

単線入力 N A N D 回路と N O R 回路の 簡約形の同時論理設計

豊田 準三・鶴見 正司

* 岡山理科大学電子理学科

** 三菱電機コントロールソフトウェア株式会社

(昭和61年9月30日 受理)

まえがき

集積回路の高密度化に関連して単線入力による論理回路化が問題になるが、その議論は関数の積和形に偏っているようである。和積形をも含めての検討には大変な労力が必要になるからであろう。任意の関数の無駄のない姿での全ての関数形を把握して初めて真の最簡形がえられるのであり、同様のことは、どの変数も時間的な変化をすることのある非同期順序回路の安定化についても言える。このような場合には積和、和積両関数形について、冗長項はないが主項の全てを含み関数の論理特性を洩れることなく備えた完全主項形による論理設計が必要になる。これにも厄介な手順が伴い実施された例は見当たらない。カルノ図の機能を拡張した双対図によるとこの種の問題には対処し易い。ここでは単線入力での最少ゲート段数の3段で、入出力のレベル調整がされていて非同期安定性もある最小の論理回路のための関数の積和、和積両関数形を一挙に求める事を5変数と6変数の関数について検討した。また双対図形演算の効果的応用として関数合成の論理設計についても述べ、得られた結果について同様の検討をした。^{1) 2) 3)} 最簡同値関数群については従来の手法によるComputer Program "Simple Logic" をつくり相互検証して満足な結果がえられた。双対図の主要な事項については付録として添付してある。

1. 双対図による完全主項形

展開定理によれば、 n 個の論理変数 $x_1, x_2 \dots x_n$ の任意の n 変数の論理関数 $F(n)$ はその任意の変数のひとつ x_i に定数の 1 あるいは 0 を代入して x_i を取り除いて $n-1$ 変数になった関数と x_i との組合せで次の 2 形式のいずれかに展開できる。

(1, 1) 積和展開形 $F(n)=x_i F(x_i=1)+\bar{x}_i F(x_i=0)$

(1, 2) 和積展開形 $F(n)=(x_i+F(x_i=0))(\bar{x}_i+F(x_i=1))$

変数 x_i 又は \bar{x}_i の領域毎に積和展開最簡形を求めて、それらを積和形に纏めると完全主項

和がえられ、和積最簡形を和積形に纏めると完全主項積がえられる。ここでは変数は A, B, C, D および E であるとして、下記の関数 I が最小項展開形または最大項展開形で与えられた場合に双対図により完全主項形をも含めて両関数形を一挙に求める例をしめす。

(付録 1, 付録 2, 付図 2, および図 1 参照)

(1, 3) 最小項展開形

$$I(m) = \sum (3, 4, 5, 7, 10, 11, 15, 17, 19, 20, 22, 23, 24, 25, 28, 29)$$

(1, 4) 最大項展開形

$$I(M) = \prod (31, 30, 29, 25, 23, 22, 19, 18, 17, 15, 13, 10, 5, 4, 1, 0)$$

この関数の為の定数の配分に基づいた展開構造と双対図表示については付録 1 及び付図 1, 1~1, 5 を参照されたい。それらで示したように単位セルの論理特性と番地設定、即ち

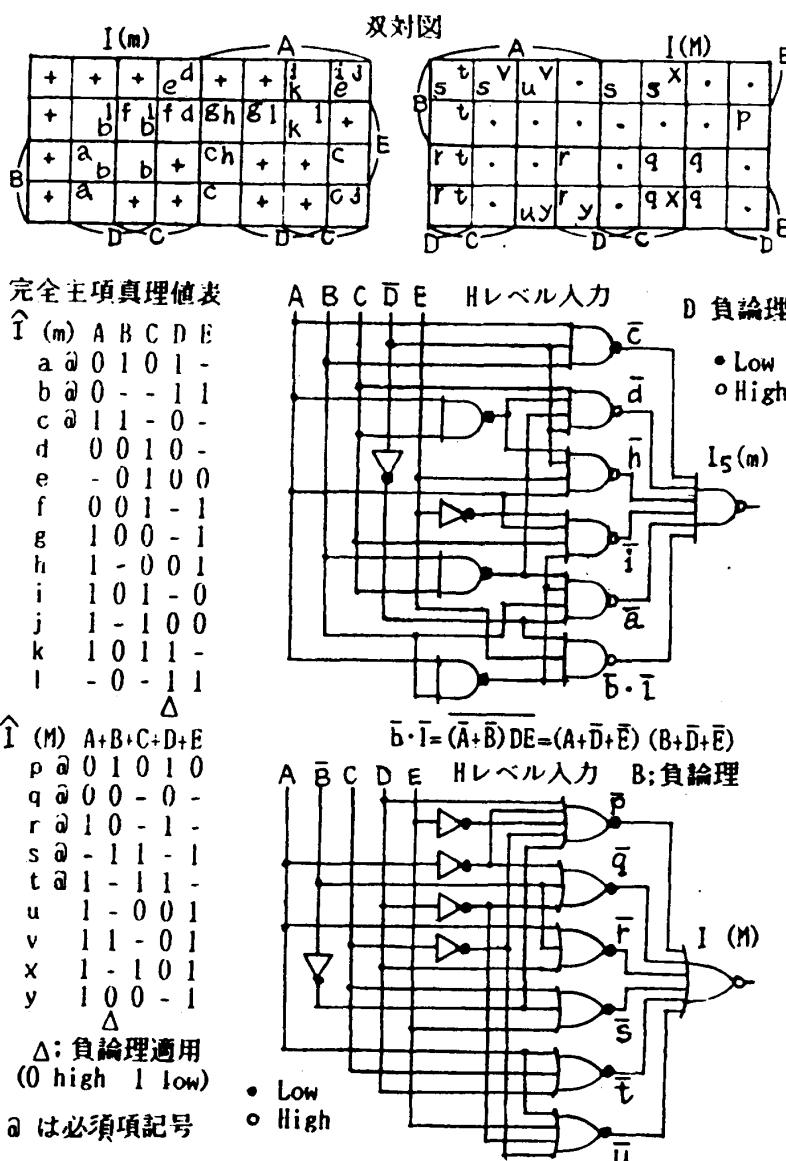


図 1 関数 I の双対図、完全主項真理値表及び論理回路

変数領域設定をすると互いに補完的になった一対の図形表示がえられる。・記号による積和形の空白部は和積形の記入領域になり、+記号による和積形表示の空白部は積和形の記入領域になる。それぞれ限定された部分への記入だからそれだけ容易である。記入には同値関数形検討の為にも展開真理値表作成のためにも英小文字を用いるのが便利で、いずれにしても局限された領域への記入あるから、記入しようとする変数領域を設定して記入すると図1に示した一対の図形表示は容易にえられる。（積和形にはa, b, • n を、和積形にはp, q, • z, を必要に応じて適当な記号を追加して起用する。）両関数形が一図形に表示できることは付図1,5にも示してあるが記入にも読み取りにも一方が記号であることが望ましい。えられた関数形を論理式で示すと次のようになる。

$$(1, 5) I(M) = pqrstu = (\bar{A} + B + \bar{C} + D + \bar{E}) (\bar{A} + \bar{B} + \bar{D}) (A + \bar{B} + D) \\ (B + C + E) (A + C + D) (A + \bar{C} + \bar{D} + E)$$

$$(1, 6) I(m) = a + b + c + d + g + i + 1 = \bar{A} B \bar{C} D + \bar{A} D E + A B \bar{D} + \bar{A} \bar{B} C \bar{D} \\ + A \bar{B} \bar{C} E + A \bar{B} C \bar{E} + \bar{B} D E$$

これらは同値関数のひとつを示したものである。单一記入になったセルが、ひとつでもある展開項は必須項であり、これが多い程記入は容易である。この程度の展開項数の場合は同値関数の全てを求めるとはそれほど困難ではない。しかしこれで完全主項形がえられているかについての保証はない。そのためには(1,1)または(1,2)に基づいた代入によらねばならない。しかしこの方法は展開項数が多い場合は大変な作業になる。双対図の演算性を起用するとこれは変数領域にある最小項又は最大項と、展開に用いる変数との論理演算として求めることができる。その内容は付録1と2を参照されたい。図1でえられた展開項全てを論理和又は論理積に纏めると付録2に示されている結果と同じになる。また後述のComputer Program "Simple Logic" の印字出力図6に示されているものとも一致しており充分な相互検証になっている。このように双方の完全主項形が一挙に得られるので関数形検討の具体的範囲は身近かなものになり、各種の条件に対処できることになる。その検討には関数の展開真理値表が用いられるが、"Simple Logic" では展開真理値表は各論理式の印字出力に引き続いて図6のようにえられるようになっている。

1, 1 最簡形の検討

所要素子数の少ない事を最小回路の条件に考えた2線入力の場合には、展開項数の少ない和積形I(M)の方を取り上げて回路化すればよい。これには必須項5、主項4で出来た完全主項和があるが、必須項に1項だけの追加で最簡形が作れのはu項だけ(図1参照)で、これが優先主項となって必須項と共に6項の唯一最簡形ができる。この関数をAND又はORの素子を構成単位として回路化すると7個の素子で回路化できる。一般に展

