設計自動化　５０－１０
（1989.12.15）

入力デコーダ付きＡＮＤ－ＥＸＯＲ形ＰＬＡの設計アルゴリズム

A design algorithm for AND-EXOR PLA's with input decoders

笹尾　勤　Tsutomu SASAO
東田基樹　Motoki HIGASHIDA

九州工業大学・情報工学部
Department of Computer Science and Electronic Engineering,
Kyushu Institute of Technology

三菱電機㈱
Mitsubishi Electric Corporation
Kamakura, Japan

あらまし　入力デコーダ付きＡＮＤ－ＥＸＯＲ形ＰＬＡの簡単化問題は，多値入力二値出力関数を表現するＥＳＯＰ（排他的論理和を用いた積和形論理式）の簡単化問題に対応する．本稿では，５種類の積項の変形ルールを用いたＥＳＯＰの簡単化アルゴリズムを提案する．本アルゴリズムを用いて，多くの算術回路用ＰＬＡを簡単化した．その結果，１ビット入力デコーダ付及び２ビット入力デコーダ付のＰＬＡのいずれの場合でも，ほとんどの例で，ＡＮＤ－ＥＸＯＲ形ＰＬＡはＡＮＤ－ＯＲ形ＰＬＡより少ない積項数で実現できた．

Abstract: An optimization problem of AND-EXOR PLA's with input decoders can be regarded as a minimization problem of Exclusive-or Sum-Of-Products expressions (ESOP's) for multiple-valued input two-valued output functions. In this paper, we propose a minimization algorithm forESOP's. The algorithm is based on an iterative improvement. Five rules are used to replace a pair of products with another one. We minimized various ESOP's for arithmetic circuits. In most cases, ESOP's required fewer products than SOP's to realize same functions.

１．まえがき

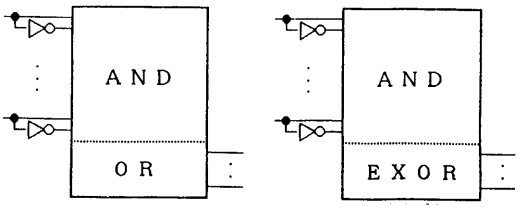
PLAは，自動設計が容易であり，設計変更に対してもアレイパターンの変更のみでよい等の特徴をもつため，多くのLSIで使用されている．これらは，図1.1(a)のようなAND-OR２段論理を実現するPLAである．PLAの論理構造としては，この他にORアレイをEXORのアレイに置き換えた，図1.1(b)のようなAND-EXOR形PLAが考えられる．

AND-EXOR形PLAは，AND-OR形PLAに比べて，いくつかのよい性質をもっている．一つは，AND-EXOR形PLAの方がAND-OR形PLAより，一般に積項数が少ないと推測できることである．二つめは，AND-EXOR形PLAはAND-OR形PLAに比べて検査が容易であることである．AND-OR形PLAは，若干の回路を付加することにより，万能検査を有するようにできる[FUJ81]．AND-EXOR形PLAも，同様に万能検査を有するようにできるが，AND-OR形PLAに比べ，必要な付加回路は少なく，検査長も短くなる[SAS87]．

このようにAND-EXOR形PLAは多くの特長を持っているが，実用化するには問題も多い．一つは，EXORの実現は，ORの実現に比べてコストがかかることである．もう一つの問題は，実用的な設計手法が確立されていないことである．本稿では，この設計手法に関して考察する．

AND-OR形PLAの設計問題は，積和形の論理式(SOP:Sum-Of-Products)の簡単化問題に対応する．SOPの簡単化問題は，古くから研究されており，そのための優れた設計プログラムも開発されている[MUR79,HON74,SAS84]．一方，AND-EXOR形PLAの設計問題は，排他的論理和を用いた積和形論理式(ESOP:Exclusive-or Sum-Of-Products)の簡単化問題となる．

