Optimization of Multiple-Valued AND-EXOR Expressions using Multiple-Place Decision Diagrams

Tsutomu SASAO
Kyushu Institute of Technology,
Iizuka 820, Japan

Abstract

This paper presents an optimization method for pseudo-Kronecker expressions of p-valued input two-valued output functions by using multi-place decision diagrams for $p = 2$ and $p = 4$. A conventional method using extended truth tables requires memory of $O(3^n)$ to simplify an n-variable expression, and is only practical for functions of up to $n = 14$ variables when $p = 2$. The method presented here utilizes multi-place decision diagrams, and can optimize considerably larger problems. Experimental results for up to $n = 39$ variables are shown.

I Introduction

Increasing complexity of LSIs has made human design of bug-free logic circuits very difficult. Thus, various automatic logic synthesis tools have become indispensable in LSI design. Most of the logic synthesis tools use the design theory for the circuits consisting of AND, OR, and NOT gates. As for control circuits, these tools produce good circuits comparable to the human design. However, they are not so good at the design for arithmetic circuits, error correcting circuits and circuits for tele-communication: such circuits can be simplified when EXOR gates are effectively used. Therefore, in order to develop a logic synthesis tool for such circuits, a design theory utilizing EXOR gates is very important.

In this paper, we consider the most basic design problem using EXORs, i,e., a simplification of AND-EXOR two-level logic circuits. Various classes exist in AND-EXOR type logical expressions. Among them, ESOP (Exclusive-or sum-of-products expression) is the most general class, and requires the fewest products to represent given functions. However, no efficient minimization method is known. This paper considers the minimization of the AND-EXOR type expressions called PSDKROs (pseudo-Kronecker expressions), which require fewer products than fixed polarity Read-Muller expressions to represent given functions, and the minimization is relatively easy. A conventional minimization method for PSDKROs utilizes an extended truth table with 3^n elements, and is practical for functions of up to $n = 14$ variables when $p = 2$ [28].

This paper presents a minimization method for PSDKRO using a MDD (multi-place decision diagram) instead of the extended truth table. The memory requirement of the new method is $O(3^n/n)$. Experimental results up to $n = 39$ variables are shown.

II Definitions and Basic Properties

An arbitrary logic functions can be represented by an AND-EXOR expression. In this part, pseudo-Kronecker expressions (PSDKROs) and exclusive-or sum-of-products expressions (ESOPs) are defined. Also, some basic properties of two-valued input functions are shown.

Theorem 2.1 (Expansion Theorem) *An arbitrary logic functions f can be represented as either*

$$f = 1 \cdot f_0 \oplus X \cdot f_2, \quad -(1)$$
$$f = \overline{X} \cdot f_2 \oplus 1 \cdot f_1, \quad -(2) \text{ or}$$
$$f = \overline{X} \cdot f_0 \oplus X \cdot f_1, \quad -(3)$$

where
$f_0 = f(0, X_2, X_3, \cdots, X_n)$, $f_1 = f(1, X_2, X_3, \cdots, X_n)$, *and* $f_2 = f_0 \oplus f_1$.

(Proof) f can be represented as $f = \overline{X} f_0 \vee X f_1$. Because two terms are mutually disjoint, we have (3). Replacing $\overline{X}$ with $1 \oplus X$ in (3), we have
$f = (1 \oplus X) f_0 \oplus X f_1 = 1 \cdot f_0 \oplus X(f_0 \oplus f_1) = 1 \cdot f_0 \oplus X f_2$.
Replacing X with $1 \oplus \overline{X}$ in (3), we have
$f = \overline{X} f_0 \oplus (1 \oplus \overline{X}) f_1 = 1 \cdot f_0 \oplus \overline{X}(f_0 \oplus f_1) = \overline{X} \cdot f_2 \oplus 1 \cdot f_1$. (Q.E.D.)

Definition 2.1 *Pseudo-Kronecker expressions (PSDKROs) of n-variable two-valued functions are defined recursively as follows:*

1) *Constants* 0 *and* 1 *are PSDKROs.*
2) *Literals X_n and $\overline{X}_n$ are PSDKROs.*
3) *Let $G_0(X_{k+1}, X_{k+2}, \cdots, X_n)$ and $G_1(X_{k+1}, X_{k+2}, \cdots, X_n)$ be PSDKROs, then $G_0 \oplus X_k G_1$, $\overline{X}_k G_0 \oplus G_1$, and $\overline{X}_k G_0 \oplus X_k G_1$ are PSDKROs.*
4) *The only expressions given by 1), 2) or 3) are PSDKROs.*

A PSDKRO for the function f is said to be minimum *if it contains the minimum number of products.*

For two-valued input functions, PSDKROs may have both true and complemented literals of each variable. When the order of the variables for expansion is fixed, an n-variable function has at most $3^{2^{n-1}}$ different PSDKRO expansions [8]. A minimum PSDKRO of

0195-623X/92 $3.00 © 1992 IEEE

the given function can be obtained from the extended truth table with 3^n elements [8] : an ordinary workstation can minimize PSDKROs with up to $n = 14$ variables [28].

If the ordering of the input variables is permuted, the number of products in a PSDKRO may change. So, to obtain the minimum expansion, we need to consider $n!$ different combinations, which is impractical for large n.