開項数 m の場合、変数入力の正論理について NAND 又は NOR の機能を示す素子を用いると上記の AND - OR 又は OR - AND の回路は入力の変数関係はそのままで同数の $m + 1$ 個の素子で NAND - NAND 又は NOR - NOR の回路に変更できる。

单線入力の場合は事情が違ってくる。NAND - NAND または NOR - NOR の 2 線入力回路の初段に現れる否定変数の全てを否定素子を用いて作り出せば、 n 変数の場合はどのような関数でも $n + m + 1$ 個あれば 1 線入力の回路化が出来る。⁴⁾ しかし関数形によつては上記に達しない数で 1 線回路化出来る場合がある。それは展開項の兼用又は前置段素子の兼用ができる関数形で、それを取り出すには同値関数の全てを把握する必要があり、それを求めるために完全主項形が必要になる。関数 I (m) には必須項 3, 主項 9 の項数の完全主項（図 1 参照）があり同値関数形の多いものである。それらは各項を双対図の対応した空白部に記入してもとめられ、それらの展開真理値表を作る事もできる。これを“Simple Logic”的印字出力で示すと図 6 のように 7 展開項の 6 種の同値関数がえられることがわかる。（脚注） そのうち $I_1(m)$ $I_3(m)$ および $I_5(m)$ が展開項兼用型の関数型で、 $I_5(m)$ の論理回路と兼用についての論理式は図 1 に示してある。前置段については部分的に兼用できても全体として n に満たない個数で回路化できる関数形ではない。1 線入力回路としては積和形、和積形とも同数の 12 素子で回路化出来ることになる。この関数にかぎらない他の回路との共用については積和形の方が選択の幅が広いといえる。

1, 2 入出力レベル関係の検討

高位レベル（H レベル）入力の 3 段ゲート回路の出力側で関数形をそのまま維持して高位のものを得ようすると、終段が NAND の場合にはその入力には最低 1 変数入力は低位（L レベル）のものが必要であり、NOR の場合には 1 入力でも H レベルのものがあつてはならない。関数 I は 2 線入力の最簡形の前置段に否定素子を用いてそのまま単純に 1 線入力化したのでは L レベルの出力がえられる関数形である。（積和形には正関数項がなく、和積形には負関数項がある。）この場合に関数形を維持したまま H レベル出力とすることは、展開真理値表を検討して負論理が適用できる変数を見出すことにより目的が達せられる。 $I_5(m)$ については変数 D に負論理を適用し、 I (M) については変数 B に負論理を適用して回路化した結果は共に図 1 に示してある。各素子の出力レベルは。で H レベル、。で L レベルを示してある。負論理を適用しない場合との比較は容易であろう。特定の関数に限定された範囲で、ある変数に負論理を適用することで差し支えが生じることはない。

脚注：展開項名の与え方は双対図と Program とでは一致させられるものではない。まぎれを避けるために後者の項名に揃えて示した。（双対図読み取りについては付録 3 参照）

浮動小数点数の仮数部の正負符号には正論理、指数部の正負符号には負論理を適用して指数比較の論理回路の論理設計が行われるのはその例である。

1, 3 非同期安定性の検討

非同期順序回路としての論理設計には変数の時間的変化には特別の配慮が必要になる。ブール代数では、例えば変数Aが \bar{A} に変わるために要する時間 Δt は0という理想状態での取扱である。現実には $\Delta t = 0$ と言うことはありえないで、 $\Delta t = \tau$ と言う有限値に対してはAでも \bar{A} でもない状態A'を考慮にいれる必要がある。このような過渡的状態から生じる不安定は同期順序回路ではクロック信号の導入で対処できるが、それがない非同期順序回路では関数形の問題として取り扱う必要がある。この場合にどの変数も時間的に変化するとしても1時点ではただひとつの変数だけが変化すると言う前提があれば、例えばA'と言う不安定状態の存在を考えても、それが避けられる関数形の選択により安定回路を求めることができる。上記のような不安定な関数形は関数の論理式で \bar{A} をA'に取替えてみて、他の変数に0または1を与えても $A + A' \neq 1$ （積和形の場合）；又は $A \cdot A' \neq 0$ （和積形の場合）；が残ることで見出すことができる。どの変数についてもこれが言えることであるから、言いえると、どの変数についても欠けた項があれば、それにより欠けていない項の不安定は吸収されることになる。この観点から図6で関数Iを検討すると項e、又は1のない $I_2(m)$ と $I_4(m)$ では変数A不在の項がない。 $I_2(m)$ については、 $\bar{B}C(\bar{D} + \bar{E}) = d + i = 1$ の時に $A + A' = 1$ が残り、Aの時間的変化について不安定である。 $I_4(m)$ についても同様である。^(脚注)これらを除く積和形とI(M)はいずれも5変数のそれぞれについて欠けている項で出来た関数形であり、1時点ではただひとつの変数しか変化しないというかぎりは安定である。図1に示した両回路は入出力レベル調整、非同期安定性の双方を考慮した3段ゲートの单線入力の最小論理回路と言うことができる。

2. 双対図による論理関数合成

論理式演算によると論理関数の合成には大変な手順が必要になる。双対図の演算性を活用するとこれが極めて効果的にできることを次の関数Kと前記の関数Iとの排他和関数P、および両者の対等論理和関数Qを求めることを引用して下記にしめす。

$$(2, 1) K(m) = \sum (0, 1, 2, 3, 5, 6, 8, 9, 10, 11, 19, 20, 21, 22, 25, 26, 29, 30, 31)$$

$$(2, 2) K(M) = \prod (27, 24, 19, 18, 17, 16, 15, 14, 13, 8, 7, 4, 3)$$

脚注：この場合は $\bar{B}C = d + k = 1$ の時に $A + A' \neq 1$ が残る。

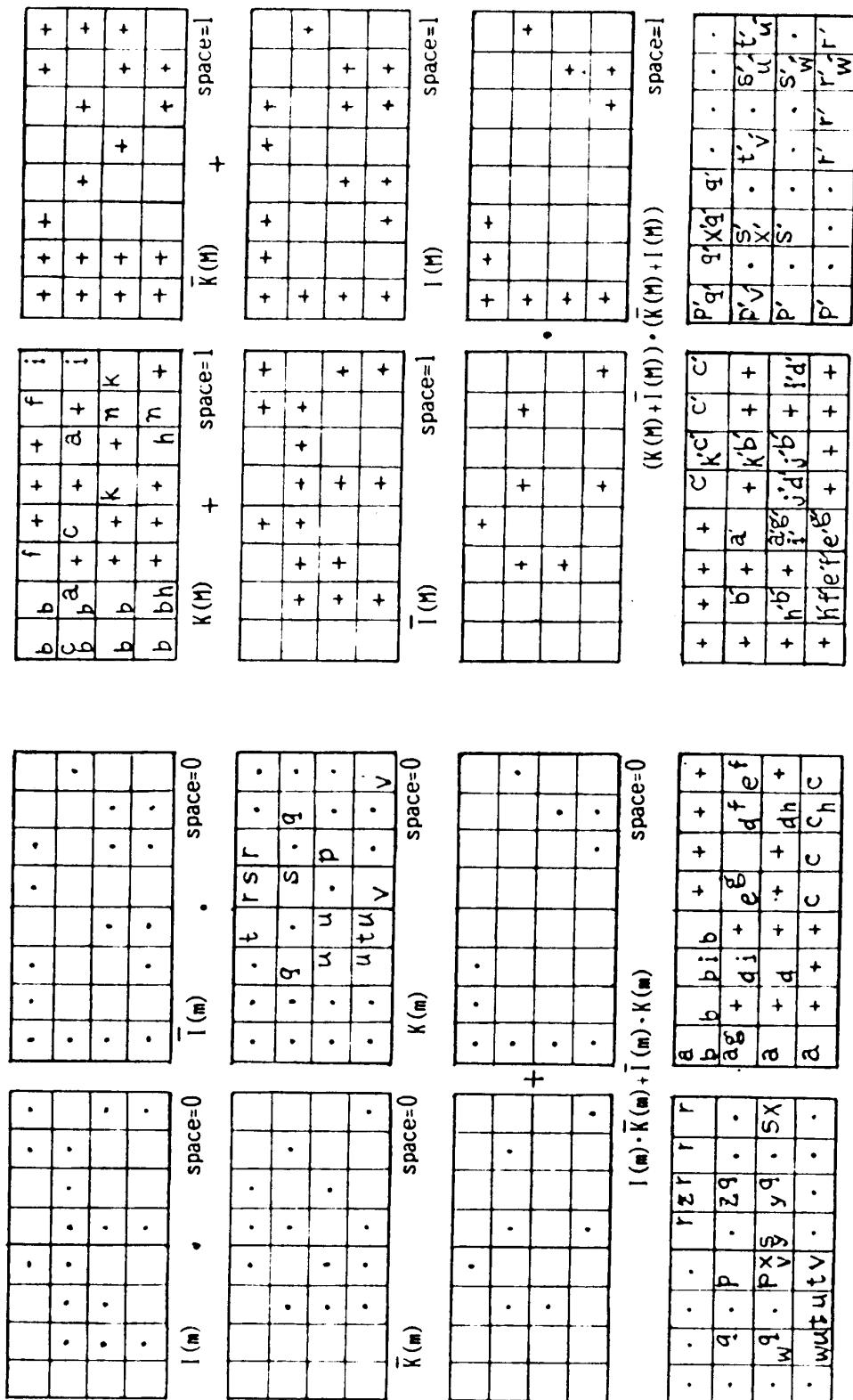


図2.1 関数Pの双対図論理合成と関数形の変換
 $P = I \oplus K$ (最小項図)