ESOPの簡単化問題は，SOPの簡単化に比べて，はるかに



(a)1ビットデコーダ付 AND-OR形PLA　(b)1ビットデコーダ付 AND-EXOR形PLA

図1.1　PLAの論理構造

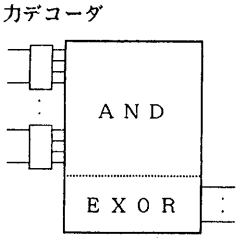


図1.2　入力デコーダ付AND-EXOR形PLA

難しい問題であり，現在も最簡形を効率よく求める方法は知られていない．ただし，ESOPに必要な積項数は，SOPに必要な積項数より一般に少ないと推測されている[EVE67,SAS86b]．従って，AND-EXOR形PLAはAND-OR形PLAより少ない積項数により実現できると考えられる．

本稿では，通常のAND-EXOR形PLAと併せて，図1.2のような入力デコーダ付のAND-EXOR形PLAを考える．そのため，従来の二値入力のESOPを多値入力のESOPに拡張する．また，積項の変形ルールの繰り返し適用による，多値入力ESOPの簡単化アルゴリズムを示す．さらに，このアルゴリズムによる簡単化結果を示す．

2．多値入力ＥＳＯＰ

2.1　多値入力ESOP

入力デコーダ付AND-OR形PLAの簡単化は多値入力二値出力関数を表現するSOPの簡単化に対応する[SAS84]．同様に，入力デコーダ付AND-EXOR形PLAの簡単化は，多値入力二値出力関数を表現するESOPの簡単化に対応する．ここではまず，ESOPを定式化する．

【定義2.1】　写像　$f:P_1\times P_2\times\cdots\times P_n\rightarrow B$，$P_i=\{0,1,\cdots,p_i-1\}$, $B=\{0,1\}$　を多値入力二値出力関数とよぶ．

【定義2.2】　Ｘを$P=\{0,1,\cdots,p-1\}$のいずれかの値をとる変数とする．$S\subseteq P$とすると，X^SをＸのリテラルとよび，

$$X^S = \begin{cases} 1 & (X\in S\text{のとき}) \\ 0 & (X\notin S\text{のとき}) \end{cases}$$

と定義する．

【定義2.3】　$X_1^{S_1}\cdot X_2^{S_2}\cdot\cdots\cdot X_n^{S_n}$ を論理積，

$\Sigma_{(S_1,S_2,\cdots,S_n)} \oplus X_1^{S_1}\cdot X_2^{S_2}\cdot\cdots\cdot X_n^{S_n}$ を，

排他的論理和を用いた積和形論理式（ ESOP：Exclusive-or Sum-Of-Products）とよぶ．

【定理2.1】　任意の多値入力二値出力関数ｆは，ESOPにより表現できる．

（証明）　ｎ変数の最小項を$(a_1,a_2,\cdots,a_n)$により表す．このとき，関数ｆは，

$$\Sigma_{(a_1,a_2,\cdots,a_n)} \oplus f(a_1,a_2,\cdots,a_n)\cdot X_1^{a_1}\cdot X_2^{a_2}\cdot\cdots\cdot X_n^{a_n}$$

と表現できる．この式はESOPである．（証明終）

【定義2.4】　１つの関数を表すESOPは多数存在する．このうち，積項数が最小のESOPを最小ESOPと呼ぶ．

【例2.1】　表2.1に多値入力二値出力関数ｆを示す．ここで，$f:P_1\times P_2\times P_3\rightarrow B$，$P_1=P_2=B$，$P_3=\{0,1,2\}$である．カルノー図により表現すれば，図2.1のようになる．関数ｆの最小ESOPは，

$$f=X_1^{\{1\}}\cdot X_2^{\{1\}}\oplus X_3^{\{2\}}$$

となる．　（例題終）

表2.1　関数ｆ

$X_1X_2X_3$	f
0 0 0	0
0 0 1	0
0 0 2	1
0 1 0	0
0 1 1	0
0 1 2	1
1 0 0	0
1 0 1	0
1 0 2	1
1 1 0	1
1 1 1	1
1 1 2	0

