設 計 自 動 化 74－2
（1995．3．8）

多端子EXOR三分決定グラフを用いたAND-EXOR論理式の最小化法について

笹尾 勤，　泉原 史幸
九州工業大学情報工学部·電子情報工学科
〒820 飯塚市大字川津680-4

あらまし: 積項数が最小な固定極性リード-マラー論理式(FPRM)およびクロネッカ論理式(KRO)を求める方法を示す．データ構造として，EXOR三分決定グラフ(ETDD: EXOR ternary decision diagram)と多端子EXOR三分決定グラフ(MTETDD: Multi-terminal EXOR ternary decision diagram)を用いる．計算時間やメモリを削減するため種々の手法を用いることにより，既存のどのプログラムよりも高性能なものが得られた．入力数が94までの論理式の最小化結果を示す．

キーワード: AND-EXOR論理式，リード·マラー論理式，論理式最小化，BDD，三分決定グラフ，論理合成．

Exact Minimization of AND-EXOR Expressions using Multi-terminal EXOR Ternary Decision Diagrams.

Tsutomu SASAO and Fumitaka Izuhara
Department of Computer Science and Electronics
Kyushu Institute of Technology, Iizuka 820, Japan

Abstract: This paper presents methods to derive a Fixed Polarity Reed-Muller expression (FPRM) and a Kronecker expression (KRO) having minimum number of products for a given logic function. The minimization methods use EXOR ternary decision diagrams (ETDDs) and multi-terminal EXOR ternary decision diagrams (MTETDDs) to represent extended truth vectors and (extended) weight vectors, and outperform existing methods. Various techniques to reduce computation time and memory storage are incorporated. Experimental results upto 94 inputs are shown.

Key words: EXOR, Reed-Muller expression, logic minimization, FPRM, KRO, multi-terminal decision diagram, TDD, BDD.

1 まえがき

AND-EXOR 二段論理式としては, 種々のクラスが知られている. このうち, ESOP は図 1.1の PLA で実現可能である. 各変数に対して x_i と $\bar{x}_i$ の両方のリテラルが必要であるが, 積項数は最も少ないという特長がある [20]. FPRM では, 各変数に対して x_i と $\bar{x}_i$ のいずれか一方のリテラルしか必要とせず, 図 1.2の PLA で実現可能である. また, FPRM では, 故障検査容易 [15, 8] という特長もある. 論理関数を PLA で実現する場合, ESOP では, AND アレイのリテラル線は $2n$ 本必要であるが, FPRM では n 本で十分である. 従って, 論理関数 f を表現する ESOP の積項数を $\tau(ESOP:f)$,FPRM の積項数を $\tau(FPRM:f)$ としたとき, $2\tau(ESOP:f) > \tau(FPRM:f)$ の場合には, FPRM のアレイ面積は ESOP の面積より小さくなる.

FPRM の最小化は, 2^n 個の展開中から, 積項数が最小の論理式を求めればよい. 最小化の代表的手法として, 2 つの方法が知られている [6, 10].

1) グレイコード法
一つの FPRM から他の FPRM を順次生成し, 2^n 個の全ての FPRM を生成する. 記憶領域は $O(2^n)$ あれば十分であるが, 計算時間は 4^n に比例する [1, 9].

2) 拡張真理ベクトルと重みベクトルを用いる方法
重みベクトルを用いて, 2^n 個の FPRM の積項数を同時に導く. 記憶容量と計算時間は 3^n に比例する [3, 6, 11].

このうち高速なのは, 2) の方法であり, $n = 15$ 程度ならば, 通常のワークステーションで最適解が求まる. ただし, 記憶容量が 3^n に比例するため, n が大きい場合には, 使用不可能になる. 本稿では, 拡張真理ベクトルを表現するために EXOR 三分決定グラフ (ETDD: Ternary Decision Diagram) を, 重みベクトルを表現するために, 多端子 EXOR 三分決定グラフ (MTETDD: Multi-terminal EXOR Ternary Decision Diagram) を用いる. このデータ構造を用いることにより, n が 16 以上の関数に対して最適 FPRM を求めることができた. また, 同様な手法で, 最適 KRO(Kronecker expression: クロネッカ論理式) も求まった. $n = 96$ までの実験結果を示す.

