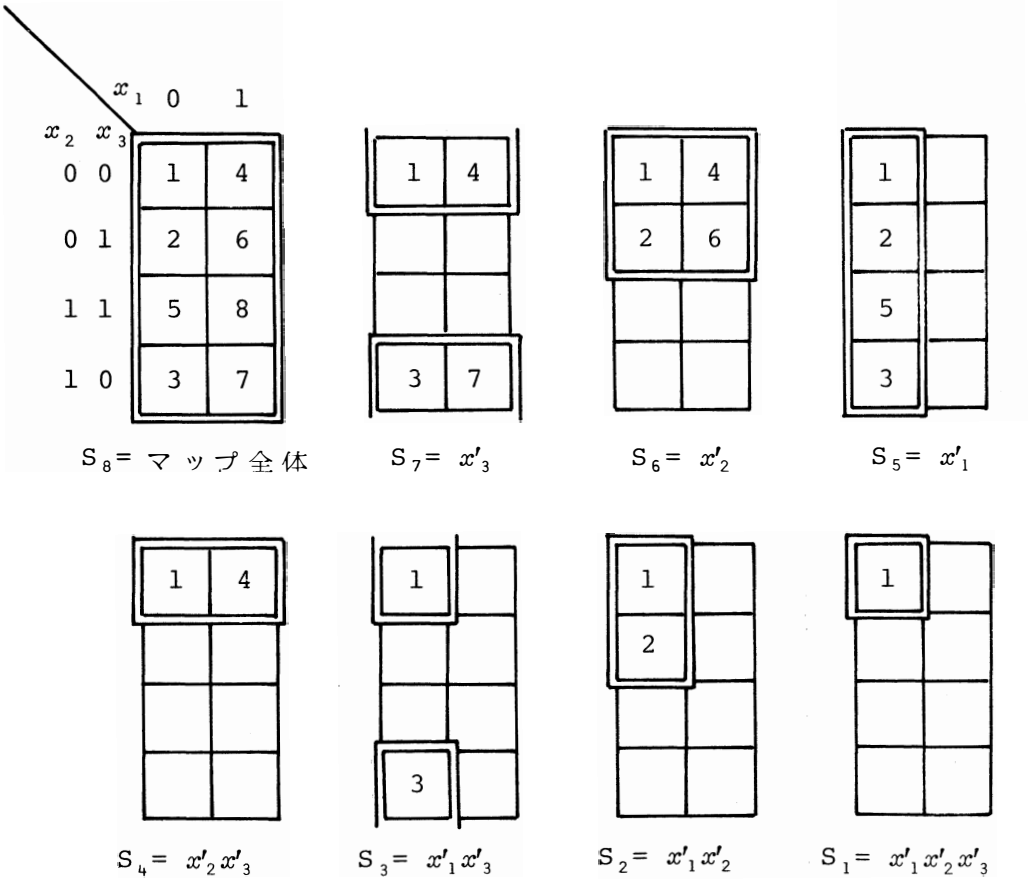

図3　3変数のマップの部分マップ（═わく）

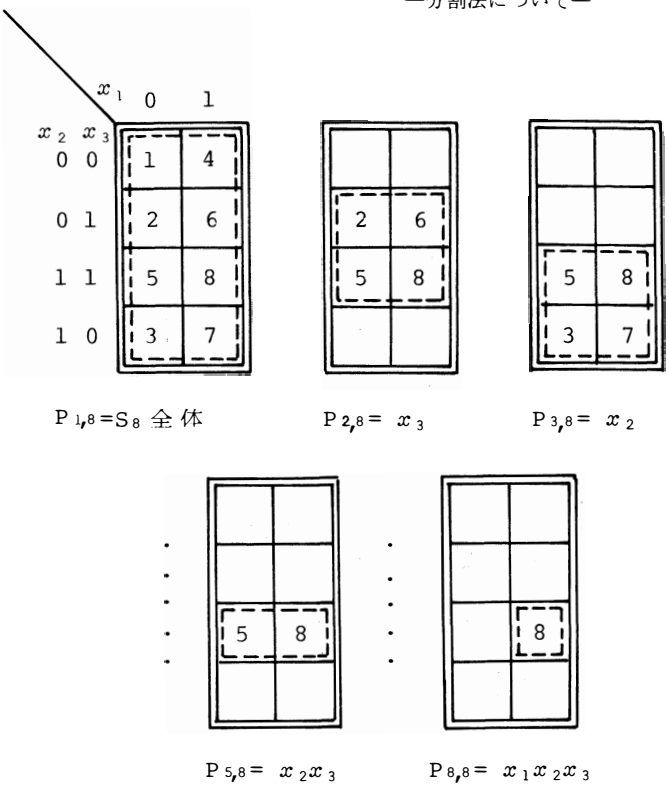

図4　S_8の許容キューブ（…わく）の例

座標が（0,0,0）で1をとる変数がないので，特別に$P_{1,8}=S_8$全体とするが，セル2は(0,0,1)なので，$P_{2,8}=x_3$,セル3は(0,1,0)なので，$P_{3,8}=x_2$,…，セル5は(0,1,1)で$P_{5,8}=x_2x_3$,…，そして，最後にセル8は(1,1,1)であるから，$P_{8,8}=x_1x_2x_3$となる。これらの許容キューブを図4に示す。

他の部分マップの許容キューブについても同様で，S_7の許容キューブ$P_{1,7}$は$S_7=\overline{x_3'}$全体，$P_{3,7}$は3の座標が（0,1,0）であるから，S_7上でキューブx_2と表わされる部分，すなわち，$P_{3,7}=x_2x_3'$,又S_4の許容キューブ$P_{1,4}$は　$S_4=x_2'x_3'$全体，$P_{4,4}$は4の座標が（1,0,0）であるから，S_4上でキューブx_1と表わされる部分，つまり$P_{4,4}=x_1x_2'x_3'$,更にS1の許容キューブ$P_{1,1}$はS_1全体，$P_{1,1}=x_1'x_2'x_3'$となる。これらの許容キューブの例を図5に示す。

一般に，部分マップjの許容キューブPi,jはセル番号iが小さい程大きく，これを部分マップ8の場合で例示すれば

$$P_{1,8}\geq P_{2,8}\geq P_{3,8}\geq\cdots\geq P_{8,8}$$