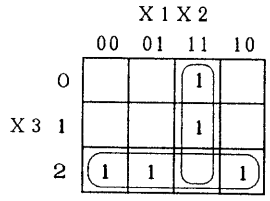


図2.1　関数ｆのカルノー図

2.2　多出力関数

論理回路の出力数が２以上の場合，各出力を独立に最小化しても，全体として積項数が最小のPLAが得られるとは限らない．多出力関数を全体として最小化する方法を示す．

【定義2.5】　多値入力二値ｍ出力関数を

$f_i(X_1,X_2,\cdots,X_n)$　　$(i=0,1,\cdots,m-1)$とする．このとき，多出力関数$(f_0,f_1,\cdots,f_{m-1})$の特性関数Ｆを，

$$F(X_1,X_2,\cdots,X_n,X_{n+1}) = \bigvee_{i=0}^{m-1} X_{n+1}^{\{i\}}\cdot f_i(X_1,X_2,\cdots,X_n)$$

と定義する[SAS84]．

特性関数Ｆでは，入力と出力の組合せが，元の多出力関数で許されるときＦ＝１となり，許されないときＦ＝０となる．

【定理2.2】　多値入力二値多出力関数$(f_0,f_1,\cdots,f_{m-1})$を実現するPLAの最小化は，特性関数FのESOPの最小化に対応する．

（証明）特性関数Fを表現する最小ESOPの一つをΦ_1とし，その積項数をt_1とする．Fの定義域を$X_{n+1}=i$に制限すれば，関数f_iが得られる．従って，Φ_1の積項を積項線に対応させ，X_{n+1}を出力部とするPLAをつくれば，このPLAは多出力関数（$f_0, f_1, \cdots, f_{m-1}$）を実現する．このPLAの積項線数は，$\Phi_1$の積項数と等しい．次に，多出力関数（$f_0, f_1, \cdots, f_{m-1}$）を実現するPLAで積項数が最小のものをPLA1とする．PLA1から特性関数Fを表現するESOP（これをΦ_2とする）を得ることができる．PLA1の積項線数をt_2とすると，これはΦ_2の積項数に等しい．特性関数を表現するESOPの積項数を考えると，Φ_1が最小なので，$t_1 \leqq t_2$ が得られる．次に，多出力関数を実現するPLAを考えると，PLA1が最小なので，$t_2 \leqq t_1$が得られる．これより，$t_1 = t_2$ となる．すなわち，特性関数を表すESOPを最小することによって，多出力関数関数を実現するPLAを最小化できる．（証明終）

【例2.2】 表2.2の二値２入力３出力関数(f_0, f_1, f_2)を考える．特性関数Fは，例2.1の表2.1の関数fと等しくなる．特性関数Fの最簡形は，

$$F = X_1^{\{1\}} \cdot X_2^{\{1\}} \oplus X_3^{\{2\}}$$

である．ここで，３値変数X_3は，出力部を表す．
従って，３出力関数（f_0, f_1, f_2）の最簡形は，

$$f_0 = f_1 = X_1^{\{1\}} \cdot X_2^{\{1\}}$$

$$f_2 = X_1^{\{1\}} \cdot X_2^{\{1\}} \oplus 1$$

となる．この式に対応するAND-EXOR形PLAを，図2.2に示す．（例題終）

表2.2 多出力関数

X_1X_2	f_0 f_1 f_2
0 0	0 0 1
0 1	0 0 1
1 0	0 0 1
1 1	1 1 0

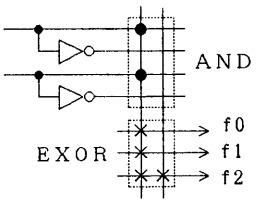


図2.2 AND-EXOR形PLA

３．簡単化アルゴリズム

3.1 簡単化アルゴリズムの基本方針

現在まで，多値入力のESOPの簡単化を直接扱った論文は[PER89]以外には知られていない．ただし，二値論理関数を表現するESOPは，スイッチング理論の初期から考察されている．本節では，現在までの主なESOPの簡単化のアルゴリズムの概略とその問題点を指摘し，本稿の提案するアルゴリズムの基本方針を述べる．

ESOPの基本形として，Reed-Muller展開(RME)がある[REE54,MUL54]．このRMEの一般化が考えられている．ESOPの簡単化アルゴリズムの一つの方式は，一般化したRMEの最適解を求める方法である[MUK70,SAL79,ROB82]．しかし，一般化したRMEの最適解を求めても，最小ESOPとは限らない．ある関数は，最小ESOPでは数個の積項で実現できるが，一般化したRMEでは積項数は入力数の指数関数に比例して増える[SAS86b]．RMEによるESOPの簡単化では，最小ESOPに比べ，積項数がかなり多くなるといった問題がある．