図2.2 関数Qの双対図論理合成と関数形の変換
 $Q = I \odot K$ (最大項図)

表 I.1 関数 P と Q の完全主項形

$\hat{P}(M)$	A+B+C+D+E	$\hat{P}(m)$	A B C D E	$\hat{Q}(M)$	A+B+C+D+E	$\hat{Q}(m)$	A B C D E
p a 1 - 0 1 0	a' a 0 - 0 0 -	$p'(m)$	1 - 1 1 -	$q(m)$	1 - 1 1 -	$q'(m)$	0 - 1 0 1
q a - - 1 0 0	b a 0 0 - - 0	$p'(a)$	1 1 - - 1	$q'(a)$	1 1 - - 1	$c(a)$	- 0 1 1
r a - - 1 - 1	c a 1 1 - - 0	$r'(a)$	0 0 - - 1	$c'(a)$	1 0 - - 0	$d'(a)$	1 0 - - 0
s a - 0 0 - 1 0	d a - 1 1 1 - 0	$s'(a)$	- 0 0 0 0	$d'(r)$	1 1 - 0 1	$e'(r)$	0 1 - 0
t a - 1 0 0 - 1	e a 1 0 - 0 1	$t'(r)$	0 1 - 1 0	$e'(t)$	0 1 - 1 0	$f'(t)$	0 1 - 1 0
u a - 1 0 - 0 1	f a 1 0 1 - - 1	$u'(r)$	0 1 0 0 0	$f'(u)$	0 1 0 0 0	$g'(u)$	0 1 0 0 -
v a - 1 0 0 1 -	g a - 0 0 0 1	$v'(r)$	- 1 1 0 0	$v'(v)$	- 1 1 0 0	$h'(v)$	0 1 0 1 -
w a - 1 0 1 0 -	h a 1 1 1 - - 1	$w'(r)$	0 0 0 0 -	$w'(w)$	0 0 0 0 -	$i'(w)$	- 1 0 1 -
x a - 0 0 0 1 0	i a 0 0 1 1 -	$x'(r)$	1 1 0 0 -	$x'(x)$	1 1 0 0 -	$j'(x)$	- 1 0 - 1
y a - 0 0 1 - 0	j a 0 1 0 - - 1	$y'(r)$	1 0 0 1 -	$y'(y)$	1 0 0 1 -	$k'(y)$	1 0 0 1 -
z a - 0 1 1 0 -	k a 0 1 1 0 -	$z'(r)$	-	$z'(z)$	-	$l'(z)$	-

表 I.2 入出力レベル調整と非同期安定性を考慮した P と Q の最簡形

$P_1(M)$	A+B+C+D+E	$P(m)$	A B C D E	$Q(M)$	A+B+C+D+E	$Q(m)$	A B C D E
p a 1 - 0 1 0	a' a 0 - 0 0 -	$p'(m)$	1 - 1 1 -	$p'(a)$	1 - 1 1 -	$a'(a)$	0 - 1 0 1
q a - - 1 0 0	b a 0 0 - - 0	$q'(a)$	1 1 - - 1	$q'(a)$	1 1 - - 1	$b'(a)$	- 0 1 1
r a 0 1 - - 1	c a 1 1 - - 0	$r'(a)$	0 0 - - 1	$r'(a)$	0 0 - - 1	$c'(a)$	1 0 - - 0
s a 0 0 - 1 0	d a - 1 1 1 - 0	$s'(a)$	- 1 1 0 0	$s'(a)$	- 1 1 0 0	$d'(r)$	1 1 - 0 1
t a 1 0 0 - 1	e a 1 - - 0 1	$t'(r)$	0 1 1 0	$t'(r)$	0 1 1 0	$e'(r)$	0 1 - 0
u a 1 0 1 0 -	k a 0 1 1 0 -	$z'(r)$	-	$z'(z)$	-	$l'(z)$	-

H レベル入力 C:真論理適用 (0 high 1 low)

$P_1(M)$	A+B,C-D-E	$P(m)$	A B C D E	$Q(M)$	A B C D E
p a 1 - 0 1 0	a' a 0 1 0	$p'(m)$	1 0 0 0 0	Q	1 ⊕ K
q a - - 1 0 0	b a 0 1 - - 1	$q'(a)$	1 0 0 0 1	Q'	1 ⊕ K
r a 0 0 - 1 0	c a 0 0 - - 1	$r'(a)$	- 1 0 0 0 0	t	1 ⊕ K
t a 1 0 0 - 0 1	e a 1 - - 0 1	$t'(r)$	0 1 0 0 1	t'	1 ⊕ K
u a 1 0 1 - 0 1	k a 0 1 1 - 0 1	$z'(r)$	-	z'	-

$$= A + \bar{B} + E + \bar{C}\bar{D} = \bar{A}B\bar{C}\bar{E} + \bar{A}D\bar{E}$$

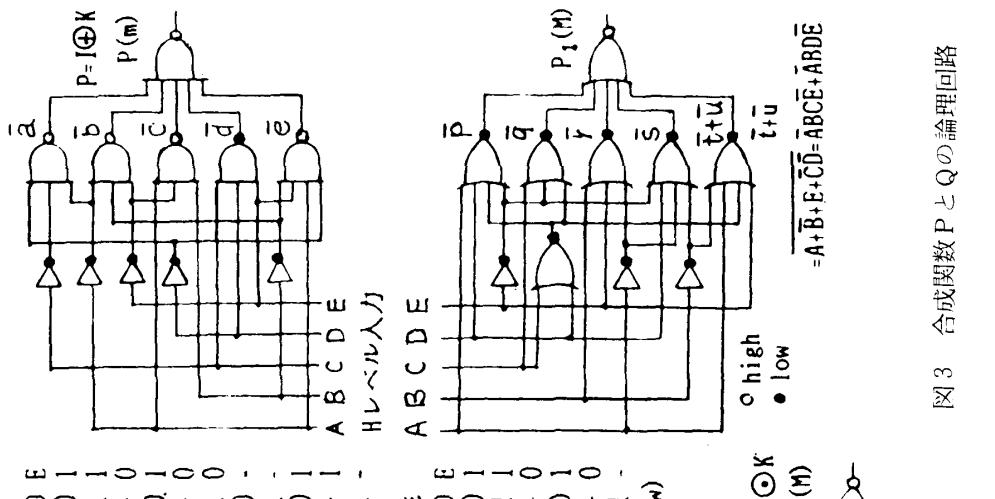


図 3 合成関数 P と Q の論理回路

$$(2, 3) P = I \oplus K ; \quad Q = I \odot K ;$$

$$(2, 4) K(m) = \overline{B} \overline{C} DE + \overline{A} \overline{C} + \overline{A} \overline{B} \overline{D} E + \overline{B} C D \overline{E} + B \overline{C} D \overline{E} + A \overline{B} C \overline{D} + \\ A B \overline{D} E + A B C D = a + b + c + f + h + i + k + n$$

$$(2, 5) K(M) = (\overline{A} + \overline{B} + C + \overline{D} + \overline{E}) (B + \overline{C} + \overline{D} + \overline{E}) (\overline{A} + B + C + E) \\ (\overline{A} + B + C + D) (A + \overline{B} + \overline{C}) (\overline{A} + D + E) (A + \overline{C} + D + E) \\ = pqrstuv$$

(2, 5) は唯一最簡形を、(2, 4) は16種類ある同値関数の一つを、付録3の読みとり(6)又は(3)により得られるもので図2のそれぞれ対応した空白部に示してある。双対図の演算はすべて最小項又は最大項でおこなわれるので同値関数を問題にする必要はない。また関数形の変換が容易であるから異種関数形の場合でも一方の番地に描いた図形を用いればよい。最小項図でのPの合成と最大項図でのQの合成はそれぞれ図2,1と2, 2に示してある。双対図の演算特性を考慮すると結果は直ちにえられ他の関数形への変換も容易である。又PとQの否定関係も図形の比較で容易に確認できる。論理式演算ではこの合成は大変なことであり、Computer Programによるとしても厄介なことであろう。得られた結果のPとQの完全主項形と最簡形は真理値表の形で図3に論理回路図と共に纏めて示してある。これらからはPは変数のすべてに正論理を適用して入出力レベルの調整が得られる関数であり、Qは変数C又はDに負論理を適用して調整が得られる関数であることがわかる。

図3のP(m)とQ(M)は入出力レベル調整と非同期安定性が共に満足なものとして示してある。表1, 2を見ると $P_1(M)$ は同値和積形のなかでは展開項、前置段共に兼用形の唯一のもので最少の素子数10個で回路化できることがわかる。しかし真理値表のE列をみると一記号のない関数であるから変数Eの時間的変化について安定性はないこともわかる。この安定性を確保するには展開項兼用形ではないがu項の変わりにv項又はw項 ($P_2(M)$) をとりあげる必要がある。このような検討は完全主項形を把握して初めてできることである。