の関係がある。

なおn変数のマップでは部分マップの数はセルの数だけ，すなわち，2^n個あり，許容キューブの数は各部分マップのセルの数だけあり，これらを総計すると3^n個となる。$n=3$変数では部分マップの数は$2^3=8$個，許容キューブの数は$3^3=27$個あり，$n=2$変数では部分マップの数は$2^2=4$個，許容キューブの数は$3^2=9$個ある。図6に2変数マップのすべての部分マップ（═わく）及び許容キューブ（…わく）を示しておく。

1.2　基本部分マップ法[1]

以上で定義した部分マップS_j及び許容キューブ$P_{i,j}$を用いて，基本部分マップ法の手順を述べる。

(1)　論理関数Fをマップで与える。

(2)　部分マップS_jをセル番号の大きい順に発生する（但し，trueのセルのみ）。

(3)　各S_jごとに許容キューブ$P_{i,j}$をセル番号をiの小さい順に発生する（但し，trueのセルのみ）。

(4)　関数Fと$P_{i,j}$との論理積をとる。

$$F\cap P_{i,j}=P_{i,j} \tag{1}$$

式(1)が成り立てば，既知の主項と包含関係を調べ，含まれなければ$P_{i,j}$を主項として登録する。

(5)　(2)～(4)を繰り返す。

但し，(i)　大きな$P_{i,j}$が主項となれば，それに含まれるセルの許容キューブは発生不要である，

(ii)　ある部分マップで，すべての許容キューブと関数との論理積がとれれば，その部分マップに含まれるセルの部分マップは以後発生する必要がない，の諸性質があるので，(2)～(4)の繰り返し数は大幅に減少できる。

