高岡短期大学紀要　第17巻　平成14年３月
Bull. Takaoka National College, Vol.17, March 2002

多値論理回路に関する一設計法

藤田徹也　荒井　勇＊　高松　衛＊　中嶋芳雄＊

要　旨

本稿では、２値論理に関する出力多段NANDゲートの一設計法であるMA法を、多値論理系へ拡張する手法を示し、生成される初期回路の簡単化を含めた本方法のプログラム化と、種々の多値論理関数に対するシミュレーションの結果について述べる。

結果として、多値論理は２値論理に比べて入力線数がより少なく設計できることが裏付けられた。また、簡単化の結果、初期回路は入力線数が77%、ゲート数が94%、段数が96%の規模まで縮小することができた。

キーワード

NANDゲート、MA法、多値論理系、簡単化、４値論理関数

１　はじめに

近年のパーソナルコンピュータに搭載されるCPUなどの高集積回路では、論理演算の手法として２値論理が広く用いられ定着した技術となっている。これに対し、２値論理にはない利点を持つとされる理論が多値論理である。多値論理回路では、入力線数が２値論理に比べ少なくなると言われている。

我々は、２値論理回路をNANDゲートのみで設計する手法、MA法[1]を発表してきた。複雑な高集積回路といえども、その回路を構成する要素はNANDゲートのような単純な素子から構成される。その中でNANDゲート（あるいはNOR）のみによる論理回路設計は、構造の一様性から集積回路化に適している。

多値論理においても、２値論理の場合と同様に高集積化のための同一単一ゲートでの論理展開ができることが重要である。そこで本研究では、様々な多値論理系の中からAND、ORの他に基本演算子として${}^{a}X^{b}$を考え、これに否定演算子$\overline{{}^{a}X^{b}}$を加えた多値論理を扱うSuらの多値論理系[2]にMA法を拡張する手法を述べる。

２　多値論理系

多値論理においては、２値論理と異なり、AND、ORの他にこれまでに、種々の基本論理演算子が考えられ、様々な多値論理系とそれに見合う主項が定義されてきた。

その中で、AND、ORの他に基本演算子${}^{a}X^{b}$としてと否定演算子$\overline{{}^{a}X^{b}}$を定義したSuらの多値論理系にＭＡ法を拡張する。多値論理系で否定演算が定義されていると、２値論理のように、ド・モルガンの定理、$\overline{A \cdot B}=\overline{A}+\overline{B}$及び、$\overline{A+B}=\overline{A} \cdot \overline{B}$が成り立つ。そのため多値論理系でもORゲートやANDゲートがNANDゲートのみで表せることになり、２値論理回路をNANDゲートのみで実現するMA法が、多値論理系にも適用できると考えられる。

２.１　基本となる定義、及び定理

ここでは、まず多値論理系の基本となる定義、及び、定理について述べる。

地域ビジネス学科　＊富山大学工学部

2.1.1　準備

（定義１）変数X_1，X_2，…，X_nは、p値論理変数で$L=\{0, 1, 2, \cdots, p-1\}$の値をとる。$p$値$n$変数論理関数$f(X_1, X_2, \cdots, Xn)$は，$L^n$から$L$への写像である。図２．１のマップ（４値３変数論理関数Fの例）で与えることもできる。

X_1 \ X_2	0				1				2				3			
X_3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
0	1	1	1	1	2	3	2	3	1	1	1	1	1	1	1	1
1	1	1	1	1	3	3	3	3	1	1	1	1	1	1	1	1
2	1	1	1	1	2	3	2	3	1	1	1	1	1	1	1	1
3	1	1	1	1	3	3	3	3	1	1	1	1	1	1	1	1

図２．１　４値３変数論理関数F

完全系として、次の演算を考える。

（定義２）

OR演算　$Xi+Xj=\max(Xi, Xj)$　(1)

AND演算　$Xi \cdot Xj=\min(Xi, Xj)$　(2)

NOT演算　$\overline{Xi}=(p-1)-Xi$　(3)

（定義３）

$$ {}^{a}X^{b}=\begin{cases} p-1 & a \leqq X \leqq b \\ 0 & \text{その他} \end{cases} \quad (4)$$

$$ \overline{{}^{a}X^{b}}=\begin{cases} p-1 & X<a \text{　又は　} X>b \\ 0 & \text{その他} \end{cases} \quad (5)$$

但し、a，$b \in L$、かつ、$a \leqq b$である。${}^{a}X^{b}$、及び、$\overline{{}^{a}X^{b}}$をリテラルと呼び、冗長でないリテラルの積形式

$$X={}^{a1}X_1^{b1} \cdot {}^{a2}X_2^{b2} \cdot \cdots \cdot {}^{an}X_n^{bn} \quad (6)$$

を項という。

（定理１）［４］n変数の任意の多値論理関数fは、積和形式

$$f=\sum_k (C_k \cdot {}^{ak1}X_1^{bk1*} \cdot {}^{ak}X_2^{bk2*} \cdot \cdots \cdot {}^{akn}X_n^{bkn*}) \quad (7)$$

で表すことができる。

但し、C_k，a_{ki}，かつ，b_{ki}は、Lの値をとる。ド・モルガンの定理を用いて、積和形式を

$$f=\prod_k (\overline{C_k} \cdot \overline{{}^{ak1}X_1^{bk1*}} \cdot \overline{{}^{ak}X_2^{bk2*}} \cdot \cdots \cdot \overline{{}^{akn}X_n^{bkn*}}) \quad (8)$$