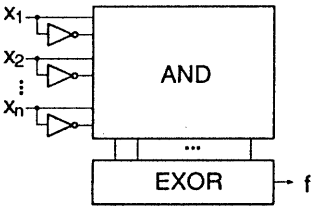

図 1.1: ESOP を実現する AND-EXOR PLA

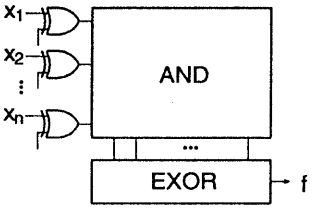

図 1.2: FPRM を実現する AND-EXOR PLA

2 定義および基本的性質

ここでは, AND-EXOR 形論理式の種々のクラスの定義を行い, それらの関係を明確にする [20].

定理 2.1 (展開定理) 任意の論理関数 f は

$$f = f_0 \oplus x f_2 \tag{2.1}$$
$$f = \bar{x} f_2 \oplus f_1 \tag{2.2}$$
$$f = \bar{x} f_0 \oplus x f_1 \tag{2.3}$$

の形で展開可能である. ただし, $f_2 = f_0 \oplus f_1$ である. (2.1) を**正極性ダビオ展開**, (2.2) を**負極性ダビオ展開**, (2.3) を**シャノン展開**という.

定義 2.1 与えられた関数 f を正極性ダビオ展開を用いて展開すると, 肯定リテラルのみの論理式

$$a_0 \oplus a_1 x_1 \oplus \cdots \oplus a_n x_n \oplus a_{12} x_1 x_2 \oplus a_{13} x_1 x_3 \oplus \cdots \oplus a_{nn-1} x_n x_{n-1} \oplus \cdots \oplus a_{12\cdots n} x_1 x_2 \cdots x_n \tag{2.4}$$

が得られる. これを**正極性 Reed-Muller 論理式** (PPRM: positive polarity Reed-Muller expression) という.

PPRM は関数に対して一意的に定まるので標準展開であり, 最小化問題も存在しない.

例 2.1 $f = \bar{x}_1\bar{x}_2\bar{x}_3$ を PPRM で表現してみよう. $\bar{x}_1 = x_1 \oplus 1$, $\bar{x}_2 = x_2 \oplus 1$, $\bar{x}_3 = x_3 \oplus 1$ を代入すると,

$$\begin{aligned} f &= (x_1 \oplus 1) \cdot (x_2 \oplus 1) \cdot (x_3 \oplus 1) \\ &= 1 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_1x_2 \oplus x_2x_3 \oplus x_1x_3 \oplus x_1x_2x_3 \end{aligned}$$

となり, fは, すべて肯定リテラルのみの論理式で表現できる.

一般に, $\bar{x}_1\bar{x}_2\cdots\bar{x}_n$ を PPRM で表現するには, 2^n個の項が必要である.

定義 2.2 与えられた関数 fの各変数に対して正極性ダビオ展開または負極性ダビオ展開を適用すると, (2.4) と同様の形をした論理式が得られる. ただし, 各変数に対して, いずれか一方の展開法のみを使用する. この論理式を**固定極性 Reed-Muller 論理式** (FPRM: fixed polarity Reed-Muller expression) という.

n 変数関数に対して FPRM は 2^n個存在する. 最小化問題は 2^n個の FPRM 中で積項数が最小の式を求める問題となる.

例 2.2 $f = x_1x_2x_3x_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$ を FPRM で表現してみよう. 二つの積項は同時に 1 とならないので, $f = x_1x_2x_3x_4 \oplus \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$と表現できる. x_1と x_2に正極性ダビオ展開を, x_3と x_4に負極性ダビオ展開を適用すると, x_1と x_2は肯定リテラルのみ, x_3と x_4は否定リテラルのみの式となる. $\bar{x}_1 = x_1 \oplus 1$, $\bar{x}_2 = x_2 \oplus 1$, $x_3 = \bar{x}_3 \oplus 1$, $x_4 = \bar{x}_4 \oplus 1$ を fに代入すると,

$$\begin{aligned} f &= x_1x_2(\bar{x}_3 \oplus 1)(\bar{x}_4 \oplus 1) \oplus (x_1 \oplus 1)(x_2 \oplus 1)\bar{x}_3\bar{x}_4 \\ &= x_1x_2(1 \oplus \bar{x}_3 \oplus \bar{x}_4 \oplus \bar{x}_3\bar{x}_4) \oplus (1 \oplus x_1 \oplus x_2 \oplus x_1x_2)\bar{x}_3\bar{x}_4 \\ &= x_1x_2 \oplus x_1x_2\bar{x}_3 \oplus x_1x_2\bar{x}_4 \oplus \bar{x}_3\bar{x}_4 \oplus x_1\bar{x}_3\bar{x}_4 \oplus x_2\bar{x}_3\bar{x}_4 \end{aligned}$$

となる. これは, FPRM である.