1.3　分　割　法[2]

n 変数（$x_1, x_2, \cdots, x_n$）のマップを n_1（$\leqq n$）変数の座標（0,0,…, 0），（0,0,…, 1），…，（1,1,…, 1）で分けると，n_2（$=n-n_1$）変数（$x_{n_1+1}, x_{n_1+2}, \cdots, x_n$）のマップが 2^{n_1} 個でき，それぞれ $M_1, M_2, \cdots, M_{2^{n_1}}$ と表わす。又関数 F もこれらの分割にしたがって，各マップ上で，それぞれ $F_1, F_2, \cdots, F_{2^{n_1}}$ となるものとする。例えば，$n=5$ 変数のマップを $n_1=3$ 変数（x_1, x_2, x_3）の座標で分けると，図7のように $n_2=2$ 変（x_4, x_5）のマップが $M_1, M_2, \cdots, M_8$ の $2^3=8$ 個できる。図8のように与えられた関数 F もこれらの分割に対応して，同図の如く，$F_1, F_2, \cdots, F_8$ の8個に分かれる。分割法の原理を大まかに表現すれば，以下の通りである。

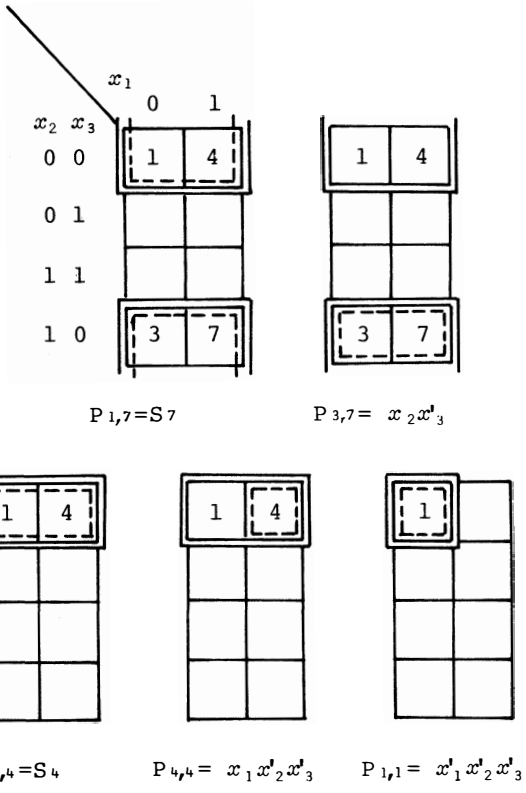

図5　許容キューブ（……わく）の他の例

(1)　n_2 変数のマップでセル番号 j の大きい順に部分マップ S_j を発生する。各部分マップ S_j ではセル番号 i の小さい順に許容キューブ $P_{i,j}$ を発生して，$P_{i,j}$ ごとに各 F_k（$k=1,2,\cdots, 2^{n_1}$）と論理積をとり