の和積形式に変換できる。

但し、${}^{aki}X_n^{bki*}={}^{aki}X_n^{bki}$又は、$\overline{{}^{aki}X^{bki}}$とする。

2.1.2　多値論理関数

例題として、図２．１の４値３変数関数Fを考えると、変数X_1，X_2，X_3は、４値論理変数で$L=\{0, 1, 2, 3\}$の値をとる。４値３変数関数$f(X_1, X_2, X_3)$は、L^3からLへの写像である。

積和形式で表すと

$$F=1+2 \cdot {}^{1}X_2^{1}+3 \cdot {}^{1}X_1^{1} \cdot {}^{1}X_2^{1}+3 \cdot {}^{3}X_1^{3} \cdot {}^{1}X_2^{1} +3 \cdot {}^{1}X_2^{1} \cdot {}^{1}X_3^{1}+3 \cdot {}^{1}X_2^{1} \cdot {}^{3}X_3^{3} \quad (9)$$

となる。

又、図２．１の４値３変数関数Fの関数$\overline{F}$をマップで表したものを図２．２に示す。

X_1 \ X_2	0				1				2				3			
X_3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
0	2	2	2	2	1	0	1	0	2	2	2	2	2	2	2	2
1	2	2	2	2	0	0	0	0	2	2	2	2	2	2	2	2
2	2	2	2	2	1	0	1	0	2	2	2	2	2	2	2	2
3	2	2	2	2	0	0	0	0	2	2	2	2	2	2	2	2

図２．２　４値３変数関数$\overline{F}$

図２．２のマップを積和形式で表すと、

$$\overline{F}=2 \cdot \overline{{}^{1}X_2^{1}}+1 \cdot {}^{0}X_1^{0} \cdot {}^{0}X_3^{0}+1 \cdot {}^{0}X_1^{0} \cdot {}^{2}X_3^{2} +1 \cdot {}^{2}X_1^{2} \cdot {}^{0}X_3^{0}+1 \cdot {}^{2}X_1^{2} \cdot {}^{2}X_3^{2} \quad (10)$$

となり、（定理１）から、

$$F=(1+{}^{1}X_2^{1}) \cdot (2+\overline{{}^{0}X_1^{0}}+\overline{{}^{0}X_3^{0}}) \cdot (2+\overline{{}^{0}X_1^{0}}+\overline{{}^{2}X_3^{2}}) \cdot (2+\overline{{}^{2}X_1^{2}}+\overline{{}^{0}X_3^{0}}) \cdot (2+\overline{{}^{2}X_1^{2}}+\overline{{}^{2}X_3^{2}}) \quad (11)$$

を導ける。

2.2　キューブ表現

p値n変数関数fは、幾何学的な表現としてn次元のキューブ（又は項）P^nに写像することができる。P^nの頂点$(e_1, e_2, \cdots, e_n)$のn個の組は、入力変数$(X_1, X_2, \cdots, X_n)$のn個の組にそれぞれ対応している。e_iはLの中の値をとる。各e_iは、pビット$(b_0, b_1, \cdots, b_{p-1})$からなり、$e_i$の$bj$ $(j=Xi)$のビットが１、その他はビット０で表現される。

例えば、３値３変数関数F_1［$L=\{0,1,2\}$, $V=\{X_1, X_2, X_3\}$］において、キューブの頂

点が、3入力の組(X_1，X_2，X_3)=(2，1，0)である時、X_1=2であるから、e_1=(b_{10}，b_{11}，b_{12})のb_{10}=b_{11}=0，b_{12}=1となり、e_1=(001)である。キューブ表現すると、$e_1e_2e_3$=(001)(010)(100)となる。

2.3　項表現とキューブ表現の関係

2.2節で、項をキューブ表現する方法を示したが、リテラル${}^aX^b$、$\overline{{}^aX^b}$の積項をビット列で表現することになる。この変換のために、演算子Mをここで与える。

リテラルをビット列に変換する利点は、論理演算を行う上で有益である。特に多値MA法のアルゴリズムをプログラム化する際の、論理演算、及び許容項の表現等に最適な表現方法である。

2.3.1　ビット列演算子M

リテラルをビット列で表現すると、変換演算子Mを用いて次のように示せる。

$$M({}^aX^b)=\underbrace{00\ldots0}_{a}\underbrace{11\ldots10\ldots0}_{b+1}$$

また、リテラルの否定$\overline{{}^aX^b}$は、リテラルのビット列表現を反転したものとなる。

$$M(\overline{{}^aX^b})=\underbrace{11\ldots1}_{a}\underbrace{00\ldots01\ldots1}_{b+1}$$

例えば、4値3変数関数でX，Y，Zは、4値論理変数でL={0，1，2，3}の値をとる時の以下のリテラルのビット列表現、及び、ビット列表現を用いた種々の演算を行うことができる。

リテラル　${}^1X^3$　→　$M({}^1X^3)$=(0111)

リテラルの否定　$\overline{{}^1X^3}$

→　$M(\overline{{}^1X^3})$=(1000)

項　$2\cdot{}^1X^3\cdot{}^2Z^3\rightarrow M(2\cdot{}^1X^3\cdot{}^2Z^3)$

=$2\cdot M({}^1X^3\cdot{}^0Y^3\cdot{}^2Z^3)$

=2(0111)(1111)(0011)

AND演算　$\overline{{}^0X^0}\cdot\overline{{}^3X^3}\rightarrow M(\overline{{}^0X^0})\cdot M(\overline{{}^3X^3})$