一般に, $x_1x_2\cdots x_n \vee \bar{x}_1\bar{x}_2\cdots\bar{x}_n (n = 2r)$ を FPRM で表現すると, $2^{r+1} - 2$ 個の項が必要である.

定義 2.3 与えられた関数 fの各変数に対して, 正極性ダビオ展開, 負極性ダビオ展開, シャノン展開のいずれか一つを用いて展開すると, FPRM を一般化した論理式が生成できる. この論理式を **Kronecker 論理式** (KRO: Kronecker expressioin) と呼ぶ.

n 変数関数に対して KRO は 3^n個存在する.

例 2.3 $f = x_1x_2x_3x_4 \vee \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$ を KRO で表現してみよう. すべての変数にシャノン展開を適用すると, $f = x_1x_2x_3x_4 \oplus \bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4$と表現できる.

定義 2.4 任意の積項を EXOR で結合した論理式を AND-EXOR 論理式 (ESOP: EXOR sum-of-products expression) という.

ESOP は最も一般的な論理式であり, 積項数は最も少なくてよい. 能率の良い最小化方法は知られておらず, 繰り返し改善法が用いられている [21].

定理 2.2 $\mathcal{FPRM}$, $\mathcal{KRO}$, $\mathcal{ESOP}$をそれぞれ対応する論理式の集合とするとき, 次の関係が成立する.
$\mathcal{FPRM} \subset \mathcal{KRO} \subset \mathcal{ESOP}$

3 FPRM の最適化法

本章では, EXOR 三分決定グラフ (ETDD)[20] を導入する. ETDD は FPRM の最適化に必要な情報をすべて含む. ETDD の形式的定義を述べる前に, 簡単な例を示そう.

例 3.1 図 3.1は二変数関数 $f(x_1, x_2)$ の完全 ETDD を示す. ここで

$$\begin{aligned} &f_0 = f(0, x_2), \quad f_1 = f(1, x_2), \quad f_2 = f_0 \oplus f_1, \\ &f_{00} = f(0, 0), \quad f_{01} = f(0, 1), \quad f_{02} = f_{00} \oplus f_{01}, \\ &f_{10} = f(1, 0), \quad f_{11} = f(1, 1), \quad f_{12} = f_{10} \oplus f_{11}, \\ &f_{20} = f_{00} \oplus f_{10}, f_{21} = f_{01} \oplus f_{11}, f_{22} = f_{20} \oplus f_{21}. \end{aligned}$$

(例終)

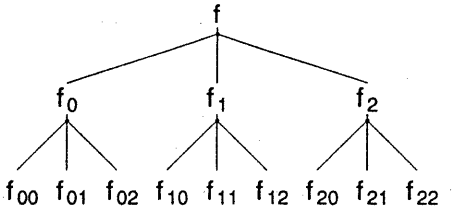

図 3.1: 二変数の ETDD

定義 3.1 EXOR 三分決定木 (ETDT: EXOR Ternary Decision Tree) とは, 節点集合 Vと以下の二つのタイ

プの節点を持つ木である:非終端節点 vは, 属性としてインデックス $index(v) \in \{1,2,\ldots,n\}$ と, 三つの子, $low(v), high(v), EXOR(v) \in V$をもつ. 終端節点 vは, 属性として, 節点の値 $value \in \{0,1\}$ を持つ.

1. vが終端節点の時:
   (a) $value(v) = 1$ ならば, $f_v = 1$.
   (b) $value(v) = 0$ ならば, $f_v = 0$.
2. vが非終端節点で $index(v) = i$ ならば, f_vは関数 $f_v(x_1,x_2,\ldots,x_n) = \bar{x}_i f_{low(v)}(x_1,x_2,\ldots,x_n) \oplus x_i f_{high(v)}(x_1,x_2,\ldots,x_n)$, で $f_{EXOR(v)}(x_1,x_2,\ldots,x_n) = f_{low(v)}(x_1,x_2,\ldots,x_n) \oplus f_{high(v)}(x_1,x_2,\ldots,x_n)$. である.