3. 6 変数関数の完全主項による論理設計

双対図では限定された空白部分への記入であるから完全主項は多変数の場合でも比較的に容易に求められる。ただし付録2に示した演算によるチェックは必要であろう。関数Jと関数Lを引用して最簡形、入出力レベル調整、および非同期安定性を考慮した論理設計の実際例を下記する。(図4, 図5を参照)

$$(3, 1) J(m) = \sum (0, 2, 5, 7, 13, 15, 16, 18, 19, 21, 23, 25, 26, 27, 29, \\ 31, 32, 34, 35, 37, 39, 41, 45, 47, 48, 50, 56, 57, 58, \\ 59, 60, 62)$$

$$(3, 2) J(M) = \prod (62, 60, 59, 57, 55, 54, 53, 52, 51, 49, 46, 43, 41, 39, \\ 35, 33, 30, 27, 25, 23, 21, 20, 19, 17, 14, 12, 11, 10, \\ 9, 8, 2, 0)$$

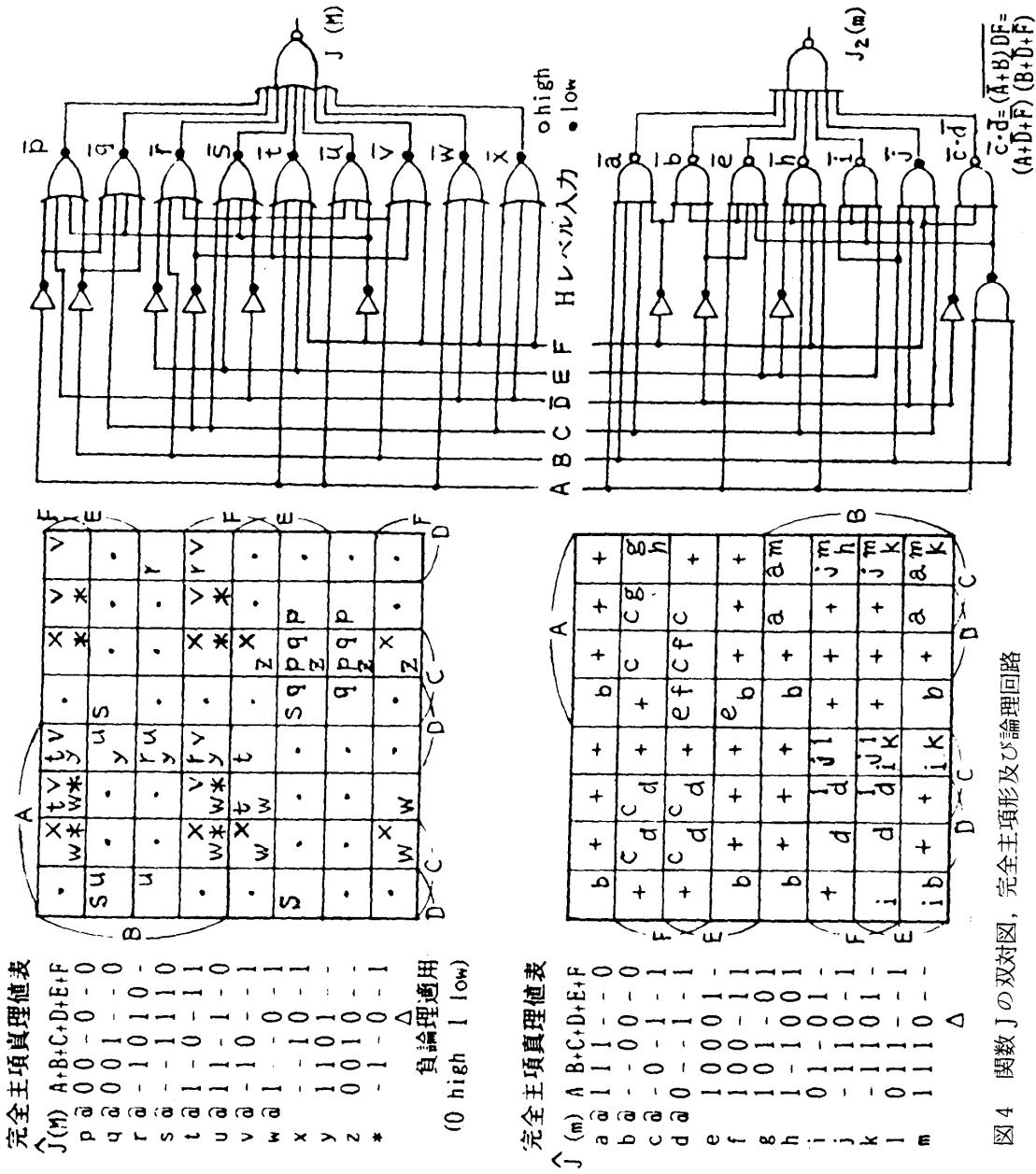
$$(3, 3) L(m) = \sum (1, 3, 4, 5, 6, 7, 13, 15, 17, 20, 22, 24, 26, 28, 30, 33, \\ 36, 38, 40, 44, 46, 49, 51, 52, 54, 56, 57, 58, 59, 60, \\ 62)$$

$$(3, 4) L(M) = \prod (63, 61, 55, 54, 53, 52, 51, 49, 47, 45, 44, 42, 40, 38, \\ 36, 34, 32, 31, 29, 26, 24, 22, 21, 20, 18, 16, 15, 13, \\ 10, 8, 2, 0)$$

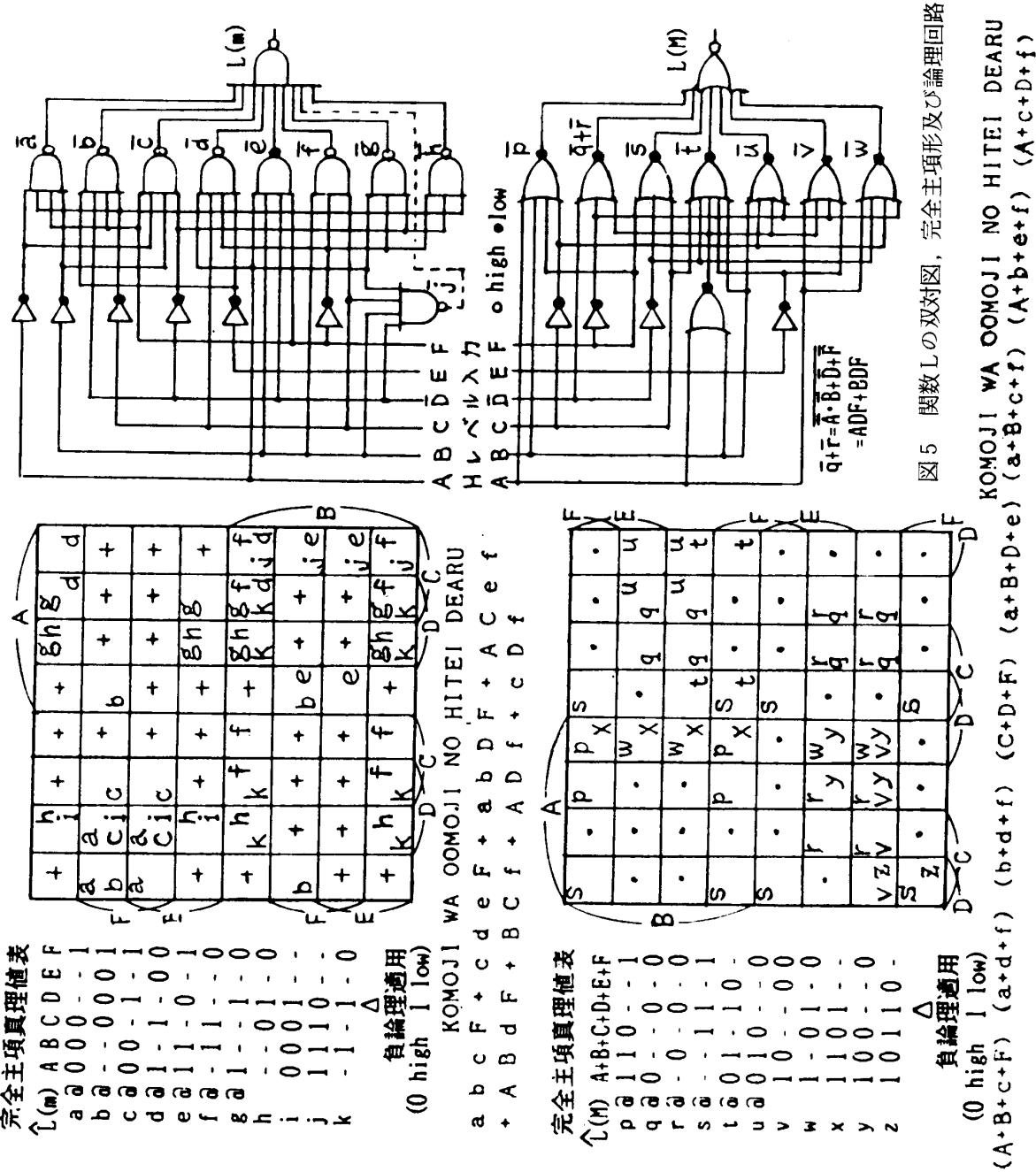
関数 J について得られた図 4 を検討すると、同図の完全主項形の真理値表に示したように、12 項の完全主項積と 13 項の完全主項和がえられる。前者では 4 主項のなかで \times 項が優先主項となり、必須項と共に唯一最簡和積形がえられ、後者では 9 主項のなかで e, f, g, h, i 及び j がとりあげられて 4 種類の同値最簡形がえられるることは、図 4 の双対図からわかることがあるが、図 7 に示した “Simple Logic” の印字出力でも確かめられる。同図の各項の真理値表を検討すると積和、和積両関数形共に負論理の適用を必要とする関数であることがわかる。また各変数列にはいずれも $-$ 記号があるからどの変数についても非同期安定性があることもわかる。また兼用性については $J(M)$ にはないが $J(m)$ は、展開項についても前置段についても兼用出来る関数形であり展開項数も少ないとから和積形よりも 3 素子少ない 13 素子で回路化出来ることがわかる。図 4 の論理回路には変数 D に負論理を適用したものが示してある。