=(0111)・(1110)=(0110)　→　${}^1X^2$

OR演算　${}^1X^2+{}^3X^3\rightarrow +M({}^1X^2)+M({}^3X^3)$

=(0110)+(0001)=(0111)　→　${}^1X^3$

3　2値論理のMA法

MA法とは、2値論理における一出力多段NANDゲート回路を生成する一設計法であり、肯定変数のみの積項で表される許容項を利用して論理関数の“1”の部分を実現していく。もしその許容項に“0”の部分を含めば、それを取り除く許容項を生成していき全部“1”、又は、全部“0”の許容項となるまで、この手法を繰り返して許容項の木を生成する。この許容項の木から一定の規則でNANDゲート回路に置き換える回路設計法である。

MA法は、積和形式で論理関数や許容項を扱っているが、ここからは分かりやすく関数Fをカルノー図などのマップで与えて説明する。

3.1　2値論理のMA法のアルゴリズム

2値論理関数がマップで与えられているとする。マップの各セルは、2進座標(X_1，X_2,...，X_n)で表せるので、これを10進数に直して番号付ける。各セル毎に許容項が生成され、n変数のセルj(X_{1j}，X_{2j},...，X_{nj})に対して、各セルの座標成分をセルk(X_{1k}，X_{2k},...，X_{nk})とすると、セルjの座標成分に対して、座標成分がそれぞれ大きいか、又は、等しいセルkの集まりを許容項と言う。

2値論理のMA法では、最初に木の根に相当する許容項を出力ゲートP_0として、以下の方法で初期回路を生成する。

(1)真理値1のセルを、セル番号の小さい順に許容項P_1，P_2，…，P_iを生成してすべて覆う。

(2)(1)の許容項P_j(j=1，2，…，i)について真理値0のセルを含むものがあれば、同様にセル番号の小さい順に真理値0のセルを取り除くように許容項を生成する。さらに、

⑵の許容項の中に、真理値１のセルがあれば、そのセルを関数上実現するため、ステップ⑴へ戻る（真理値１のセルを復活させる）。

この過程を許容項の中に真理値１と０が混在しなくなるまで続ける。こうして、できた許容項の木（図３．１）から許容項をゲートに置き換え、許容項をなす変数をその入力リテラルとする。また、その許容項から生成した許容項からなるゲートの出力を入力として、多段NANDゲート回路（図３．２）に置き換える。

２値４変数関数$F=X_1X_2+\overline{X}_1X_4+\overline{X}_3X_4$を例に、ＭＡ法を適用して初期回路を生成する。許容項の木を図３．１、そして、初期回路を図３．２に示す。

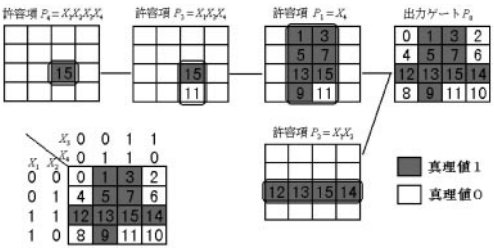

図３．１　許容項の木

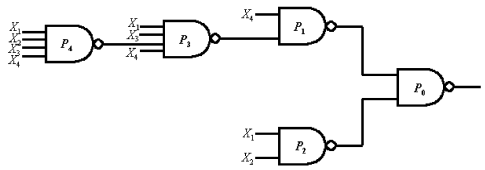

図３．２　関数Fの初期回路

４　多値ＭＡ法

従来のＭＡ法では、２値論理を扱ってきたが、多値MA法では真理値が増加するので２値論理のMA法のアルゴリズムを改良する必要がある。

多値論理を扱う上で、従来の２値論理系とは別に多値論理系で、リテラル、項を表現し、また、論理演算を行わねばならない。そこで本方法では、先に述べた否定演算の定義されたSuらの多値論理系にMA法を拡張する。

本章以降で示される回路図の中のNANDゲート素子は、多値NANDゲート素子である。

４．１　基本アルゴリズム

p値n変数論理関数をfとして、多値MA法の基本アルゴリズムを以下に示す。以下に“MA法”と記述してあるものは、２値論理のMA法である。また“true”は、２値論理における“１”であり、“false”は、“０”を表す。

⑴　真理値$k=p-1$から始める。

⑵　真理値kのセルを、trueのセル、kより小さいセルをfalseのセル、そして、kより大きいセルがあれば、そのセルをドント・ケアdのセルとしてMA法を適用する。

⑶　真理値kが$k=1$の場合は、ここで多値MA法を終了する。$k>1$の場合は、$k=k-1$としてステップ⑵に戻る。

ステップ⑴から⑶より、生成されたk毎の許容項の木を回路に変換するため、次に進む。

⑷　真理値k毎に得られた許容項の木をそれぞれの回路へ変換する。各回路は、出力ゲートを持つが、回路の出力と真理値kを入力とするNANDゲート（新たに真理値kの回路の出力となる）を付加する。

⑸　k毎の回路の出力ゲートを入力として持つ、NANDゲートですべての回路を１つにまとめる。この回路を多値MA法の初期回路とする。（図４．１）

ドント・ケアdとは、真理値を“true”、“false”どちらとも考えられるという意味であり、許容項の生成個数を削減できる。すなわち初期回路のゲート数、入力線数を減らす効果がある。

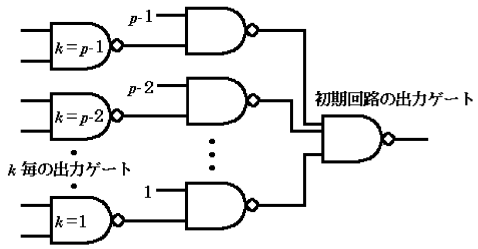

図４．１　真理値k毎の回路の合成(初期回路)