$$F_k \cap P_{i,j} = P_{i,j}$$

が成り立てば，k に対応する n_1 変数のマップ上のセルを1(true)，成り立たなければ0(false)として，新しい関数 $f_{i,j}$ を作る。

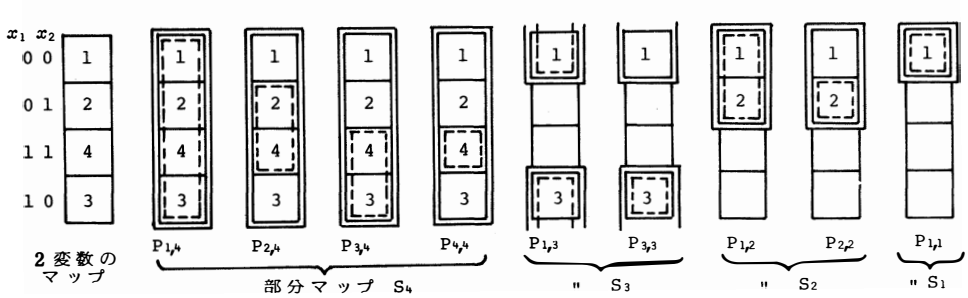

図6　2変数のマップの部分マップ（═══わく）及び許容キューブ（……わく）

(2) この$f_{i,j}$に基本部分マップ法を適用して主項を求める。

(3) この主項に$P_{i,j}$のリテラルを組合わせると，元の関数Fの内項となる。

(4) それまでに得られているすべての主項と包含関係を調べて，含まれなければ新しい主項として記憶しておく。すべてのキューブ$P_{i,j}$（3^{n_2}個ある）についてこれらの操作を繰り返せば，Fの主項が導出される。

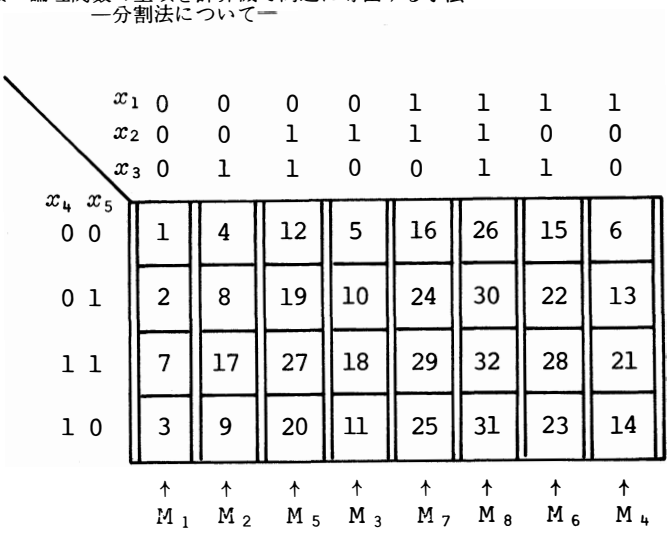

図7　マップの分割

次節では，例を上げて分割法について詳述する。

1.4　例　　題

図8の上図で与えた関数Fの例で考えてみる。n_2の変数のマップは同上左図のように（x_4,x_5）の2変数のマップとなる。ここで，分割法の(1)により，まず部分マップS_4を発生する。S_4のセル1の許容キューブ$P_{1,4}$は部分マップS_4全体であり（図6参照），これと各F_kとの論理積をとっている。つまり，$P_{1,4}$に含まれるすべてのセルで1になっている関数はF_8のみなので，F_8でだけ式(2)が成り立ち，n_1＝3変数（x_1,x_2,x_3）のマップのセル8を1，他はすべて0とする関数$f_{1,4}$ができる（図8）。手順(2)により，この$f_{1,4}$に基本法を適用するのだが，ここでは説明の便宜上図9の$f_{1,4}$のように書き直してみると，$P_{8,8}=x_1x_2x_3$唯1つが$f_{1,4}$の主項であることがわかる。（図9の$f_{1,4}$）。手順(3)により，これに$P_{1,4}$のリテラル（いまの場合，マップ全体なのでリテラルなし）をつけて，$x_1x_2x_3$がFの主項として得られる。これがはじめて得られた主項なので，手順(4)は不用である。続いて許容キューブ$P_{2,4}$と各F_kとの論理積をとる。図8の上図に示したのがこの場合で，同図下の$f_{2,4}$が得られる。これも便宜上，図9の$f_{2,4}$のように書き直し，図4の3変数の部分マップ，許容キューブを参照しながら基本法を適用してみると，部分マップS_8で$P_{8,8}=x_1x_2x_3$が，又部分マップS_2で$P_{1,2}=x_1'x_2'$が$f_{2,4}$の主項となることがわかる。$P_{2,4}$のリテラルはx_5であるから，$x_1x_2x_3$と$x_1'x_2'$にこれを合成して，$x_1x_2x_3x_5$と$x_1'x_2'x_5$がFの内項となる。手順(4)により，$x_1x_2x_3x_5$は先に得られている$x_1x_2x_3$に含まれるので除かれ，$x_1'x_2'x_5$だけが新たな主項として記憶される。