関数 L について同様の検討をした結果は図 5 に示してある。 $L(M)$ は必須項 6 項と 2 主項の唯一最簡形で、 $L(m)$ は 4 主項のなかで h だけが最優先主項となって 8 展開項の最簡形となるものである。 $L(M)$ は展開項、前置段共に兼用出来る形であるから $L(m)$ よりも 2 個少ない数で回路化できることになる。共同期安定性については、 $L(M)$ の各項の真理値表では変数の各列にいずれも $-$ 記号があるのにたいして $L(m)$ のほうは最優先主項 h だけを取りいれた最簡形では変数 F の列に $-$ 記号がなく F の時間的変化に対しては不安定が残る関数形である。従って i 又は j の項を追加して安定性を確保する必要がある。図 5 ではそれは点線で示してある。入出力レベル調整については共に変数 D に負論理を適用したものが示してある。以上を総合するとこのような条件のもとでは関数 J については積和形が有利で、関数 L については和積形が有利と言うことになる。

このような関数形の検討には完全主項形の把握が必要であり、それにより場合によつては必要になる主項の把握も出来るのであり、単なる最簡単形の追求には注意が必要である。



单線入力NAND回路とNOR回路の簡約形の同時論理設計



4. Computer Program "Simple Logic" (脚注)

このComputer Programは論理関数の最簡形群を求める為のもので、作表法¹⁾²⁾とPetric関数法³⁾をアルゴリズムとして含み、えられた関数の論理式とその展開項真理値表を印字出力として得ようとするもので、計算機NEC-PC9801-F2 (MS-DOS)用に使用言語Fortran-77で纏めたものである。

論理関数を最小項展開形又は最大項展開形で与え、それぞれの項を論理関係式 $A \cdot B + A \cdot \bar{B} = A$ 又は $(A + B) \cdot (A + \bar{B}) = A$ を項の簡単化に用いて 1 文字づつ少ない項にする変換を繰り返し、変換終了時にえられる簡単化された項と最小項又は最大項とを組み合わせる主項表をつくる。その組合せされ方により必須項と主項がえられそれに項名をつける。ここまでが双対図での空白部への項名の記入に対応しているが詳細は他書に譲る。⁵⁾

つけられた項名を新たな変数として、元の関数の最小項又は最大項は新変数の論理和項であると見なして作成された和積関数がPetric関数である。これを積和形に展開して最少文字数の項をとりあげると元の関数の最小同値関数がそれらの文字の組合せでえられるというアルゴリズムを用いたのがPetric関数法である。³⁾ 関数 I の場合はPetric関数を P F で示すと次のようになる。

$$(3, 1) \quad P F = abc(a+b)(b+l)(b+l+f)(f+d)(d+e)(c+h)(h+g)(g+l) \\ (l+k)(k+i)(i+e+j)(j+c) \\ = abcdegk + abcdgil + abcdhil + abedgik + abcefgk + abcdefg + \\ \dots \dots \text{以下 8 文字以上の項}$$

Fortranではこの展開は文字処理になるが、データ配列の大きさに対応して演算作業用の配列も大きなものになり、かなりの記憶領域と演算時間が必要である。論理演算と印字の都合で否定変数には小文字を用いた。

Programは大別すると次の 5 部分で構成されている。

- (1) 入力制御とサブルーチン制御 (STEP 1)
- (2) 展開項の簡単化 (STEP 2)
- (3) 主項表作成と必須項取り出し (STEP EPI)
- (4) PETRIC関数処理 (STEP PETPIC)
- (5) 出力処理

前記で引用した図 6 と図 7 には負数データ入力によって生じるDATA ENDを時刻の原点として演算進行時刻の概要値を示してある。(1)は入力データと反対関数形の展開項番数算出とグループ分け整理であるが入力確認のための印字動作に隠れている。(2)は簡単化し

脚注：“Simple Logic”は著者の一人鶴見が岡山理科大学理学部昭和60年度卒業研究として纏めたものである。

て1文字少なくなった項を取り出す演算で、組合せ数値の差に相当する2進ベキ数により簡単化できる桁に-記号を与える演算である。これも多くの場合印字動作に隠れる速度であった。(3)は主項表から必須項と主項をとりだして項名を与える演算で、ここからが文字処理になる。項名と最小項(又は最大項)の組合せでPetric関数を作り(5)に引き継ぐ。(5)は和積形から積和形への()を取り外す展開演算で、ブール代数の分配則、吸収則等の基本法則によりえられる新しい項の組合せを新配列を作る演算である。最小字数の項を取り出すための作業配列に大きな記憶領域が必要である。文字処理としての演算時間も長くなり全時間の大半がここで消費される。前記K(m)の16種の同値関数演算には約28秒を要し、特別な場合にここだけに45秒かかって20種類の同値関数を得た例がある。6変数でも記憶容量不足によるLoopingを経験した。この辺がこの計算機での性能限界であろう。

変数領域毎の展開による場合には最小項又は最大項を、簡単化のために多数回取り上げる必要があり、そのための繰り返しも多くなるが、最小文字数の組合せを求めるために必要になる冗長文字組合せの演算の必要はなく、必須項、優先主項に隠れる主項の把握もできる。プログラム化については文字処理の点で有利と思われるが今後の課題としたい。

“Simple Logic”の内容は、紙面の都合で図8の概要フローチャートと図9の“Program List”的一部を示すに止めておく。後者は入力処理の一部である。

あとがき

論理設計では簡単化以前の問題として、色々の条件に対応出来ることが必要で、それには完全主項形を基盤にした設計が重要である。双対図により得られる完全主項形を用いて、入出力レベル調整と非同期安定性に対応する為に、積和、和積両関数形の双方を一挙に求めて、それらの選択により单線入力の場合の簡単化設計をする例を示した。この分野での参考になれば幸いである。

検討に参加して頂いた各位に深謝する。

文 献

- 1) McClusky, E. Minimization of Boolean functions. Bell Syst. tech. J., 1956, 35, 1417-1444.
- 2) Quin, W.V. The problem of simplyfying truth function. Am. Math. MON., 1952, 59, 521-531.
- 3) Petric, S.R. A direct determination of the irredundant form of a Boolean function from the set of prime implicants. Air Force Cambridge Research Center report Bedford, MSS. 1956. AFCRD. TR-56-110.
- 4) D.T.Ellis. A synthesis of combinational logic with NAND or NOR elements.

- IEEE Trans. Comput. EC-14, No. 5, October, 1965.
- 5) S.Muroga著, 室賀三郎, 笹尾 勤訳 論理設計とスイッチング理論 bit別冊 7, 1981, p187.
- 6) J.Toyoda Application of dual Karnaugh mapping for processing of logical switing functions and designing combinational switching circuits. Bullet. Okayama Univ. of science. No18A, p85-98. 1983.

Simultaneous Logic Design of Simplified Form of NAND Type and NOR Type Switching Circuit with Mono-Railed Input

Junso TOYODA and Masasi TSURUMI

*Department of Electronic Science, Okayama University of Science.
Mitsubishi-denki Control Software Co. Ltd.*

(Received September 30, 1986)

Abstract

Simultaneous logic design of NAND circuit and NOR circuit with monorailed input are discussed in this paper. Complete logical sum and complete logical product, which are induced by dual Karnaugh Mapping, are utilized as the designing methode. Discussion, as for 3 -staged NAND (or NOR) circuits, are of minimum form, input output level adjusting, and stability of switching. Satisfactory performance are obtained comparing with computer programmed method utilizing Petric funcion.