最近，RMEの制限のないESOPの簡単化アルゴリズムも多く研究されている．これらのアルゴリズムでは，RMEまたは最小項表現を初期解とし，積項の変形ルールを用いてESOPの積項数の削減を行う[FLE87,EVE67,HEL88,BES83,PAP79]．この場合，入力数が大きくなると初期解の積項数が爆発的に増大するといった問題がある．

このような問題を考慮して，本稿では次のような簡単化アルゴリズムを提案する．

(i)SOPをディスジョイントに分解したESOPを初期解として用いる．

(ii)５種類の積項の変形ルールを繰り返し用いることにより，ESOPの簡単化を行う．

(iii)多出力関数を品質よく簡単化するため，一度各出力を独立に簡単化した後，全体を簡単化する．

3.2 初期解

本稿で提案するアルゴリズムは，ルールを繰り返し適用して簡単化を行う．従って，計算時間の短縮のためにも，また大きなデータを扱えるようにするためにも，初期解として積項数の少ないESOPを得ることは重要である．本アルゴリズムでは，簡単化したSOPをディスジョイントに分解し，ESOPの初期解としている．

【定義3.1】 どの積項も互いに同じ最小項を共有しないようなＳＯＰをＤＳＯＰ（Disjoint SOP）と呼ぶ．

DSOPのOR演算をEXOR演算に置き換えると，同じ関数を表すESOPとなる．DSOPによる初期解は，最小項を基にして併合と整形演算を繰り返し行って得られるESOPとほぼ同数の積項数となる．従って，この初期解は，かなり積項数の少ないESOPとなっている．

3.3 適用ルール

本稿で提案するアルゴリズムは，次の５種類の積項の変形ルールを繰り返し適用して，ESOPの簡単化を行う．等号の左の２つの積項が，右の１つまたは２つの積項に変形する．

(1) exor-merge

$$X^a \oplus X^b = X^{(a \oplus b)}$$

(2) reshape

$$X^a Y^b \oplus X^c Y^d = X^a Y^{(b \cap \bar{d})} \oplus X^{(a \cup c)} Y^d$$

$$\text{if } (a \cap c = \phi,\ b \supset d)$$

(3) dual-complement

$$X^{a}Y^{b} \oplus X^{c}Y^{d} = X^{c}Y^{(b \cap \bar{d})} \oplus X^{(\bar{a} \cap c)}Y^{b}$$

if (a ⊂ c , b ⊃ d)

(4) x-cover-1

$$X^{a}Y^{b} \oplus X^{c}Y^{d} = X^{a}Y^{(b \cup d)} \oplus X^{(a \cup c)}Y^{d}$$

$$= X^{(a \cup c)}Y^{b} \oplus X^{c}Y^{(b \cup d)}$$

if (a∩c = φ , b∩d = φ)

(5) x-cover-2

$$X^{a}Y^{b} \oplus X^{c}Y^{d} = X^{(a \cup c)}Y^{b} \oplus X^{c}Y^{(b \cap \bar{d})}$$

if (a∩c = φ , b ⊃ d)

それぞれのルールに対応した積項の変形の模式図を，４値入力の場合を例にとって，図3.1に示す．この５種類のルールのうち，積項数を削減できるルールはexor-mergeのみである．その他のルールは，(1)のルールが適用可能になるように，積項の変形を行うためのルールである．(2)はSOPの簡単化にも用いられる積項の変形ルールである[SAS86a]．(3)～(5)のルールがESOPに特有なルールである．SOP実現では多くの積項が必要であるがESOP実現では少ない積項でよいパリティ関数のような関数に対しては，(4),(5)のルールをうまく適用することにより積項数を多く削減できる．(4)のルールには，２種類の変形が可能であるが，その間の変形は(3)のルールを用いて行うことができる．また，(2)と(3)のルールは，同じルールを２度適用すれば，元の積項に戻る．