ETDT において, どの非終端節点も異なる三つの子を持ち, 根から終端節点へのどの経路も n 個の変数を含むとき, その ETDT は**完全** (complete) であるという. 与えられた関数を完全 ETDT で表現したとき, 3^n個の終端節点は, 拡張真理ベクトルを形成する.

定義 3.2 n 変数関数の**拡張真理ベクトル** (extended truth vector) とは, 3^n個の要素からなる二値ベクトルであり, 各要素は, 完全 ETDT の終端節点の値をに対応する.

例 3.2 図3.1の完全ETDT において, $f(x_1,x_2)$ の真理ベクトルは $[f_{00}, f_{01}, f_{10}, f_{11}]$ である. また, 9 つの要素からなる二値ベクトル $[f_{00}, f_{01}, f_{02}, f_{10}, f_{11}, f_{12}, f_{20}, f_{21}, f_{22}]$ は, $f(x_1,x_2)$ の拡張真理ベクトルである.

完全 ETDT の 3^n個の終端節点が拡張真理ベクトル $[f(\alpha_0), f(\alpha_1), \ldots, f(\alpha_{3n-1})]$ を形成する. ここで, 各要素は, 3 値ベクトル$\alpha = (\alpha_1,\alpha_2,\ldots\alpha_n), \alpha_i \in 0,1,2, (i = 1,2,\ldots,n)$ をインデックスとした時, 昇順に並んでいる. 次に, 拡張真理ベクトルの計算法について示そう. 一般的計算法を示す前に, 3 変数の場合について考えてみよう.

例 3.3 真理ベクトルが[0,1,0,0, 0,1,1,1] である 3 変数関数の拡張真理ベクトルは, 図 3.2のように計算できる. ラベル 1 の列は, 3 値ベクトル$\alpha = (\alpha_1,\alpha_2,\alpha_3)$ を示す. ラベル 2 の列は, 真理ベクトルを表す. 要素が存在するのは, インデックス $\boldsymbol{a} = (a_1,a_2,a_3)$ が 2 値の時, 即ち $a_i \in \{0,1\}$ のときのみである. ラベル 3 の列では, インデックスが$(a_1,a_2,2), a_i \in \{0,1\}$ に対する要素の値を計算する. これらの値は $f(a_1,a_2,0) \oplus f(a_1,a_2,1)$ として計算できる. ラベル 4 の列では, インデックスが$(a_1,2,\alpha_3)$ に対する要素の値を計算する, ここで, $a_i \in \{0,1\}$ で$\alpha_3 \in \{0,1,2\}$ である. これら値は $f(a_1,0,\alpha_3)$ と $f(a_1,1,\alpha_3)$ のEXORで計算する. ラベル 5 の列では, インデックスが $(2,\alpha_2,\alpha_3)$ に対する要素の値を計算する, ここで$\alpha_i \in \{0,1,2\}$ である. これら値は $f(0,\alpha_2,\alpha_3) \oplus f(1,\alpha_2,\alpha_3)$ として計算できる. (例終)

n 変数論理関数の拡張真理ベクトルは次のようにして再帰的に計算できる [19].

アルゴリズム 3.1 (拡張真理ベクトル)

$i = 1,2,\ldots,n$ に対して$\alpha_i \in \{0,1\}$ が成立するとき, $f(\alpha)$ は真理ベクトルから直接得られる. ある値 i に対して$\alpha_i = 2$ ならば,

$$f(\alpha_1,\ldots,\overset{i}{2},\ldots,\alpha_n) = f(\alpha_1,\ldots,\overset{i}{0},\ldots,\alpha_n) \oplus f(\alpha_1,\ldots,\overset{i}{1},\ldots,\alpha_n).$$

と再帰的に計算できる.

定義 3.3 n 変数 FPRM の**極性ベクトル** (polarity vector) を $\boldsymbol{a} = (a_1,a_2,\ldots,a_n)$, $a_i \in \{0,1\}$ とする. ここで, $a_i = 0$ は, x_iに関して正極性ダビオ展開を用いることを示し, $a_i = 1$ は x_iに関して負極性ダビオ展開を用いることを示す. n 変数の FPRM では, 2^n個の異なる極性をとることができ, 極性ベクトル $\boldsymbol{a}$ を定めることによって展開の方法が一意的に定まる.