４．２　多値論理での許容項の基本概念

多値MA法では、２値論理のMA法と同様に、許容項を利用して許容項の木を生成し、初期回路を設計する。

p値n変数関数fで、この関数をマップで表すとセルの数は、p^n個となる。各セルは、p進座標(X_1，$X2$,...，X_n)で表されるので、これを10進数に変換して番号付ける。各セル毎に許容項が生成され、セルj(X_{1j}，X_{2j},...，X_{nj})に対して、各セルの座標成分をセルk(X_{1k}，X_{2k},...，X_{nk})とすると、

$X_{1j} \leqq X_{1k}$、$X_{2j} \leqq X_{2k}$、…、$X_{nj} \leqq X_{nk}$

のセルk、すなわちセルjの座標成分に対して、座標成分がそれぞれ大きいか、又は、等しいセルkの集合を許容項と言う。また、許容項は、連続キューブとなる。

	X_2	0				1				2				3			
	X_3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
X_1	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
	3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

セル3の許容項 ${}^3X_3{}^3$　　セル36の許容項 ${}^2X_1{}^3\,{}^1X_2{}^3$

図４．２　セルに対する許容項

４．３　部分マップを用いた許容項の生成

真理値k(k=０，１,...，p－１)が局部的に集中している関数(図４．３のマップ)の場合、trueの許容項を生成すると、多くのfalseのセルを含む許容項となることがある。そうなるとその許容項からfalseのセルを取り除くため、より多くのfalseの許容項を生成することになる。それは、すなわち初期回路のゲート数を増加させることを意味する。

そこで、真理値k毎にtrueのセルがすべて存在する最小のマップ部分を生成し、その部分マップにMA法を適用すると、生成するtrueの許容項の中に含まれるfalseのセルの数を少なくできる。

許容項は、部分マップ内の各セルのみに対して、セル毎に生成する。また許容項は、連続キューブとなる。

ここで、例をあげて説明する。図４．３に局部的に真理値k=３が集中する４値３変数関数Fのk=３に着目したマップを示す。k＜３(false)は、この図では空白としている。)

	X_2	0				1				2				3			
	X_3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
X_1	0						3		3								
	1					3	3	3	3								
	2						3		3								
	3					3	3	3	3								

図４．３　真理値k=３の４値３変数関数Fのマップ

このマップから、k=３のセルがすべて存在する最小のマップを生成すると、部分マップ ${}^1X_2{}^1$=(1111)(0100)(1111)を得る。この部分マップを図４．４に示す。

	X_2	1			
	X_3	0	1	2	3
X_1	0		3		3
	1	3	3	3	3
	2		3		3
	3	3	3	3	3

図４．４　部分マップ

４．４　多値MA法のアルゴリズム

多値MA法により初期回路を生成するアルゴリズムを説明する。

p値n変数関数FにおいてX_1，X_2,...，X_nは、p値論理変数でL={０，１，２,...，p－１}の値をとる。真理値k(k=１，２,...，p－１)毎にMA法を適用し、生成される許容項をP_{j-k}(j=０，１，２,...，q)とする。

ステップ(6)で用いるディスジョイント・シ

ョープ(⊕)演算は、trueの許容項からfalseの許容項を取り除くための演算である。

ここで、真理値kのセルをtrueのセル、真理値kより小さいセルをfalseのセル、そして、真理値kより大きいセルをドント・ケアdのセルとみなして、真理値$k=p-1, p-2,..., 1$の順に以下を行う。

（アルゴリズム１）

⑴ 真理値kに対して、部分マップP_{M-k}を生成する。部分マップが生成できない場合、すなわちp値n変数関数Fにおいて、真理値kが存在しない場合は終了する。

⑵ 部分マップP_{M-k}に対して、ドント・ケアdを利用して部分マップの拡大を行う。拡大後の部分マップを新たにP_{M-k}とする。また、拡大できない場合は、そのまま部分マップP_{M-k}を用いる。

⑶ 部分マップP_{M-k}をfalseの許容項P_{0-k}とする。

⑷ $i=0$、$j=0$として、（アルゴリズム２）に移る。

許容項P_{i-k}（$i=0, 1, 2,..., s$)から、矛盾するセルを取り除くために生成する許容項をP_{j-k}とする。

関数fは、tを項として$f=t_1+t_2+\cdots+t_m$とする。

（アルゴリズム２）

⑴ $j=j+1$とする。

⑵ 関数fをfalseの許容項P_{i-k}とする。

⑶ 関数fのマップ内に存在し、真理値がtrueでセル番号の最も小さいセルでtrueの許容項P_{j-k}を生成する。関数fのマップ内に真理値trueのセルが存在しない時、$i=i+1$としてステップ⑵に戻る。

⑷ 許容項P_{j-k}に対して、ドント・ケアdを利用して許容項の拡大を行う。拡大後の許容項を新たにP_{j-k}とする。また、拡大できない場合は、そのまま許容項P_{j-k}を採用する。

⑸ 生成済みの許容項の中で、許容項P_{j-k}と同一のものがある場合１つにまとめる(簡単化１)。$i=i+1$としてステップ⑵に戻る。

⑹ f⊕P_{j-k}の演算を行い、その結果を新たに関数fとする。関数$f\neq\phi$の時、ステップ⑴に戻る。また、関数$f=\phi$の時、$i=i+1$としてステップ⑴に戻る。