更にn_2変数（x_4,x_5）のマップの許容キューブ$P_{3,4}$, $P_{4,4}$と発生させて，F_kと論理積をとっていくと，図8の$f_{3,4}$,$f_{4,4}$が得られ，便宜上図9のように書き改めて，基本法を適用すると,$f_{3,4}$の主項として$P_{6,8}=x_1x_3$が，$f_{4,4}$の主項として$P_{6,8}=x_1x_3$, $P_{7,8}=x_1x_2$及び$P_{1,2}=x_1'x_2'$が得られ，それぞれ$P_{3,4}$及び$P_{4,4}$のリテラルを合成して，包含関係を調べると，$x_1x_3x_4$と$x_1x_2x_4x_5$がFの主項として残る。以下，同様の操作を更に残りのキューブ$P_{1,3}$から$P_{1,1}$まで（図6参照）繰り返すと，Fの主項がすべて求まる。

1.5　論理積演算についての考察

部分マップ法で基本となる演算は式(1)のキューブと関数の論理積演算である。実際のプログラムでは，これはキューブ$P_{i,j}$に含まれるセルについてのみ関数の真理値が調べられ，全部1なら，式(1)が成立，さもなくば不成立とみなすようになっている。したがって論理積演算に要する計算時間はキューブに含まれるセルの数に比例してくる。

n 変数の関数の場合，許容キューブ$P_{i,j}$ の総数は 3^n 個あり，各許容キューブに含まれるセルの総数は 4^n 個ある。

いま，上述したように n 変数の関数をn_1変数の座標で分割すると$n_2(=n-n_1)$ 変数のマップが 2^{n_1} 個できる。n_2変数の許容キューブ $P_{i,j}$ ごとに 2^{n_1} 個の関数と論理積をとって，１つの関数$f_{i,j}$を作るので，すべての$f_{i,j}$を作るに要するキューブに含まれるセルの総数は $4^{n_2}\cdot 2^{n_1}$ である。

又，１つの関数$f_{i,j}$ができると，n_1変数のマップを用いて基本法を適用するわけだが，n_1変数のキューブに含まれるセルの総数は 4^{n_1} 個である。$f_{i,j}$が全部で 3^{n_2} 個だけあるから，分割後の関数$f_{i,j}$に基本法を適用するために許容キューブに含まれるセルの総数は $4^{n_1}\times 3^{n_2}$個となる。

結局，分割法では

許容キューブに含まれるセルの総数＝ $4^{n_2}\cdot 2^{n_1}+4^{n_1}\cdot 3^{n_2}$　　(3)

となり，この数が論理積演算のためのセルの総数の上界を与える。表１に計算して示したように，n_1, n_2の値をnの½程度に選ぶと，基本法のセルの総数4^nに比べ相当小さくなることがわかる。

式(1)又は式(2)の論理積のためのbit演算の回数はキューブ $P_{i,j}$ に含まれるセルの数に比例するので，n変数の関数の主項を直接基本法で求めるより，n_1変数で分割して，しかるのち分割法を用いた方がよ