例 3.4 二変数関数 $f(x_1,x_2)$ の FPRM の極性ベクトルを $\boldsymbol{a} = (0,1)$ とする. このとき $\boldsymbol{a}$ は x_1に関しては正極性ダビオ展開を, x_2に関しては負極性ダビオ展開を使用することを示す. 従って, $\boldsymbol{a} = (0,1)$ のとき, FPRM は $f = 1 \cdot (1 \cdot f_{01} \oplus \bar{x}_2 f_{02}) \oplus x_1(1 \cdot f_{21} \oplus \bar{x}_2 f_{22})$ と表現できる. (例終)

定義 3.4 n 変数関数の**重みベクトル** (weight vector) とは, 2^n項の整数ベクトルで, 各要素は, 極性ベクトル $\boldsymbol{a} = (a_1,a_2,\ldots,a_n)$, $a_i \in 0,1$ で展開した際の積項数を示す.

例 3.5 例 3.3の 3 変数関数の重みベクトルは, 図 3.3に示すように, 拡張真理ベクトルから計算できる. 最初の加算では, x_1を変数とする FPRM の積項数を求める. ラベル 3 の列には, 18 個の FPRM が存在する. 最初の 9 個の要素には, x_1に対して正極性ダビオ展開を用いてお

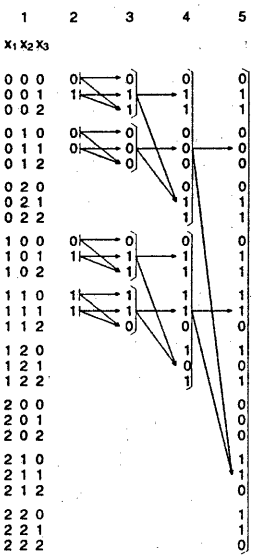

図 3.2: 三変数の拡張真理ベクトル

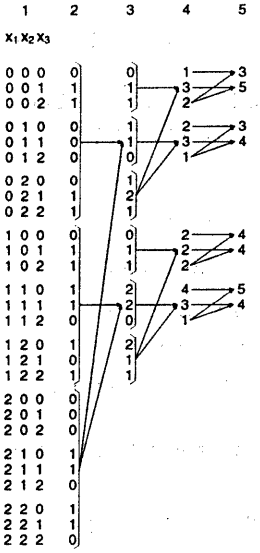

図 3.3: 三変数の拡張重みベクトル

り, 最後の 9 個の要素では, x_1に対して負極性ダビオ展開を用いている. 第 2 番目の加算では, x_1と x_2を変数とする FPRM の積項数を示す. この場合, ラベル 4 の列に示すように, 12 個の FPRM が得られる. この列の最初の 3 つの要素では, x_1と x_2に関して正極性ダビオ展開を用いている. 次の 3 つの要素では, x_1では正極性ダビオ展開を, x_2では負極性ダビオ展開を用いている. ラベル 5 の列は, 全ての可能な FPRM の積項数を示している. この列には 8 個の FPRM があり, この列が与えられた関数の全ての FPRM の積項数を示している. ラベル 5 の列が重みベクトルに対応し, ベクトルの各要素は, 対応する極性ベクトルで展開した際の FPRM の積項数を示す. 図 3.3の最初の行は, 極性ベクトル $\boldsymbol{a} = (0,0,0)$ に対応し, 全ての変数に対して正極性ダビオ展開を用いた FPRM に対応する. この場合, FPRM の積項数は 3 で $f = x_3 \oplus x_1x_2 \oplus x_2x_3$ となる. これは正極性 Reed-Muller 展開である. 図 3.3のラベル 5 の列の最後の要素は, 極性ベクトル $\boldsymbol{a} = (1,1,1)$ に対応する. この場合, 負極性ダビオ展開のみを用いた FPRM に対応する. 積項数は 4 で $f = 1 \oplus \bar{x}_1 \oplus \bar{x}_1\bar{x}_2 \oplus \bar{x}_2\bar{x}_3$ と表現できる. 重みベクトルから次のことが明らかになる:

極性ベクトルが (0,0,0) または (0,1,0) のとき積項数は 3.
極性ベクトルが (0,1,1),(1,0,0),(1,0,1) または (1,1,1) のとき積項数 4.
極性ベクトルが (0,0,1) または (1,1,0) のとき積項数は 5.
(例終り)

n 変数関数の重みベクトルは次のようにして再帰的に計算できる [19].