3.4 簡単化アルゴリズム

アルゴリズムは，積項数を削減できる(1)のルールを中心にして，(1)のルールが適用できなければ，ルール(2)～(5)を順次適用する．現在のアルゴリズムでは，ルールの適用順，また積項の順序に関するストラテジは考えていない．ただし，二値入力の多出力関数の場合，各出力を単独で簡単化し，その後，全体を簡単化するようにしている．次節の例題からもわかるように，ルールの適用順を考慮することは重要である．

多値入力の関数の場合，これらのルールを適用するに当たって，注意すべき点がある．それは，x-cover-1またはx-cover-2とdual-complementを適当に組み合わせると，元のESOPに戻ることである．一例を，図3.2に示す．このようなルールの適用によるプログラムの無限ループを防止するため，ESOPの体積という概念を導入する．

【定義3.2】　集合Ｓの要素の数を｜Ｓ｜により表す．このとき，

積項 $X_1^{S_1} \cdot X_2^{S_2} \cdot \cdots \cdot X_n^{S_n}$ の体積を，

$|S_1| \cdot |S_2| \cdot \cdots \cdot |S_n|$

と定義する．さらに，ESOPの全ての積項の体積の総和を**ESOPの体積**と呼ぶ．

ルールの適用による体積の変化を考えてみよう．(2)のルールでは，ルール適用前と適用後では積項の体積は変化しない．(4)及び(5)のルールは適用により体積が増大する．(3)のルールでは，ルールの適用により，二値入力では体積は変化しないが，多値入力では変化する場合がある．図3.2の例では，x-cover-2により体積が増大するが，dual-complementにより体積は減少しているため，元のESOPに戻っている．このように，(4)または(5)のルールの適用後に(3)のルールを適用すると，(4),(5)のルールの適用前の積項の形に戻ることがある．

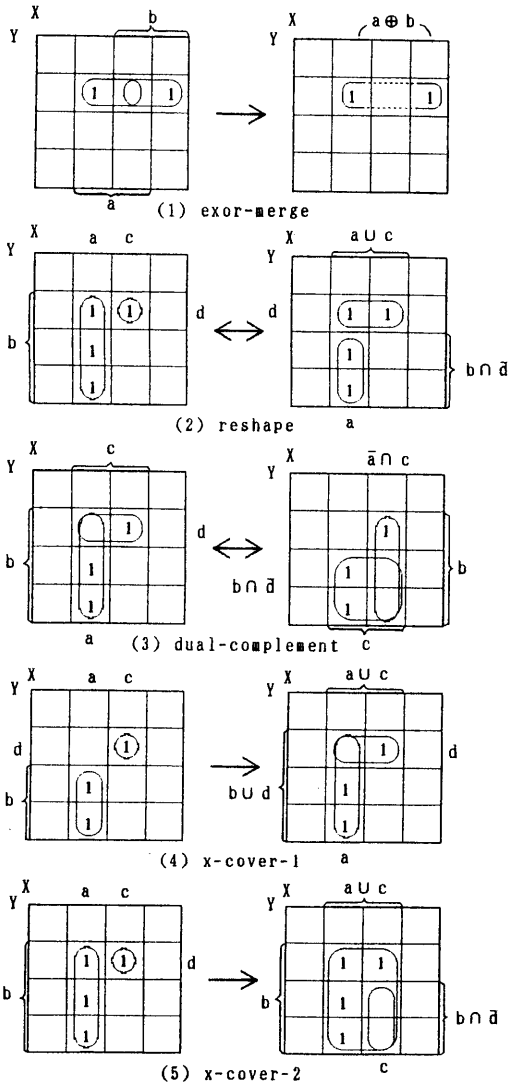


図3.1　適用ルールの模式図

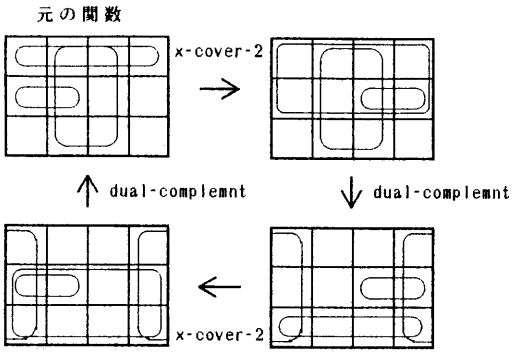


図3.2　(3)と(5)のルールの適用によるループ

アルゴリズム3.1は，ルールの適用によりESOPの体積が増大し，最終的にルールが適用できなくなり停止する．多値入力の場合には，(3)のルールの適用に特別の配慮を行って，無限ループに陥ることを防止している．