（アルゴリズム２）において関数fとする許容項P_{i-k}が、tureの許容項となる場合は、（アルゴリズム２）の“true”を“false”と置き換えて行う。すなわち、trueの許容項からfalseのセルを取り除くためにfalseの許容項を生成し、falseの許容項からtrueのセルを取り除く(復活させる)ために、trueの許容項を生成する。

（アルゴリズム２）は、$i=j$となるとき処理を終了し、次の真理値$k=k-1$として(アルゴリズム１)に戻る。また、k毎に部分マップを生成し、許容項P_{0-k}としているが、最後に許容項P_{0-k}=(1111)(1111)(1111)とする。

5　初期回路の簡単化

多値ＭＡ法で生成される初期回路(多段NANDゲート回路)は、非常に冗長な回路となる。そこで、初期回路の簡単化をする。

初期回路の簡単化として、(i)ゲート数の削減、(ii)ゲートへの入力線数の削減を行う。ゲート数の削減として、簡単化１、３、及び４、入力線数の削減として簡単化２を行う。

簡単化１は、多値ＭＡ法の中で行われる。簡単化２、３、４の行う順位によって簡単化後の回路は異なる。本方法では、入力線数の削減を優先的に行うので、簡単化１、２、３、４の順に行う。

ここで、ある許容項P_{p-k}（真理値$k=1, 2,..., p-1$)から矛盾するセルを取り除くために生成される許容項P_{c-k}をP_{p-k}の子許容項と呼び、また、P_{p-k}をP_{c-k}の親許容項と呼ぶ。

同様に、許容項の木をゲートに置換した回路でも、あるゲートP_{c-k}の出力がゲートP_{p-k}の入力とするとき、P_{p-k}をP_{c-k}の親ゲートと

呼び、P_{c-k}をP_{p-k}の子ゲートと呼ぶ。

5.1　同一許容項の削減(簡単化1)

簡単化1は、真理値k(k=1，2,...，p−1)毎にMA法を用いて、許容項を生成する際に行われる。MA法により許容項を生成していくときに、生成済みの許容項と同一のものが生成された場合、1つにまとめる。

生成される1つの許容項は、回路中の1つのゲートに対応するので、許容項の数を減らすことによって、初期回路のゲート数が削減される。

簡単化1では、真理値k(k=1，2,...，p−1)で生成される許容項のみで、許容項を1つにまとめるものとする。

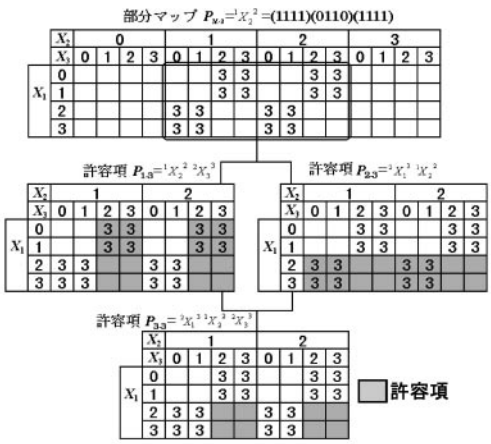

図5.1　許容項の木

例えば、図5.1のようにtrueの許容項P_{1-3}からfalseのセルを取り除くため、falseの許容項P_{3-3}を生成する。同様に許容項P_{2-3}からfalseの許容項P_{4-3}を生成するが、生成済みの許容項P_{3-3}と同じ許容項なのでこれらを1つにまとめる。

5.2　入力線の削減(簡単化2)

あるゲートP_{c-k}のすべての親ゲートP_{p-k}に共通に含まれる入力変数と、P_{c-k}の入力変数で同一の変数をP_{c-k}の入力変数から削除する。

また、最大の段数のゲートから段数順に入力線の削減を行う。段数は、出力ゲートを1段目として、出力ゲートの子ゲートを2段目、その子ゲートを3段目、…、i段目とする。

例として、ある回路の6から8段目(8段目が最大の段数)を図5.2に示す。

段数が最大のゲートから、入力線を削減するので、まず8段目のゲートP_{9-3}から始める。ゲートP_{9-3}の親ゲートは、ゲートP_{8-3}である。同一の入力変数を調べると、${}^2X_1{}^3$と${}^1X_2{}^1$である。従って、ゲートP_{9-3}から入力変数${}^2X_1{}^3$と${}^1X_2{}^1$を削除する。

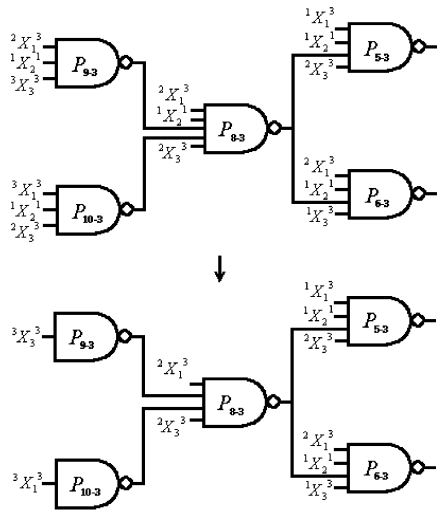

図5.2　簡単化2の削減例

5.3　同一ゲートの削減(簡単化3)

多値MA法により初期回路を生成する段階で、真理値k毎での同一許容項、つまり、同一ゲートをまとめる(簡単化1)。

簡単化3では、さらに初期回路生成後に真理値k(k=1，2,...，p−1)の異なる回路全体での同一ゲートを1つにまとめる。同一ゲートとは、そのゲートへの入力変数と、その子ゲートがすべて同じであるようなゲート同士をいう。しかし、同一ゲートであっても、段数が奇数段目のゲートと偶数段目のゲート同士は、ゲートをまとめないものとする。な