アルゴリズム 3.2 (重みベクトル)

関数 fを極性ベクトル $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ で展開した際の FPRM の積項数を$\tau(f : \boldsymbol{a})$ とする. n 変数関数 $f = x_nf_0 \oplus x_nf_1$に対して, $n = 0$ のとき, $\tau(0) = 0, \tau(1) = 1$.

$n = 1$ のとき,

$$\tau(f : 0) = \tau(f_0) + \tau(f_2), \tag{3.1}$$

$$\tau(f : 1) = \tau(f_1) + \tau(f_2), \tag{3.2}$$

ここで, f_0と f_1は二値の定数であり, $f_2 = f_0 \oplus f_1$.

$n \geq 2$ のとき

$$\tau(f : a\$0) = \tau(f_0 : a) + \tau(f_2 : a), \tag{3.3}$$

$$\tau(f : a\$1) = \tau(f_1 : a) + \tau(f_2 : a), \tag{3.4}$$

ここで, $f_2 = f_0 \oplus f_1$である. \$は連接を表し, $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ としたとき, $\boldsymbol{a}\$a_{n+1} = (a_1, a_2, \cdots, a_n, a_{n+1})$.
(3.1) と (3.3) は正極性ダビオ展開を用いたことに対応し, (3.2) と (3.4) は負極性ダビオ展開を用いたことに対応する.

最小 FPRM は次のようにして計算できる.

アルゴリズム 3.3 (最小 FPRM)

1) 拡張真理ベクトルを計算する.
2) 重みベクトルを計算する.
3) 重みベクトルの最小値に対応する極性ベクトルを求める.
4) 極性ベクトルに対応する FPRM を求める.

4 最小化プログラムのデータ構造と実現

前章で述べた方法を率直に用いても最小 FPRM は計算できるが, この手法では, 必要なメモリと計算時間が 3^n に比例する. 従って, n が大きいとき使用不可能となる. 本章では, EXOR 三分決定グラフを用いて, 必要なメモリや計算時間を大幅に減らす方法を示す.

4.1 拡張真理ベクトル

n 変数論理関数の拡張真理ベクトルは, 3^n 個の要素をもつ. 完全 ETDT(Complete EXOR ternary decision tree) で表現すると, 節点数は $(3^{n+1}-1)/2$ となる. 完全 ETDT において, 同じ関数を表現する節点や, 冗長な節点を省略すると, EXOR 三分決定グラフ (ETDD) が得られるが, その節点数は, $O(3^n/n)$ となる [20]. 決定グラフの簡単化方法は, 基本的には, BDD と同じである.

4.2 重みベクトル

n 変数論理関数 f の重みベクトルは, 2^n 個の要素をもち, 各要素は, f を極性ベクトル $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ で展開した際の積項数 (整数) を示す. 整数値を取る離散関数を決定グラフを表現するために [5] で開発された手法を用いる. 重みベクトルを完全 MTETDT(Multi-terminal EXOR ternary decision tree) で表現すると, 節点数は $2^{n+1}-1$ となる. MTETDT の場合, 終端節点は, 0, 1 以外の整数にもなり得るため, 率直に MTETDD を構成すると, 通常の BDD に比べ, はるかに多数の節点が必要となる. 論理式最小化プログラムでは, 積項数最小の展開を一つ求めればよく, 最小にならない展開に対しては, 必ずしも正確な積項数 (重み) を求める必要はない. ここでは, 重みがある一定値 (しきい値)T以上になると同じ値 T を代入している. 本手法により, 必要メモリを大幅に削減できる.

4.3 多出力関数の最適化

多出力関数の場合, 関数の表現法を工夫することにより, 必要なメモリ容量が大幅に削減できる.

定義 4.1 n 入力 m 出力関数の拡張真理ベクトルは, n 変数完全 ETDT の各終端節点に対応する 3^n 項ベクトルであり, 各要素は, 対応する m 項二値ベクトルである.

多出力関数の重みベクトルを作成する場合, 次に示す修正拡張真理ベクトルを用いる.

定義 4.2 m 出力関数の拡張真理ベクトルにおいて, 0 ベクトルを 0, それ以外のベクトルを 1 としたベクトルを**修正拡張真理ベクトル** (modified extended truth vector) という.

多出力の拡張真理ベクトルを率直に作成すると大量のメモリを必要するが, 次の手法を用いると, 必要メモリを削減できる.