付録1 双対図の演算特性と完全主項形

カルノ図で積和形論理関数の図形表示ができるについては、各区画には定数の1を記憶させて対応した最小項との間に論理積演算ができる、その結果には区画間に論理和演算が適用可能と言う演算性が前提となっている。付図1.1は関数Iの展開構造を示したものであるが、カルノ図は同図の上半分を図形表示したものである。この空白部に双対の定理を適用して変数領域をカルノ図とは補完的に設定すると、和積形関数の図形表示区画になり任意の論理関数の積和形と和積形の同時表示による関数形の検討が可能になる。このような双対図には多様に活用できる論理特性があり、有力な論理設計手段として起用できる。

最大項図として図形表示された2図形間の論理演算はカルノ図の場合とは全く双対的である。カルノ図では両図形に含まれる最小項のなかで、その添字pとqの異なるものについての論理積については、 $m_p \cdot m_q = 0$ の関係が適用されることにより、定数0の部分が多くなり、添字が一致している部分だけが結果として残る。これに対して最大項図の場合上記とは双対関係にある演算 $M_p + M_q = 1$ が適用されるのは論理和の場合であり論理和演算結果は添字の一致した部分だけがえられる。また、カルノ図での論理和演算には区画間の論理和結合特性が適用されるので両図形を含む図形が結果としてえられるのに対して、最大項図では区画間論理結合特性が論理積に設定されているので論理積演算について両図形を含む図形が結果としてえられることになり、全く双対的な関係にある。

このような演算特性を用いると関数の展開演算は図形演算としてえられることになる。そのためには展開式を次のように変形しておくと変数毎に定数を代入する場合よりも完全主項形を求めることが容易になる。本文(1.1)の変数はA, B, C, ……であるとして、それぞれの変数領域の最小項の連形の論理和を $\Sigma(A; m)$ で、最大項の連形の論理積を $\Pi(A; M)$ で示し、また変数毎の積和展開形は $I_A(m)$ で、和積展開形は $I_A(M)$ で表すことで上記の図形演算特性を考慮すると、(1.1)の各式は論理演算形式で次のように表せる。

$$(A1.1) \quad I_A(m) = A \cdot \Sigma(A; m) + \bar{A} \cdot \Sigma(\bar{A}; m); \quad I_B(m) = B \cdot \Sigma(B; m) + \bar{B} \cdot \Sigma(\bar{B}; m); \\ I_C(m) = C \cdot \Sigma(C; m) + \bar{C} \cdot \Sigma(\bar{C}; m); \quad I_D(m) = D \cdot \Sigma(D; m) + \bar{D} \cdot \Sigma(\bar{D}; m); \\ I_E(m) = E \cdot \Sigma(E; m) + \bar{E} \cdot \Sigma(\bar{E}; m);$$

これらの展開項を論理和の形に纏めると完全主項和 $I(m)$ は次のようにえられる。

$$(A1.2) \quad \bar{I}(m) = I_A(m) + I_B(m) + I_C(m) + I_D(m) + I_E(m)$$

変数毎の和積形の展開形も(1.2)に対応した同様な演算で求めることができて、それらを論理積の形に纏めると完全主項積は次のようにもとめることができる。

$$(A1.3) \quad \bar{I}(M) = (A + \Pi(\bar{A}; M))(\bar{A} + \Pi(A; M))(B + \Pi(\bar{B}; M))(\bar{B} + \Pi(B; M))(C + \Pi(\bar{C}; M)) \\ (\bar{C} + \Pi(C; M))(D + \Pi(\bar{D}; M))(\bar{D} + \Pi(D; M))(E + \Pi(\bar{E}; M))(\bar{E} + \Pi(E; M)) \\ = I_A(M) \cdot I_B(M) \cdot I_C(M) \cdot I_D(M) \cdot I_E(M)$$

ここにえられた完全主項形は最小項展開形または最大項展開形に含まれる論理特性を、もれることなく備えており、そのなかのいくつかの項の組合せで最簡形がえられる。その組合せは双対図の空白部の該当した場所に記入してえられる。関数 I の積和形には必須項 3 項と主項 9 種類の組合せで 6 種類の同値関数形があり、和積形には 4 項の主項があるが u 項だけが優先主項となり必須項と共に 6 項でできた唯一最簡形があることがわかる。これは “Simple Logic” でえられる結果（図 6）とも一致しているが、最簡形に取り上げられなかった主項も把握出来る事が長所である。実際の論理演算式は付録 2 に示してある。

```

A>RUN      *** I1(m) A B C D E I2(m) A B C D E I3(m) A B C D E
*** PARAMETER INPUT   *** a q 0 1 0 1 - a q 0 1 0 1 -
INPUT NVA 2-14      b q 0 - - 1 1 b q 0 - - 1 1 b q 0 - - 1 1
5      INPUT TYPE 0=AND-OR 1=OR-AND c q 1 1 - 0 - c q 1 1 - 0 - c q 1 1 - 0 -
0      *** DATA INPUT *** TYPE = 1 d 0 0 1 0 - d 0 0 1 0 - d 0 0 1 0 -
> 3      3 31 e - 0 1 0 0 e 1 0 0 - 1 g 1 0 0 - 1 g 1 0 0 - 1
> 4      4 30 f + 0 1 0 - 1 h 1 0 1 - 0 i 1 0 1 - 0
> 5      5 29 g + 1 0 0 - 1 i 1 0 1 - 0 j 1 0 1 - 0
> 7      7 25 h + 1 0 1 - k 1 0 1 1 - l 1 0 1 1 - m 1 0 1 1 -
> 10     10 23 i + 1 0 1 1 - n 1 0 1 1 - o 1 0 1 1 -
> 11     11 22 j + 1 0 1 1 - p 1 0 1 1 - r 1 0 1 1 -
> 15     15 19 k + 1 0 1 1 - s 1 0 1 1 - t 1 0 1 1 -
> 17     17 18 l + 1 0 1 1 - u 1 0 1 1 - v 1 0 1 1 -
> 19     19 17 m + 1 0 1 1 - w 1 0 1 1 - x 1 0 1 1 -
> 20     20 15 n + 1 0 1 1 - y 1 0 1 1 - z 1 0 1 1 -
> 22     22 13 o + 1 0 1 1 - b q 0 - - 1 1 b q 0 - - 1 1 q q 0 0 0 0 0 0 -
> 23     23 10 p + 1 0 1 1 - c q 1 1 - 0 - c q 1 1 - 0 - c q 1 1 - 0 - r q 1 0 - 1 -
> 24     24 5 q + 1 0 1 1 - d 0 0 1 0 - d 0 0 1 0 - e - 0 1 0 0 s q - 1 1 - 1 -
> 25     25 4 r + 1 0 1 1 - f 1 0 0 - 1 h 1 0 0 - 1 t q 1 0 - 1 -
> 28     28 1 s + 1 0 1 1 - g 1 0 1 - 0 i 1 0 1 - 0 u 1 0 1 - 0
> 29     29 0 t + 1 0 1 1 - k 1 0 1 1 - l 1 0 1 1 - m 1 0 1 1 -
> -10    DATA END (00秒)          DATA END (00秒)
STEP 2 START          STEP 2 START
EPI START (23秒)      EPI START (23秒)
PETRIC START (24秒)    PETRIC START (24秒)
PRINT OUT START (38秒) PRINT OUT START (38秒)

```

NO= 1 KOMOJI WA OOMOJI NO HITEI DEARU
 a B c D + a D E + a B d + a b C d + b C d e + A b c E + A b C D [I₁(m)]

図6 関数1の“Simple Logic”の印字出力

```

A>RUN
*** PARAMETER INPUT ***
INPUT NVA 2-14
6
INPUT TYPE 0=AND-OR 1=OR-AND
0
*** DATA INPUT *** TYPE = 1
> 0      0      62      J1(m) A B C D E F   J2(m) A B C D E F
> 2      2      60      a @ 1 1 1 - - 0   a @ 1 1 1 - - 0
> 5      5      59      b @ - - 0 0 - 0   b @ - - 0 0 - 0
> 7      7      57      c @ 0 - - 1 - 1   c @ 0 - - 1 - 1
> 13     13     55      d @ - 0 - 1 - 1   d @ - 0 - 1 - 1
> 15     15     54      e 1 0 0 0 1 -   e 1 0 0 0 1 -
> 16     16     53      f 0 1 - 0 1 -   f 0 1 - 0 1 -
> 18     18     52      g 1 0 1 - 0 1   g 1 0 1 - 0 1
> 19     19     51      h 1 - 1 0 - 1   h 1 - 1 0 - 1
> 21     21     49      i - 1 1 0 - 1   i - 1 1 0 - 1
> 23     23     46      j - 1 1 0 - 1   j - 1 1 0 - 1
> 25     25     43      J3(m) A B C D E F   J4(m) A B C D E F
> 26     26     41      a @ 1 1 1 - - 0   a @ 1 1 1 - - 0
> 27     27     39      b @ - - 0 0 - 0   b @ - - 0 0 - 0
> 29     29     35      c @ 0 - - 1 - 1   c @ 0 - - 1 - 1
> 31     31     33      d @ - 0 - 1 - 1   d @ - 0 - 1 - 1
> 32     32     30      e 1 0 0 - 1 1   e 1 0 0 - 1 1
> 34     34     27      f 1 0 0 - 1 1   f 1 0 0 - 1 1
> 35     35     25      g 1 0 1 - 0 1   g 1 0 1 - 0 1
> 37     37     23      h 1 - 1 0 0 1   h 1 - 1 0 0 1
> 39     39     21      i 0 1 - 0 1 -   i 0 1 - 0 1 -
> 41     41     20      j - 1 1 0 - 1   j - 1 1 0 - 1
> 45     45     19      J(M) A+B+C+D+E+F
> 47     47     17      p @ 0 0 - 0 - 0
> 48     48     14      q @ 0 0 1 - - 0
> 50     50     12      r @ - 1 0 1 0 -
> 56     56     11      s @ - - 1 1 1 0
> 57     57     10      t @ 1 - 0 - 1 1
> 58     58      9      u @ 1 1 - 1 - 0
> 59     59      8      v @ - 1 0 - - 1
> 60     60      2      w @ 1 - - 0 - 1
> 62     62      0      x - - 1 0 - 1
> -10

DATA END      (00秒)

STEP 2 START
EPI START      (24秒)
PETRIC START    (25秒)
PRINT OUT START (45秒)

NO= 1      KOMOJI WA OOMOJI NO HITEI DEARU
A B C f + c d f + a D F + b D F + A b c d E +
A b C e F + a B d E + B C d F [J1(m)]