【アルゴリズム3.1】（ESOP簡単化）

（１）SOPの入力データをDSOPに変換し，ESOPの初期解とする．

（２）入力が二値の場合，各出力関数毎に以下の処理を行い，さらにその結果を統合して全関数で再び以下の処理を行う．入力が多値の場合，以下の処理を全関数に対して，１度だけ行う．

(2-1)Exor-mergeの条件を満たす全ての積項の組合せに対して，Exor-mergeを行う．

(2-2)全ての積項の組合せに対して，Resahpe, Dual-complement, X-cover-1, X-cover-2の適用を順次試みる．ルールが適用できた場合，次の手順の処理を行う．

(i)　ルールの適用により変化した積項に対して，Exor-mergeの適用を試みる．

(ii)　(i)によりExor-mergeできれば，(2-2)の処理を初めから行う．また，X-cover-1，X-cover-2のルールでは，Exor-mergeが適用できなくとも(2-2)の処理を初めから行う．

(iii)　多値入力関数の場合，Dual-complementを適用し，かつ，(i)でExor-mergeが適用できなければ，Dual-complementの適用前の状態に戻す．

3.5　適用例

図3.3のカルノー図に示す二値４入力関数に，アルゴリズム3.1を適用する．図3.3の表すSOPは，既にDSOPとなっている．また，これ以上exor-mergeを適用できない．従って，アルゴリズム3.1の(2-2)のみが残された処理である．

適用例Ⅰでは，アルゴリズム3.1に従ったルールの適用を行う．適用例Ⅱは，(4)と(5)のルールの適用順を入れ換えた場合のアルゴリズムの適用例である．

（適用例Ⅰ）　図3.4(a)の積項に(2)のルールを適用してもexor-mergeを適用できない．また，(3)のルールも適用できないので，x-cover-1の適用を試みる．適用できる積項対は，（①，②）と（②，③）の２対ある．（①，②）にx-cover-1を適用すると図3.4(b)となる．この時，（④，⑤）の積項対がexor-mergeできる．exor-mergeの結果，図3.4(c)となる．図3.4(c)の表すESOPには(1)～(5)のどのルールも適用することができない．従って，アルゴリズムは停止し，積項数４のESOPとなる．

（適用例Ⅱ）　図3.5(a)の積項には，適用例Ⅰと同様に(2),(3)のルールは適用できない．次に，ルールの適用順を変えて，x-cover-2の適用を考える．適用できる積項対は，（①，②）と（③，④）と（③，⑤）の３対存在する．（①，②）の積項対にx-cover-2を適用すると，図3.5(b)となる．ここで（⑦，④）に対してx-cover-2を適用すると，図3.5(c)となる．（③と⑧）に対して再び，exor-mergeを適用すると