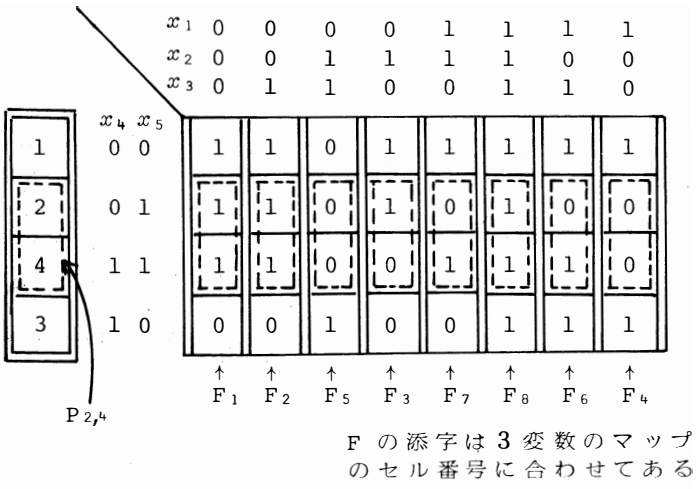

図８　分割法の説明図

表１　許容キューブに含まれるセルの総数

n	n_1	n_2	$4^{n_2}\cdot 2^{n_1}+4^{n_1}\cdot 3^{n_2}$	4^n
5	3	2	704　(0.688)	1024
6	3	3	2240　(0.547)	4096
7	4	3	7936　(0.484)	16384
8	4	4	24832　(0.379)	65536
9	5	4	91136　(0.348)	262144

（　）の数字は 4^n に対する比

り早く求まることがこのような理論的根拠から推察される。

実際には基本法のアルゴリズムのところで述べたように，trueのセルについてのみ部分マップと許容キューブを発生していくので，又性質(i)，(ii)があるので論理積数がこれらの数より大幅に減少することが予想されるが，それは基本法，分割法ともに同じ割合で減少すると考えられ，bit 演算数の割合はほぼ表1の( ) 中の比率に近いものと推論できよう。

3．計　算　結　果

われわれは基本部分マップ法及び分割法をFORTRAN言語でプログラム化し，FACOM 230−45S（富大計算センター）で実行してみた。

表2は基本法の計算結果で，$n=3$ 変数から8変数まで100個の関数の平均計算時間，9変数は50個の関数の平均計算時間（C.P.U.TIME）である。( ) 中の数字は2進ベクトルを発生するための処理時間で，何百個関数を計算しようと，はじめに唯1回必要とする，いわゆる前処理時間である。

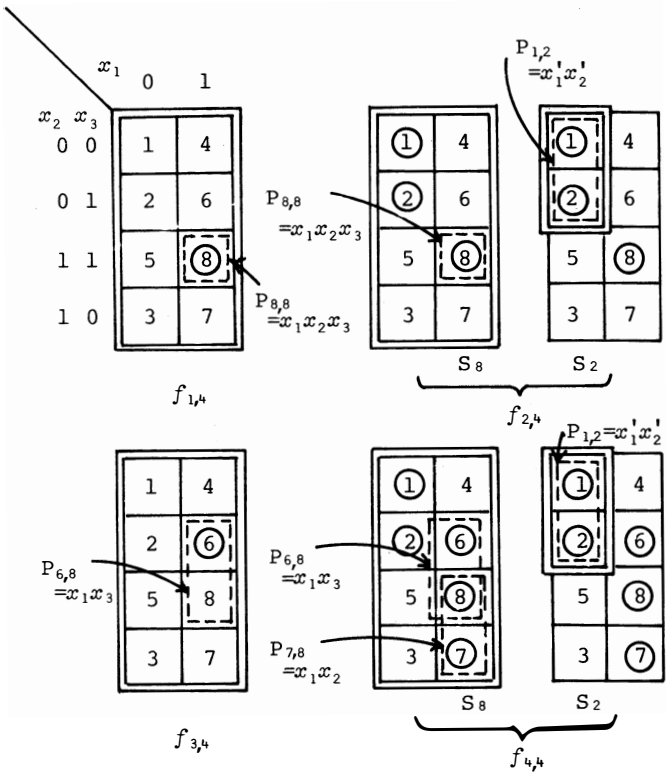

図9　$f_{i,j}$ に基本法を適用した例（○印のセルがtrue）

表3は分割法の計算時間で，$n=9$ 変数の関数について，n_2 を1から6まで変えたとき，計算時間がどのように変化するかをみたものである。これも50個の関数の平均計算時間である。n_2 が4ないし5，すなわち，n の約½のとき，最も分割法が効率的になることがわかる。

表2　基本部分マップ法の計算時間

変数 n	3	4	5	6	7	8	9
計算時間	4	16	67	290	1229	5662	26383
前処理	(3)	(6)	(14)	(26)	(55)	(118)	(250)