アルゴリズム 4.1 (多出力最小 FPRM の作成)

1. m 個の出力を表す m 値の変数を根に配置した TDD で拡張真理ベクトルを作成する (つまり, Shared ETDD を用いて多出力関数を表現する).
2. 関数 $f_i (i = 1, 2, \cdots, m)$ の拡張真理ベクトルを $\boldsymbol{EXTRU}(f_i)$ としたとき, 修正拡張真理ベクトルを, $\bigvee_{i=1}^{m} \boldsymbol{EXTRU}(f_i)$ で計算する. ここで, $\boldsymbol{EXTRU}(f_i)$ は要素数が 3^n の二値ベクトルであり, $\bigvee$ はビット毎の OR 演算を示す. ただし, $\boldsymbol{EXTRU}(f_i)$ は ETDD で表現する.
3. 修正拡張真理ベクトルから, 重みベクトルを作成する. ただし, 重みベクトルは, MTETDD で表現する. 整数加算を率直に行うと節点数が増加するため, 加算結果がしきい値 T 以上になれば, 加算結果を T とする.

4. 積項数最小の展開に対する極性ベクトルを求める.
5. 極性ベクトルに対応する FPRM を生成する.

5 クロネッカ論理式の最適化

n 変数の論理関数に対して 3^n 個のクロネッカ論理式が存在する. 最適化の手法は, FPRM とほぼ同じであるが, 重みベクトルの要素数が 3^n になる点が異なる. この重みベクトルを**拡張重みベクトル** (Extended weight vector) 呼ぶ. 従って, 最適化には FPRM の場合よりも多くのメモリが必要となる.

6 実験結果

第3章のアルゴリズムを FORTRAN 言語を用いてプログラムした. 入力変数 n が 14 以下の場合は容易に最適解が求まる. [20] の FPRM や KRO は, 本方法で最適化した. 計算時間や必要メモリは, 入力変数の個数のみに依存する.

$n > 15$ の関数を最小化するために, 4節のアルゴリズムを C 言語でプログラムした. 使用計算機は HP715(メインメモリ 256MB) である. この場合, 与えるしきい値によって, 計算時間が異なる. 最小論理式の積項数よりも, 少しだけ大きいしきい値を入れると, 必要なメモリや計算時間を大幅に削減できる. 本方法を用いると, $n \geq 90$ の多出力関数に対しても, 最適解が求まった. 実験結果を表 6.1に示す. 表 6.1で最も計算時間がかかった関数は FPRM では mish で求めるのに 6 分, KRO では bc0 で求めるのに 30 分必要とした. また ESOP は EXMIN2[21] で SOP(AND-OR 論理式) は MINI2 で簡単化した, ESOP と SOP は準最適解である. 表 6.1より mish や t481 では FPRM と ESOP の積項数は変わらないが, cordic や tial では FPRM の積項数は ESOP よりも 8 倍近く多くなっている事が分かる.

表 6.1: 実験結果

			Products				
Data	IN	OUT	FPRM	KRO	PSDKRO	ESOP	SOP
amd	14	24	156	104	64	58	66
b9	16	5	105	105	81	81	119
bc0	26	11	1117	467	180	168	177
cordic	23	2	6438	1980	1036	776	914
cps	24	109	291	249	154	137	172
duke2	22	29	255	209	102	81	86
ex7	16	5	105	105	81	82	119
gary	15	11	349	242	115	102	107
in2	19	10	355	262	117	108	134
in7	26	10	72	72	36	35	54
intb	15	7	1815		492	327	629
m181	15	9	67	65	31	29	41
misg	56	23	104	104	67	66	69
mish	94	43	53	53	53	53	82
misj	35	14	17	17	16	16	35
misex3	14	14	3536	1421	689	553	690
rckl	32	7	32	32	32	32	32
ryy6	16	1	64	48	40	40	112
t1	21	23	232	209	101	90	103
t481	16	1	13	13	13	13	481
tial	14	8	3683	2438	790	487	579
ts10	22	16	432	128	128	128	128
x6dn	39	5			104	95	81

7 まとめ

FPRM や KRO の最小化法に関しては, 古くから, 種々の方法が提案されてきた [1, 12]. $n \leq 14$ の場合は, 拡張真理ベクトルと (拡張) 重み表を用いた方法が最も高速である. $n \geq 15$ の場合は, BDD や FDD を用いた方法が提案されているが, 入力変数の個数が多い場合は, 最適化が困難で準最適解しか求まらなかった [2, 7, 13, 17, 22].