ぜなら、段数が偶数段目のゲートは、trueの許容項からなり、奇数段目のゲートは、falseの許容項であるため、まとめることができないからである。

例えば、図５．３の奇数段目のゲートP_{2-3}とP_{2-2}は、入力変数がすべて同じで、子ゲートは共に持たないゲートなので同一ゲートとして１つにまとめる。

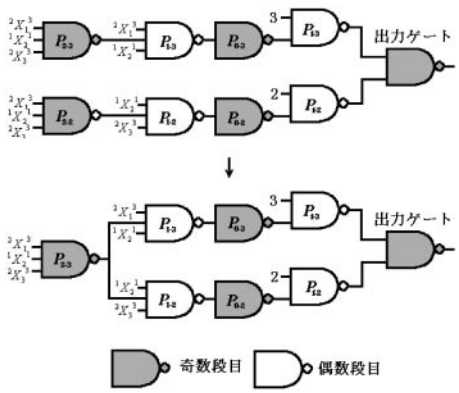

図５．３　簡単化３の削減例

５．４　積形式を利用したゲートの削減（簡単化４）

多値MA法で生成される初期回路は、一出力多段NANDゲート回路であり、その特徴としてド・モルガンの定理から奇数段目をORゲート、偶数段目をANDゲート、そして、奇数段目の末端ゲートをNOTゲートに置き換えることができる。p値n変数関数Fの初期回路において奇数段目の末端ゲートP_{e-k}（入力変数のみを持つゲート）の入力変数(X_1、X_2、…、X_n)の積の否定を項$\overline{X_{1e}\cdot X_{2e}\cdot\cdots\cdot X_{ne}}$とする。その親ゲート$P_{p-k}$は偶数段目となり、入力変数の積を項$X_{1p}\cdot X_{2p}\cdot\cdots\cdot X_{np}$とすると、その２つの項の積

$X_{1p}\cdot X_{2p}\cdot\cdots\cdot X_{np}(\overline{X_{1e}\cdot X_{2e}\cdot\cdots\cdot X_{ne}})$を演算した結果が１つの積項の場合のみ、変数をそれぞれ親ゲートP_{p-k}の入力変数とする。そして、ゲートP_{e-k}を削除する。それぞれの項は、ゲートP_{e-k}の場合、P_{e-k}に対応する許容項の否定を示し、ゲートP_{p-k}は、P_{p-k}に対応する許容項である。また、親ゲートが複数ある場合は、すべての親ゲートと置き換えなければならない。

例として、図５．３の簡単化３を行った初期回路について簡単化４を適用する。

奇数段目の末端ゲートを検索するとゲートP_{2-3}が存在する。ゲートP_{2-3}に対応する許容項P_{2-3}'$={}^2X_1^3\,{}^1X_2^1\,{}^2X_3^3$=(0011)(0100)(0011)である。ゲートP_{2-3}の親ゲートは、ゲートP_{1-3}とP_{1-2}である。各ゲートに対応する許容項は、許容項P_{1-3}'$={}^2X_1^3\,{}^1X_2^1$=(0011)(0100)(1111)とP_{1-2}'$={}^1X_2^1\,{}^2X_3^3$=(1111)(0100)(0011)である。

許容項P_{2-3}'の否定と許容項P_{1-3}'の積をとると、

${}^2X_1^3\,{}^1X_2^1(\overline{{}^2X_1^3\,{}^1X_2^1\,{}^2X_3^3})={}^2X_1^3\,{}^1X_2^1\,{}^0X_3^1$

となる。

また、許容項P_{2-3}'の否定と許容項P_{1-2}'の積をとると

${}^1X_2^1\,{}^2X_3^3(\overline{{}^2X_1^3\,{}^1X_2^1\,{}^2X_3^3})={}^0X_1^1\,{}^1X_2^1\,{}^2X_3^3$

となる。

結果としてどちらも、１つの項となるのでそれぞれの親ゲートの入力変数と置き換える。そして、ゲートP_{2-3}を削除する。

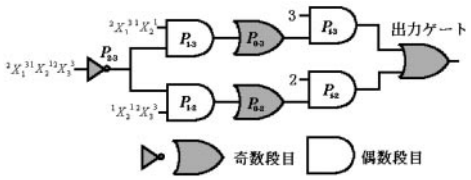

図５．４　置換されたAND-OR-NOT回路

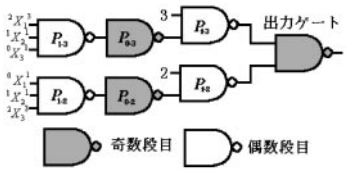

図５．５　簡単化４を適用後の初期回路

6 結果

多値論理に関する一設計法として、これまでに説明してきた多値論理系に拡張したMA法（多値MA法)、及び、４つの簡単化のアルゴリズムを実際にＣ言語でプログラム化し、その実行した結果を述べる。

6.1 開発環境

プログラムの開発、及び実行に使用したコンピュータのスペックを以下に示す。

（CPU） Intel PentiumII 300MHz
（メモリ） 128MB
（OS） FreeBSD 3.2－RELEASE
（コンパイラ） gcc
（プログラミング言語） Ｃ言語

6.2 実行結果

４値n変数関数(n=２～７)での本方法の簡単化の削減率、初期回路の回路規模の増加程度、及び、計算時間を示す。２値n変数関数（n=４，６，８，10，12，及び14)と真理値k=１のみの４値n変数関数(n=２，３，４，５，６，及び７)において初期回路の回路規模の比較を行った結果を述べる。