```

図7 関数Jの“Simple Logic”の印字出力

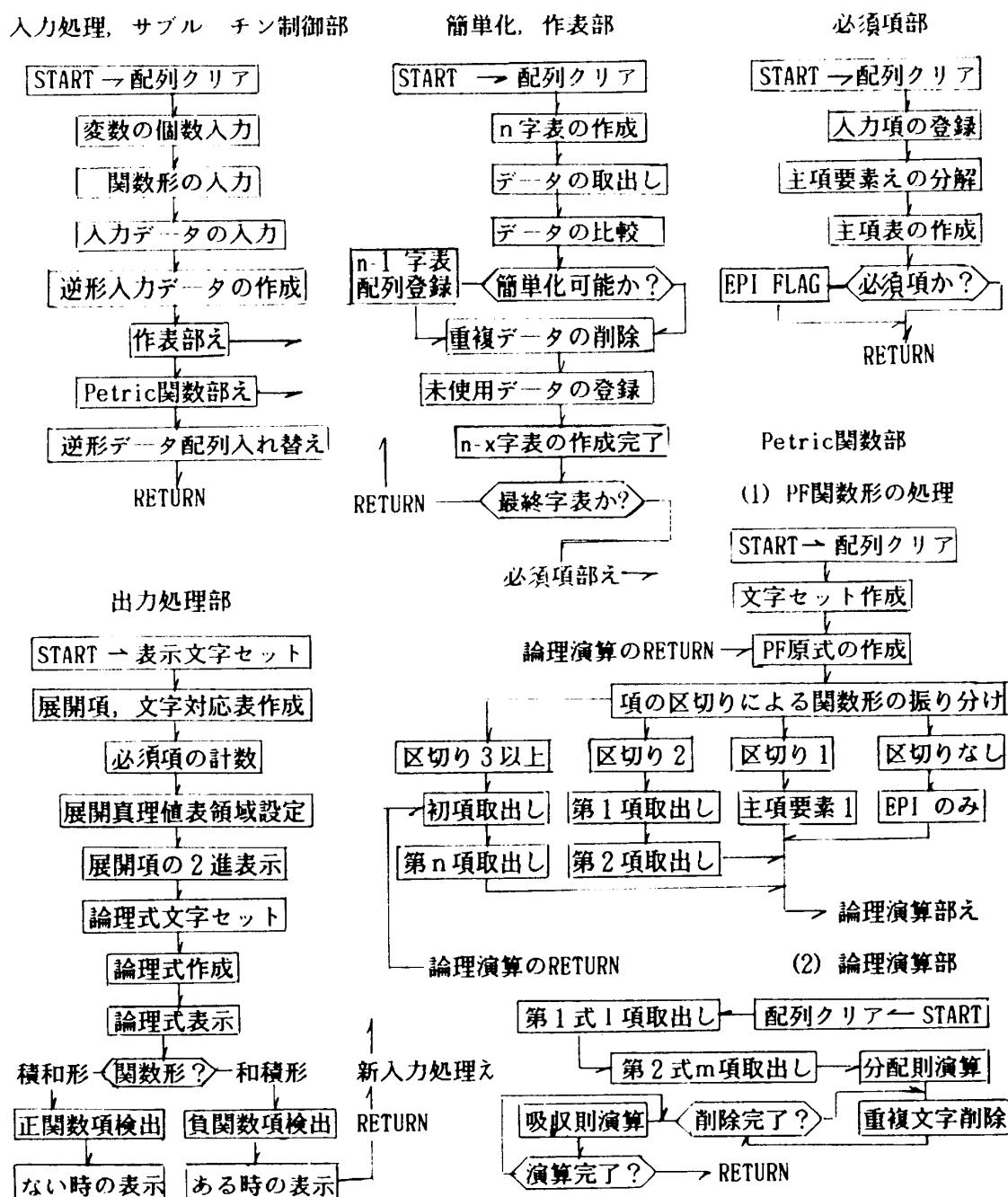


図8 “Simple Logic”の概要フローチャート

```

PC-9801 MS-FORTRAN コンパイラによる FORTRAN LISTING FILE リスト
1 ****
2 **
3 **      SIMPLE LOGIC PROGRAM
4 **
5 ****
6     PROGRAM INPUT1
7     INTEGER*2 MAXVA,MAXID,NVA,TYPE,ID,LOOP,LOOP1,D1,D2,POS,BIN1,NTYPE
8     - ,BDL(1024),IDL(1024),IDP(12),NID,EL,SP
9     - ,AAA(1024,4),AAAC,HSUM(1024,2),INV
10    CHARACTER*1 SQE(930,70)
11    COMMON /BDL/BDL,/ID/IDL, IDP,/AAA/AAA,/HSUM/HSUM,/SQE/SQE
12    DO 100 LOOP=1,1024
13      BDL(LOOP)=0
14      IDL(LOOP)=0
15      AAA(LOOP,1)=0
16      AAA(LOOP,2)=0
17      AAA(LOOP,3)=0
18      AAA(LOOP,4)=0
19      HSUM(LOOP,1)=0
20      HSUM(LOOP,2)=0
21    100 CONTINUE
22    DO 110 LOOP=1,12
23      IDP(LOOP)=0
24    110 CONTINUE
25    DO 120 LOOP=1,930
26      DO 130 LOOP1=1,70
27        SQE(LOOP,LOOP1)=' '
28    130 CONTINUE
29    120 CONTINUE
30    WRITE (*,*) '*** PARAMETER INPUT ***'
31    MAXVA=14
32    MAXID=16383
33    10 WRITE (*,*) 'INPUT NVA 2-14'
34    READ (*,*) NVA
35    IF ( ( NVA .LT. 2 ) .OR. ( NVA .GT. MAXVA ) ) THEN
36      WRITE (*,1000) 1
37      GOTO 10
38    ENDIF
39    20 WRITE (*,*) 'INPUT TYPE 0=AND-OR 1=OR-AND'
40    READ (*,*) TYPE
41    IF ( ( TYPE .NE. 0 ) .AND. ( TYPE .NE. 1 ) ) THEN
42      WRITE (*,1000) 2
43      GOTO 20
44    ENDIF
45    140 WRITE (*,*) 'INV TABLE 0=NO 1=YES 99=END'
46    READ (*,*) INV
47    IF ( INV .EQ. 99 ) GOTO 210
48    IF ( ( INV .NE. 0 ) .AND. ( INV .NE. 1 ) ) THEN
49      WRITE (*,1000) 2
50      GOTO 140
51    ENDIF
52    WRITE (*,*) '*** DATA INPUT ***'
53    30 WRITE (*,1010)
54    READ (*,*) ID
55    IF ( ID .LT. 0 ) THEN
56      WRITE (*,*) 'DATA END'
57      GOTO 40
58    ENDIF
59    IF ( ID .GT. (2**NVA-1) ) THEN

```

;; COMMON領域の初期値設定

;; 取り扱える変数の最大個数
;; $2^{14} - 1$

;; 変数個数の入力

;; 変数個数規定範囲外のエラーメッセージ

;; 積和入力か和積入力か

;; エラーメッセージ

;; 反転真理値表の作成についての制御

;; 最小項又は最大項の入力制御

;; 入力終了

;; 入力項チェック開始

図9 “Simple Logic” Program Listの一部

付録 2

関数 I の完全主項形の論理演算 (A1;2,3の演算)

完全主項和は各変数領域毎に最簡積和形を作り次のように求められる。

$$I_A(m) = A(m_{17}m_{19}m_{20}m_{22}m_{23}m_{24}m_{25}m_{26}m_{27}m_{28}m_{29}) + \bar{A}(m_3m_4m_5m_7m_{10}m_{11}m_{15}) = \bar{A}\bar{B}\bar{C}D + \bar{A}DE + A\bar{B}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{E} + A\bar{B}\bar{C}\bar{E} + A\bar{B}\bar{C}D$$

$$I_B(m) = B(m_{10}m_{11}m_{15}m_{24}m_{25}m_{26}m_{27}m_{28}m_{29}) + \bar{B}(m_3m_4m_5m_7m_{17}m_{19}m_{20}m_{22}m_{23}) = \bar{A}\bar{B}\bar{C}D + \bar{A}DE + A\bar{B}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{E} + A\bar{B}\bar{C}\bar{E} + \bar{B}DE$$

$$I_C(m) = C(m_4m_5m_7m_{15}m_{20}m_{22}m_{23}m_{28}m_{29}) + \bar{C}(m_3m_{10}m_{11}m_{17}m_{19}m_{24}m_{25}) = \bar{A}\bar{B}\bar{C}D + \bar{A}DE + A\bar{B}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{E} + AC\bar{D}\bar{E} + A\bar{B}CD$$

$$I_D(m) = D(m_3m_7m_{10}m_{11}m_{15}m_{19}m_{22}m_{23}) + \bar{D}(m_4m_5m_7m_{17}m_{20}m_{24}m_{25}m_{28}m_{29}) = \bar{A}\bar{B}\bar{C}D + \bar{A}DE + A\bar{B}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{C}\bar{D}\bar{E} + A\bar{B}\bar{C}\bar{E} + \bar{B}DE$$

$$I_E(m) = E(m_3m_5m_7m_{11}m_{15}m_{17}m_{19}m_{23}m_{25}m_{29}) + \bar{E}(m_4m_{10}m_{20}m_{22}m_{24}m_{28}) = \bar{A}\bar{B}\bar{C}D + \bar{A}DE + A\bar{B}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{E} + \bar{B}\bar{C}\bar{D}\bar{E} + A\bar{B}\bar{C}D + A\bar{B}\bar{C}E$$