3 ～ 8 変数は100個の関数、9 変数は50個の関数の平均計算時間　　単位 msec

表4は分割法で $n=8$ 変数の関数について，全く同様の計算を行なったもので，このときもやはり n_2 が½ n のとき，最も効果的となっていることがわかる。なお，表3，表4とも（ ）中の数字は2進ベクトルの発生及びマップの分割のために要する，いわゆる前処理時間で基本法と大差がないことを示している。

表３　分割法の計算時間（９変数の関数50個の平均計算時間）

n_2	計算時間（前処理）	基本法に対する比
1	16829 (269)	0.638
2	12905 (238)	0.489
3	10702 (233)	0.406
4	9920 (248)	0.376
5	9958 (277)	0.377
6	11931 (331)	0.452

単位 msec

表４　分割法の計算時間（８変数の関数100個の平均計算時間）

n_2	計算時間（前処理）	基本法に対する比
1	3514 (133)	0.621
2	2663 (117)	0.470
3	2164 (117)	0.382
4	2049 (128)	0.362
5	2374 (150)	0.419
6	3428 (186)	0.605

単位　msec

結　　言

論理関数の主項を計算機で導出する場合，分割法が基本法に比べ，９変数の関数で38％，８変数の関数で36％まで計算時間が短縮できることを示した。このように多変数の論理関数では，与えられた関数をそのまま取り扱うより，約½程度の変数になるように元の関数を分割して，処理した方が非常に効果的である。

又，分割法では計算時間が短縮できるばかりではなく，メモリの節約もでき，より大きな論理関数の設計が可能となる。例えば図２に示した２進ベクトルをたくわえるのに要するメモリ量だけを取り上げてみても，基本法では$n \cdot 2^n$であるのに対し，分割法では$n_1 \cdot 2^{n_1} + n_2 \cdot 2^{n_2}$（$n = n_1 + n_2$）であり，これは９変数で$n_1 = 5$，$n_2 = 4$と選んでみると，基本法で4608となるのに対し，分割法ではわずかに224である。実際，基本法では９変数の関数までしか扱えない計算機で，われわれは分割法で１部の関数とはいえ，12変数の関数の主項を求めた例[(3)]がある。

更に，分割法の原理を再帰的につかえば，再帰的プログラムも組める。

参 考 文 献

１）宮腰，松田；信学技報，AL78, **17**,（1979）
２）松田，宮腰；電子通信学会情報・システム部門全国大会講演論文集〔分冊２〕，２－197,（1981）
３）大西；卒業論文　富山大学工学部（1982）

A Fast Algorithm for Generating All the Prime Implicants of Logical functions
—— Divide Method ——

Hideo MATSUDA, Takashi MIYAGOSHI

We already proposed a submap method which determines prime implicants of a logical function by the computer. The present report describes a divide (submap) method which is much more effectual than the submap method. A given logical function of n-variables is divided into 3^{n_2} logical functions of n_1-variables where $n=n_1+n_2$, and then the fundamental submap method is applied to each of those divided logical functions. Because the number of bit operations for logical product and the memory space size for the program to occupy decrease remarkably by using divide technique, we can reduce the computing time by 40 percent and treat logical functions of more variables in comparision with the fundamental method.

〔英 文 和 訳〕

論理関数の主項を計算機で高速に導出する手法

—— 分割法について ——

松 田 秀 雄，宮 腰　　隆

我々はすでに論理関数の主項を計算機で求めるための方法として部分マップ法を提案した。本報告では部分マップ法より一層効率的な分割（部分マップ）法について述べる。与えられたn変数の論理関数は2^{n_2}個のn_1変数の論理関数に分割され，それぞれに基本部分マップ法が適用される。（但し、$n=n_1+n_2$)。分割することにより，論理積のためのbit演算数及びプログラムのための占有記憶量が大幅に減少するので，基本法に比べ，計算時間が40%短縮でき，又，より多変数の論理関数が取扱えるようになる。

（1982年10月20日受理）