ここで、p値n変数関数をマップで表した場合のセルの総数は、p^n個である。これらのセルの中で、真理値０，１，２,...，$p-1$の値をとるセルの数の各割合が、$c_0+c_1+c_2+\ldots+c_{p-1}=1$となるとき、この関数の真理表濃度を$c_0+c_1+c_2,\ldots,\ c_{p-1}$という。

6.2.1 4値n変数関数について

４値n変数関数において、２から７変数までの初期回路の入力線数(I)、ゲート数(G)、及び段数(S)の増加をみる。真理値表濃度は、各真理値０，１，２，３に対して$c_0=c_1=c_2=c_3=0.25$とする。同時に本方法で行う簡単化２、３、及び４でのゲート数(G)、入力線数(I)、及び段数(S)も表に示す。表は、先に述べた真理値表濃度になるようにランダムに発生させた関数について、10回の実行結果の平均値である。

表６．１ 多値MA法(簡単化１）

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	48.4	19.2	7.2
3	312.0	82.2	9.2
4	2088.6	380.2	11.2
5	12374.6	1708.4	11.2
6	70416.2	7535.2	11.4
7	376086.5	31882.2	12.5

表６．２ 本方法(簡単化１、２）

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	41.8	19.2	7.2
3	239.2	82.2	9.2
4	1599.0	380.2	11.2
5	9572.6	1708.4	11.2
6	55402.0	7535.2	11.4
7	303617.2	31882.2	12.5

表６．３ 本方法(簡単化１、２、３）

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	41.0	18.4	7.0
3	233.4	76.8	9.0
4	1583.8	369.8	11.0
5	9493.8	1654.4	11.0
6	55266.6	7461.8	11.4
7	303554.0	31862.5	12.5

表６．４ 本方法(簡単化１、２、３、４）

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	38.2	16.8	6.4
3	219.4	71.2	8.4
4	1559.4	361.8	10.8
5	9424.4	1638.0	11.0
6	55215.4	7451.0	11.4
7	303532.2	31854.0	12.5

表６．１から、入力線数(I)、ゲート数(G)、そして、段数(S)が共に変数nが大きくなるに従い増加している。

また、表６．２から表６．４を見ると、簡単化２、３、及び４で入力線数(I)、ゲート数(G)、

そして段数(S)がそれぞれ削減されていることが分かる。簡単化２は、入力線の削減のみを行い、ゲートの削減は行わない。簡単化３、４は、ゲートの削減を行い、それにともない入力線も若干削減している。

６.２.２　削減効率

６.２.１節の表６.２、６.３、６.４から入力線数(I)、ゲート数(G)、および段数(S)の削減効率を求める。多値MA法(簡単化１)のとき（表６.１)の、I、G、Sの値を基準として、簡単化(１、２)、簡単化(１、２、３)、および簡単化(１、２、３、４)の各場合における値を残存率としてパーセンテージで表し、表６.５、６.６、６.７に示す。

表６.５　(簡単化１、２)での残存率［%］

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	86.4	100.0	100.0
3	76.7	100.0	100.0
4	76.6	100.0	100.0
5	77.4	100.0	100.0
6	78.7	100.0	100.0
7	80.7	100.0	100.0

表６.６　(簡単化１、２、３)での残存率［%］

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	84.7	95.8	97.2
3	74.8	93.4	97.8
4	73.7	97.3	98.2
5	76.7	96.8	98.2
6	78.5	99.0	100.0
7	80.7	99.9	100.0

表６.７　(簡単化１、２、３、４)での残存率［%］

変数の数(n)	入力線数(I)	ゲート数(G)	段数(S)
2	78.9	87.5	88.9
3	70.2	86.6	91.3
4	74.7	95.2	96.4
5	76.2	95.9	98.2
6	78.4	98.9	100.0
7	80.7	99.9	100.0

表６.５から表６.７を見ると、それぞれの割合にはばらつきがあるものの、本方法では、特に初期回路から入力線を20%ほど削減できている。また、ゲート数、段数は、平均して５%程度の削減にとどまり、変数が大きくなるほど削減率が悪くなっている。ゲートの削減個数は変数が大きくなるにしたがい増加するが、初期回路全体のゲート数が急激に大きくなるため、削減個数の割合は小さくなる。しかし、本方法の簡単化２、３、４によって、初期回路をより縮小することができた。

６.２.３　２値論理関数と４値論理関数の比較

多値論理回路は、２値論理回路に比べ回路の入力線数(I)を少なくできると言われている。ここでは、２値n_2変数関数(n_2=４，６，８，10，12，及び14)と真理値k=１のみの４値n_4変数関数(n_4=２，３，４，５，６，及び７)において、$2^{n_2}=4^{n_4}$のときの関数同士の入力線数(I)、ゲート数(G)、そして段数(S)を比較する。すなわち、関数をマップで表した場合のセルの総数が同じ関数同士を比べる。また、２値n_2変数関数、４値n_4変数関数双方の関数は、セル番号が同じセルの持つ真理値を同一にする。例えば、ある２値４変数関数のセル12=(１，１，０，０)が真理値１であるとき、セルの総数が同じである４値２変数関数のセル12=(３，０)を真理値１とするような関数同士を比較する。表は、真理値表濃度$c_0=c_1$=0.5として、先に述べた真理値表になるようにランダムに発生させた関数について、10回の実行結果の平均値である。