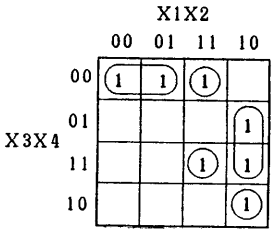


図3.3　カルノー図

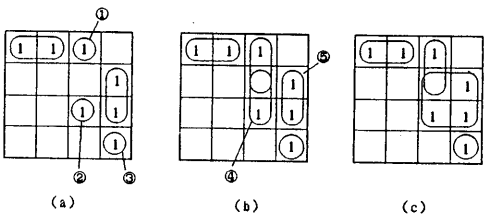


図3.4　適用例Ⅰ

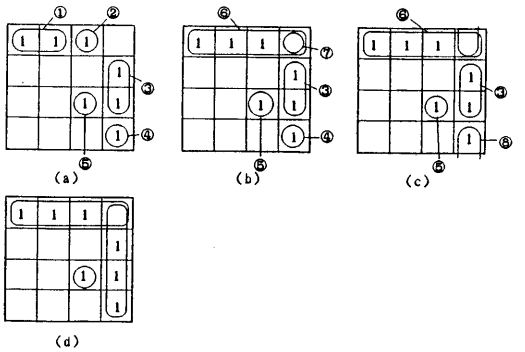


図3.5　適用例Ⅱ

図3.5(d)となる．これ以上，ルールの適用はできず，アルゴリズムは停止する．最終結果は，積項数３のESOPとなっている．

適用例Ⅰでは積項数４となり，適用例Ⅱでは積項数３となっている．このように，ルールの適用順により，最終結果が大きく左右される．適用例Ⅱの場合，図3.5(a)で積項対（③，⑤）にx-cover-2を適用した場合，適用例Ⅰと同じ結果となる．

４．実験結果と考察

３章で述べたESOPの簡単化アルゴリズムを，PC9800RA上にFORTRANを用いて実現し，多くの算術関数に対して簡単化を行った．

HelliwellとPerkowskiはESOPの簡単化プログラムを作成し，結果のデータを詳細に報告している[HEL88]．彼らの方法は，最小項表現を初期解として，Xlinkとよぶ積項の変形操作により簡単化を行う．同じ関数に対して簡単化を試み，CPU時間と積項数を示した結果が表4.1である．Helliwellらの使用計算機は10MHzのIBM-PC AT である．アルゴリズム3.1は，積項数については同じかよい結果を求めている．CPU時間に関しては，積項数の多いADR4,MLP4,SQR6ではるかに短時間で解を求めている．使用計算機が異なっているため単純に比較できないが，積項数が大きな回路では，本アルゴリズムの方が高速であると思われる．その理由の一つしては，初期解の差（本アルゴリズムではDSOP，彼らは最小項）が考えられる．

また，１ビット，及び２ビットデコーダ付のAND-OR形またはAND-EXOR形のPLAに必要な積項数を８種の算術回路[SAS86a]に対して求めた結果を表4.2に示す．AND-OR形PLAの簡単化にはＱＭ法[SAS84]を用いた．２ビットデコーダ付ＰＬＡのデータは，全ての入力変数の組合せを網羅的に調べた結果を示しており，最小解である．一方，AND-EXOR形PLAのデータは，アルゴリズム3.1において，ルールの適用順を種々変化させて得られた解のうちから最小のものを選んだ，準最適解である．そのため，ADR4とMLP4の結果は，表4.1の結果と異なっている．

表4.1　Helliwellらの論文の結果とアルゴリズム3.1の比較

データ名	Helliwell Perkowski		ｱﾙｺﾞﾘｽﾞﾑ 3.1	
	項数	TIME Sec	項数	TIME Sec
ADR2	8	0	8	1
ADR4	34	840	34	33
ROTP6	8	10	7	1
ROTP8	22	240	19	9
SQR3	7	0	7	1
SQR6	40	180	38	20
MLP3	19	48	18	4
MLP4	119	7800	73	94

表4.2　AND-OR形PLAとAND-EXOR形PLAの積項数

データ名	AND-OR 1ﾋﾞｯﾄﾃﾞｺｰﾀﾞ付	AND-OR 2ﾋﾞｯﾄﾃﾞｺｰﾀﾞ付	AND-EXOR 1ﾋﾞｯﾄﾃﾞｺｰﾀﾞ付	AND-EXOR 2ﾋﾞｯﾄﾃﾞｺｰﾀﾞ付
ADR4	75	17	33	12
LOG8	123	98	118	106
MLP4	121	85	68	67
NRM4	120	70	86	71
RDM8	76	52	33	30
ROT8	57	38	42	32
SQR8	180	147	134	129
WGT8	255	54	75	38

表4.2からわかるように，AND-OR形PLA，AND-EXOR形PLAの場合共に，１ビットデコーダ付PLAより２ビットデコーダ付PLAの方が積項数が少ない．また，１ビットデコーダ付PLAの場合，AND-EXOR形PLAの方がAND-OR形PLAより積項数が少なくなった．２ビットデコーダ付の場合も，LOG8とNRM4を除いて同様の傾向が見られる．全体的には，ほぼ次のような傾向となった．