本論文では, 多端子 EXOR TDD を使用した最小化法を示した. 関数によっては入力数が 90 以上の FPRM も最適化することができた. FPRM は ESOP よりも多数の積項数を必要とするが, 故障検査容易という特長がある. 今後, FPRM の性質を活用した応用を開発していきたい.

謝辞

本研究は一部, 文部省科学研究費補助金による.

参考文献

[1] Ph. W. Besslich, "Efficient computer method for EXOR logic design," *IEE Proc.*, vol. 130, Part E,

pp. 203-206, 1983.

[2] B. Becker and R. Drechsler, "FDD based minimization of fixed polarity Reed-Muller forms using hybrid generic algorithms," *Proc International Conference on Computer Design*, Oct. 1994.

[3] G. Bioul, M. Davio and J. P. Deschamps: "Minimization of ring- sum expansions of Boolean functions," *Philips Res. Rpts.*, vol. 28, pp. 17-36, 1973.

[4] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*, Boston, MA. Kluwer, 1984.

[5] E. M. Clarke, M. Fujita, P. C. McGeer, K. L. McMillan and J. C. Yang, "Multi-terminal binary decision diagrams: An efficient data structure for matrix representation" *Int. Workshop on Logic Synthesis*, May 1993, pp. 6A 1-15.

[6] M. Davio, J.-P. Deschamps, and A. Thayse, *Discrete and Switching Functions*, McGraw-Hill Inter-national, 1978.

[7] R. Drechsler, M. Theobald, and B. Becker, "Fast FDD based minimization of generalized Reed-Muller forms," *Proc European Design Automation Conf.*, 1994.

[8] H. Fujiwara, *Logic Testing and Design for Testability*, The MIT Press, 1985.

[9] D. Green, *Modern Logic Design*, Addison-Wesley Publishing company, 1986.

[10] D. H. Green, "Reed-Muller canonical forms with mixed polarity and their manipulations, " *IEEE Proc.*, Vol. 137, Pt. E. No1. Jan. 1990, pp.103-113.

[11] P. K. Lui and J. Muzio, "Boolean matrix transforms for the parity spectrum and the minimization of modulo-2 canoncial expansions," *IEE Proc.*

[12] A. Mukhopadhyay and G. Schmitz, "Minimization of Exclusive OR and logical Equivalence of switching circuits," *IEEE Trans. Comput.*, C-19, pp.132-140, 1970.

[13] U. Kebschull and W. Rosenstiel, "Efficient graph based- computation and manipulation of functional decision diagrams," *Proc. EDAC'93*, pp. 43-47, 1993.

[14] S. Purwar, "An efficient method of computing generalized Reed-Muller expansions from binary decision diagram," *IEEE Trans. on Comput.*, Vol. 40 No. 11, Nov. 1991, pp. 1298-1301.

[15] S. M. Reddy, "Easily testable realization for logic functions," *IEEE Trans. on Comput.*, C-21, pp. 1083-1088, 1972.

[16] K. K. Saluja and E. H. Ong, "Minimization of Reed-Muller canonic expansion," *IEEE Trans. on Comput.*, C-28, pp. 535-537,1979.

[17] A. Sarabi and M. A. Perkowski, "Fast exact and quasi-minimal minimization of highly testable fixed polarity AND/XOR canonical networks," *Proc. Design Automation Conference 1992*, June 1992, pp. 20-35.

[18] T. Sasao and P. Besslich, "On the complexity of MOD-2 Sum PLA's," *IEEE Trans. on Comput.*, vol. 32, No. 2, pp. 262-266, Feb. 1990.

[19] T. Sasao , "Transformation of multiple-valued input two-valued output functions and its application to simplification of exclusive-or sum-of-products expressions," *Proc. ISMVL-91*, pp. 270-279, May. 1991.

[20] T. Sasao, "AND-EXOR expressions and their optimization," in (Sasao e.d.) *Logic Synthesis and Optimization*, Kluwer Academic Publishers, 1993.

[21] T. Sasao, "EXMIN2: A simplification algorithm for exclusive-OR-Sum-of-products expressions for multiple-valued input two-valued output functions," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 5, May 1993, pp. 621-632.

[22] C. C. Tsai and M. Marek-Sadowska, "Efficient minimization algorithms for fixed polarity and/xor canonical networks," *Great Lake Symp. VLSI*, pp. 76-79, 1993.

[23] S. Yang,"Logic synthesis and optimization benchmark user guide, version 3.0", *MCNC*, Jan. 1991.