表６.８　２値n_2変数関数

変数の数(n_2)	入力線数(I)	ゲート数(G)	段数(S)
4	24.6	9.4	6.6
6	152.2	41.0	8.0
8	911.8	189.2	9.2
10	5654.8	859.6	9.8
12	31791.0	3796.8	10.2
14	165159.4	15785.0	11.4

表６．９　４値n_4変数関数

変数の数(n_4)	入力線数(I)	ゲート数(G)	段数(S)
2	20.6	8.4	6.6
3	127.4	37.2	8.6
4	820.8	185.2	10.4
5	4880.2	838.6	12.2
6	24858.0	3705.4	12.2
7	135966.8	15493.4	13.0

表６．８の２値n_2変数関数と、表６．９の４値n_4変数関数を比較すると、本方法でも２値論理に比べ４値論理の方が入力線数(I)の小さい初期回路となっている。これは、多値MA法で生成される許容項が、２値論理より４値論理の方が少ない変数の数の積項となるためである。すなわち許容項は、２値４変数関数の場合、最大で４変数の積項で表されるのに対し、４値２変数では、最大で２変数の積項となるからである。この許容項をゲートに置換すると２値４変数の場合は、１つのゲートへの最大の入力変数の数は４入力であり、４値２変数の場合は２入力となることから多値論理関数の方が、入力線数を少なくできると考えられる。また、ゲート数(G)も同様に２値論理に比べ４値論理の方が小さい初期回路となっている。よって、多値論理によって回路を設計する方が、規模の小さい回路を設計できると考えられる。

7　おわりに

本論文では、否定演算の定義された多値論理系へMA法を拡張する手法を述べた。すなわち、MA法は本研究室で発表してきた２値論理に関する一出力多段NANDゲート回路の一設計法であり、これを多値論理に適用するためにSuらの考案した多値論理系に拡張することができた。さらに、多値MA法で生成される初期回路を簡単化する手法も含めた本方法をプログラム化し、種々の多値論理関数を実行した。本研究では４つの簡単化を初期回路に対して、簡単化１、２、３、そして４の順に行った結果も示した。簡単化１を含む多値MA法で生成される初期回路を100%とすると、その削減率は、入力線数(I)が77%、ゲート数(G)が94%、そして、段数(S)が96%の規模まで初期回路を縮小することができた。

多段NANDゲート回路も２値論理回路と同様に３段化による初期回路の簡単化が考えられる。しかし、３段化にすると出力ゲートなどの特定のゲートに入力線が集中し、多段NANDゲート回路の方が有効な回路と言える。

多値論理は、２値論理に比べ入力線数(I)のより少ない回路を設計できると言われている。本方法でも２値論理と４値論理をある条件のもとで初期回路を生成し入力線数(I)を比較した。その結果、２値論理よりも４値論理の方が入力線数(I)を少なく回路設計することができた。

今後、更に有効な簡単化の手法の開発が必要である。また、本方法では項表現で関数などを表しプログラム化したが、２値論理では計算機のメモリ容量の縮小、及び、計算時間の短縮のためBDD(２分決定木)が使われるようになった。そこで、多値MA法においても多値決定木によるプログラム化も課題として考える。

引用文献

⑴松田　秀雄，宮腰　隆，畠山　豊正：“多段NANDゲート回路の一設計法”，情報処理学会研究報告設計自動化，DA−67，1993.

⑵S. Suand P. T. Cheung：“Computer minimization of multivalued switching functions”, IEEE Trans. Comput., C-21, 9, pp. 995-1003, Sept, 1972.

参考文献

［１］松田　秀雄，宮腰　隆：“部分マップ法による多値論理関数の主項導出について”，電子通信学会論文誌，Vol.J68-D, No.6, Jun, 1985.

［2］C.M.Allen and D.D.Givone :“ Aminimization technique for multiple-valued logic systems ”, IEEE Trans. Comput., C-17, 2, pp.182-184, Feb, 1968.

［3］Z. G. Vranesic, E. S. LeeandK. C. Ｓmith :“ Amany-valued algebra for switching systems ”, IEEE Trans. Comput., C-19, 10, pp. 964-971, Oct, 1970.

［4］三根　久，長谷川利治，原田　尚文，島田　良作：“ 多線式多値論理回路網の一論理設計法 ”，信学論(D), 56-D, 3, pp. 186-193, May, 1973.

［5］羽根　孝泰：“ ドント・ケアを含む論理回路をNANDゲート回路で実現する一設計法 ”, 修士論文，TOYAMA-University, 1997.

［6］平松　徹也：“ SBDD(共有二分決定グラフ)を用いた多出力NANDゲート回路の設計法 ”, 修士論文，TOYAMA-University, 1997.

［7］荒井　勇，松田　秀雄，中嶋　芳雄，宮腰　隆：“ 多値論理回路の一設計法 ”，平成11年度．電気関係学会北陸支部連合大会

A Method of Designing Multiple-Valued Logic Circuit

Tetsuya Fujita, Isam Arai, Mamoru Takamatsu, Yosio Nakajima

Abstract

In this paper, one method of extending MA method, which is one of designing 2-valued logic circuits using multiple NAND gates, to multiple-valued logic system is shown. The implementation of this method, including optimization of an initial circuit, and the result of simulation for some multiple-valued logic function is also discussed.

As a result, it becomes clear that the multiple-valued logic desinging needs less input line comparing to a 2-valued logic. And, the optimization made an initial circuit more compact, 77% at input lines, 94% at gates, 96% at stages.

Keywords

NAND gates, MA method, Multiple-valued logic system, Simplification, 4-valued logic function