１ビットデコーダ付AND-OR形PLAの積項数
＞１ビットデコーダ付AND-EXOR形PLAの積項数
＞２ビットデコーダ付AND-OR形PLAの積項数
＞２ビットデコーダ付AND-EXOR形PLAの積項数

５．あとがき

本稿では，入力デコーダ付AND-EXOR形PLAの設計法を考察した．このPLAの設計は，多値入力二値出力関数のESOP（Exclusive-or Sum-Of-Products expression）の簡単化により実行できる．本稿では，５種類のルールを用いた繰り返し改善法による簡単化アルゴリズムを提案した．本アルゴリズムをPC98RA上に実現し，多くの算術回路に対して簡単化を試みた．なお，本文は[SAS89]を加筆修正したものである．

参考文献

[BES83] Ph.W.Besslich,"Efficient computer method for ExOR logic design,"IEE Proc.,vol.130, Part E, pp. 203-206, 1983.

[EVE67] S.Even, I.Kohavi and A.Paz,"On minimal modulo-2 sums of products for switching functions,"IEEE Trans. Electronic Computers,

Vol.EC-16, pp. 671- 674,Oct.1984.
[FLE87] H.Fleisher, M.Tavel and J.Yeager, "A computer algorithm for minimizing Reed-Muller canonical forms,"IEEE Trans.Comput.Vol.C-36, No.2, pp. 247-250, Feb.1987.
[FUJ81] H.Fujiwara and K.Kinoshita, "A Design of programmable logic arrays with universal tests," Joint special issue on Design for Testability IEEE Trans. Comput., Vol.C-30, No.11, pp. 823-828; and IEEE trans. Circuit and Systems, Vol.CAS-28, No.11, pp. 1027-1032, Nov. 1981.
[HEL88] M.Helliwell and M.Perkowski,"A fast algorithm to minimize multi-output mixed-polarity generalized Reed-Muller forms," 25th DAC, pp. 427-432, 1988.
[HON74] S.J.Hong, R.G.Cain and D.L.Ostapko, "MINI: A heuristic approach for logic minimization," IBM J.Res. & Develop. pp. 443-458, Sept. 1974.
[MUK70] A. Mukhopadhyay and G.Schmitz, "Minimization of Exclusive OR and logical Equivalence of switching circuits," IEEE Trans. Comput., C-19, pp. 132-140, 1970.
[MUL54] D.E.Muller,"Application of Boolean algebra to switching circuit design and to error detection,"IRE Trans. Electron. Comput., EC-3, pp. 6 -12, 1954.
[MUR79] S.Muroga,Logic design and Switching Theory Wiley-Interscience Publication, 1979.
[PAP79] G.Papakonstantinou,"Minimization of modulo -2 sum of products,"IEEE Trans. Comput., C-28, pp. 163-167, 1979.
[REE54] I.S.Reed, "A class of multiple-error-correcting codes and the decoding scheme," IRE Trans. Information Theory PGIT-4. pp. 38-49, 1954.
[ROB82] J.P.Robinson and Chia-Lung Yeh,"A method for modulo-2 minimization", IEEE Trans. Comput., C-31, pp. 800-801, 1982.
[SAL79] K.K.Saluja and E.H.Ong,"Minimization of Reed-Muller canonic expansion," IEEE Trans. Comput.,C-28, pp. 535-537,1979.
[SAS84] T.Sasao,"Input variable assignment and output phase optimization of PLA's,"IEEE Trans. Comput., Vol.C-33,No.10, pp. 879-894, Oct. 1984.
[SAS86a] 笹尾，ＰＬＡの作り方・使い方，日刊工業新聞社，1986.
[SAS86b] 笹尾，Ph.W.Besslich，”ＥＸＯＲアレイ付きＰＬＡの複雑度，”電子通信学会ＦＴＳ研究会，FTS86-17, 1986-11.
[SAS87] 笹尾，藤原，”万能検査集合をもつＡＮＤ－ＥＸＯＲ形ＰＬＡ”，電子通信学会ＦＴＳ研究会，FTS86-25,1987-02.
[SAS89]笹尾，東田，”入力デコーダ付AND-EXO形PLAの設計アルゴリズムについて，”第20回FTC研究会資料，1989年１月24日.