これらの展開項を論理和に纏めると完全主項和 $\hat{I}(m)$ は次のようにえられる。

$$\hat{I}(m) = \bar{A}\bar{B}\bar{C}D + \bar{A}DE + A\bar{B}\bar{D} + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{B}\bar{C}\bar{D}\bar{E} + \bar{A}\bar{B}\bar{C}\bar{E} + A\bar{B}\bar{C}\bar{E} + A\bar{C}\bar{D}\bar{E} + A\bar{B}\bar{C}\bar{D} + \bar{B}DE$$

完全主項積については上記と双対関係にある演算で次のようにえられる。

$$I_A(M) = (A+m_{17}m_{18}m_{19}m_{22}m_{23}m_{25}m_{29}m_{30}m_{31})(\bar{A}+m_0m_1m_4m_5m_{10}m_{13}m_{15}) = (A+C+D)(\bar{A}+\bar{B}+D)(A+B+\bar{D}+E)$$

$$(A+\bar{C}+\bar{D}+E)(B+C+E)(\bar{A}+\bar{B}+\bar{D})(\bar{A}+B+\bar{C}+D+\bar{E})$$

$$I_B(M) = (B+m_{10}m_{13}m_{15}m_{25}m_{29}m_{30}m_{31})(\bar{B}+m_0m_1m_4m_5m_{17}m_{18}m_{19}m_{22}m_{23}) = (B+C+E)(\bar{B}+A+D)(\bar{B}+\bar{A}+\bar{D})(B+A+\bar{D}+E)$$

$$(\bar{B}+A+\bar{C}+E)(\bar{A}+B+\bar{C}+D+\bar{E})(A+C+D)$$

$$I_C(M) = (C+m_4m_{13}m_{15}m_{22}m_{23}m_{29}m_{30}m_{31})(\bar{C}+m_0m_1m_{10}m_{17}m_{18}m_{19}m_{25}) = (C+B+E)(A+C+D)(\bar{A}+B+\bar{C}+D+\bar{E})$$

$$(\bar{C}+A+\bar{D}+E)(C+\bar{A}+\bar{D}+E)(A+\bar{B}+D)(\bar{A}+\bar{B}+\bar{D})$$

$$I_D(M) = (D+m_{10}m_{15}m_{18}m_{19}m_{22}m_{23}m_{30}m_{31})(\bar{D}+m_0m_1m_4m_5m_{13}m_{17}m_{25}m_{29}) = (D+A+C)(D+A+\bar{B})(\bar{D}+\bar{A}+\bar{B})$$

$$(\bar{D}+A+\bar{B}+E)(\bar{B}+A+\bar{C}+E)(\bar{A}+B+\bar{C}+D+\bar{E})(B+C+E)$$

$$I_E(M) = (E+m_1m_5m_{13}m_{15}m_{17}m_{19}m_{23}m_{25}m_{29}m_{31})(\bar{E}+m_0m_4m_{10}m_{18}m_{22}m_{30}) = (E+B+C)(E+A+\bar{C}+\bar{D})(E+A+\bar{B}+\bar{C})$$

$$(\bar{E}+\bar{A}+\bar{B}+\bar{C}+D)(A+C+D)(A+\bar{B}+D)(\bar{A}+\bar{B}+\bar{D})$$

これらの展開項を論理積に纏めると完全主項積 $\hat{I}(M)$ は次のようにえられる。

$$\hat{I}(M) = (\bar{A}+B+\bar{C}+D+\bar{E})(\bar{A}+\bar{B}+\bar{D})(A+\bar{B}+D)(B+C+E)(A+C+D)(A+\bar{C}+\bar{D}+E)$$

$$(A+B+\bar{D}+E)(\bar{A}+C+\bar{D}+E)(A+\bar{B}+\bar{C}+E)$$

付録3 双対図からの論理情報の読み取り

双対図による関数形の検討には・記号または+記号の図の空白部に英小文字の展開項名を記入したものが便利である。（関数Iの双対図 参照）双対図としての演算特性により、このような図形から多様な論理情報がえられるが、その主要なものについて下記する。

- (1) 最小項番地で・記号图形をよみとると関数の積和形がえられる。
- (2) 最小項番地で・記号图形の空白部をよみとると積和形になった否定関数がえられる。
- (3) 最大項番地で・記号图形の空白部をよみとると関数の和積形がえられる。
- (4) 最大項番地で+記号图形をよみとると関数の和積形がえられる。
- (5) 最大項番地で+記号图形の空白部をよみとると和積形になった否定関数がえられる。
- (6) 最小項番地で+記号图形の空白部をよみとると関数の積和形がえられる。
- (7) ・記号图形の番地を最大項番地に変更してよみとると否定関数の和積形がえられる。
- (8) +記号图形の番地を最小項番地に変更してよみとると否定関数の積和形がえられる。
- (9) (3)でえられた関数形を積和形に展開すると元の関数の完全主項和がえられる。
- (10) (6)でえられた関数形を和積形に展開すると元の関数の完全主項積がえられる。

注記：(9), (10)は変数毎の展開による最簡形の全てを纏めたものである。（付録2 参照）

(7), (8)はモルガンの定理に対応する。

(3), (6)は(9), (10)の簡略法で最簡形だけでよい場合には便利である。

付図1.1 関数Iの展開構造

積和形	$m_3 + m_4 + m_5$	m_7	$m_{10} + m_{11}$	m_{15}	m_{17}	$m_{19} + m_{20}$	$m_{22} + m_{23} + m_{24} + m_{25}$	$m_{28} + m_{29}$
定数配置	0 0 0 1 1 1 0 0 1 1 0 0 0 1 0 0 0 1 0 1 1 0 1 1 1 0 0 1 1 0 0							
	+ + +	+ + +	+ + +	+ + +	+ + +	+ + +	+ + +	+ + +
和積形	$m_{31} m_{30} m_{29}$	m_{25}	$m_{23} m_{22}$	$m_{19} m_{18} m_{17}$	$m_{15} m_{14}$	$m_{13} m_{12}$	$m_{10} m_9 m_8$	$m_6 m_5 m_4$

付図1.2 関数Iの展開構造

積和形	$m_0 + m_1 + m_2$	m_6	$m_8 + m_9$	$m_{12} + m_{13} + m_{14}$	m_{16}	m_{18}	$m_{21} + m_{22}$	$m_{26} + m_{27}$	$m_{30} + m_{31}$
定数配置.	1 1 1 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 1 0 1 0 0 1 0 0 0 1 1 0 0 1 1								
	+ + +	+ + +	+ + +	+ + +	+ + +	+ + +	+ + +	+ + +	+ + +
和積形	$m_{28} m_{27} m_{26}$	m_{24}	$m_{21} m_{20}$	m_{16}	m_{14}	$m_{12} m_{11}$	$m_9 m_8 m_7$	$m_6 m_5 m_4$	$m_3 m_2$

付図1.3 関数I (m) の表示

付図1.4 関数I (M) の表示

付図1.5 関数Iの項名表示

s	t	s	u	d	r	s	i	j
t	l	b	l	b	d	g	l	p
r	a	b	r	c	q	q	c	
s	t	a	u	r	c	q	q	c

$$I(m) = a + b + c + d + g + i + l$$

付図2 5変数番地図
付図3 6変数番地図

付図3 6変数番地図

00	02	06	04	16	18	22	20
01	03	07	05	17	19	23	21
09	11	15	13	25	27	31	29
08	10	14	12	24	26	30	28
				D	C	C	D
				E	F	B	E
00	04	12	08	32	36	44	40
01	05	13	09	33	37	45	41
03	07	15	11	35	39	47	43
02	06	14	10	34	38	46	42
				D	C	C	D
				E	F	B	E
63	59	51	55	31	27	19	23
62	58	50	54	30	26	18	22
60	56	48	52	28	24	16	20
61	57	49	53	29	25	17	21
				D	C	C	D
				E	F	B	E
47	43	35	39	15	11	03	07
46	42	34	38	14	10	02	06
44	40	32	36	12	08	00	04
45	41	33	37	13	09	01	05
				D	C	C	D
				E	F	B	E
31	29	27	15	13	09	11	
30	28	24	26	14	12	08	10
22	20	18	16	06	04	00	02
23	21	17	19	07	05	01	03
				D	C	C	D
				E	F	B	E

付図2 5変数番地図