Definition 2.2 *An expression obtained by EXORing arbitrary logical products is called exclusive-OR sum-of-products expression* (ESOP). *An ESOP for f is* minimum *if it contains the minimum number of products.*

PSDKROs form a proper subset of ESOPs. For example, $X \oplus Y \oplus \overline{X}\,\overline{Y}$ is an ESOP, but not a PSDKRO. ESOPs require not more products then PSDKROs [28].

No efficient minimization method for ESOP is known, and iterative improvement methods are used to reduce the number of products in the ESOPs. The memory requirement of the iterative improvement method is $O(nr)$, and the computation time is $O(nr^2)$ or $O(nr^3)$, where n is the number of the input variables and r is the number of the products.

Iterative improvement methods cannot prove the minimality, and tend to be time consuming [4, 9, 10, 20, 13, 22, 25, 26]. Because minimum PSDKROs are relatively easy to obtain, they can be used as initial solutions for ESOPs [28].

III PSDKROs with two-valued inputs

3.1 Binary Decision Diagram

Binary Decision Diagrams (BDDs) have gained widespread use in the logic synthesis. We first define BDD and reduced ordered BDD (ROBDD) as in [5].

Definition 3.1 A BDD *is a rooted, directed graph with node set I' containing two types of nodes:*
A nonterminal node *v has as attributes an argument index* index(v) $\in \{1, ..., n\}$, *and two children* low(v), high(v) $\in V$.
A terminal node *v has attributes a value* value(v) $\in \{0, 1\}$.

The correspondence between BDDs and Boolean functions is defined as follows:

Definition 3.2 *A BDD G having root node v denotes a function f_v defined recursively as:*

1. *If v is a terminal node:*
   (a) *If value(v) $= 1$ then $f_v = 1$.*
   (b) *If value(v) $= 0$ then $f_v = 0$.*
2. *If v is a nonterminal node with index (v) $= i$ then f_v is the function:*
$$f_v(x_1, \cdots, x_n) = \bar{x}_i \cdot f_{low(v)}(x_1, \cdots, x_n) \vee x_i \cdot f_{high(v)}(x_1, \cdots, x_n).$$
x_i is call the decision variable for node v.

		XY			
		00	01	11	10
ZW	00	1	1	1	0
	01	0	0	1	1
	11	0	1	0	0
	10	1	0	0	1

Figure 3.1: A 4-variable function

A reduced ordered BDDs *(ROBDD) is the BDD such that:*

1) *For any nonterminal node v, if low(v) is also nonterminal, then index(v) $<$ index(low(v)). Similarly, if high(v) is also nonterminal, then index(v) $<$ index(high(v)).*
2) *low(v) $\neq$ high(v) for any node v, and no two subgraph in the BDD are identical.*

3.2 Principle of minimization

For example, consider a PSDKRO for a 4-variable function $f(X, Y, Z, W)$. As shown in Theorem 2.1, the function $f(X, Y, Z, W)$ can be expanded in one of the following:

$$f(X,Y,Z,W) = 1 \cdot f_0(Y,Z,W) \oplus X \cdot f_2(Y,Z,W), \quad -(1)$$
$$f(X,Y,Z,W) = \overline{X} \cdot f_2(Y,Z,W) \oplus 1 \cdot f_1(Y,Z,W), \quad -(2)$$
$$f(X,Y,Z,W) = \overline{X} \cdot f_0(Y,Z,W) \oplus X \cdot f_1(Y,Z,W), \quad -(3)$$

where $f_0(Y,Z,W) = f(0,Y,Z,W)$, $f_1(Y,Z,W) = f(1,Y,Z,W)$, and $f_2(Y,Z,W) = f_0(Y,Z,W) \oplus f_1(Y,Z,W)$.

Similarly, f_0, f_1, and f_2 are expanded as follows:

$$f_0 = \overline{Y} \cdot f_{00} \oplus Y \cdot f_{01} = 1 \cdot f_{00} \oplus Y \cdot f_{02} = \overline{Y} \cdot f_{02} \oplus 1 \cdot f_{01}, \; f_{02} = f_{00} \oplus f_{01},$$
$$f_1 = \overline{Y} \cdot f_{10} \oplus Y \cdot f_{11} = 1 \cdot f_{10} \oplus Y \cdot f_{12} = \overline{Y} \cdot f_{12} \oplus 1 \cdot f_{11}, \; f_{12} = f_{10} \oplus f_{11},$$
$$f_2 = \overline{Y} \cdot f_{20} \oplus Y \cdot f_{21} = 1 \cdot f_{20} \oplus \overline{Y} \cdot f_{22} = Y \cdot f_{22} \oplus 1 \cdot f_{21}, \; f_{22} = f_{20} \oplus f_{21}.$$

Definition 3.3 A ternary decision tree *is a tree with vertex set I' containing two types of nodes:*
A nonterminal node *v has as attributes an argument index* index(v) $\in \{1, \cdots, n\}$, *and three children* low(v), high(v), exor(v) $\in V$. *A* terminal node *v has attributes a value* value(v) $\in \{0, 1\}$. *The correspondence between ternary decision trees and Boolean functions is defined similar to Definition 3.2.*

Example 3.1 *The 4-variable function in Fig.3.1 can be represented by the ternary decision tree in Fig.3.2. A PSDKRO requires two sub-functions among the three. In order to reduce the number of the products in a PSDKRO, we have to chose two sub-functions with as few products as possible.*

For the expansion of f_0, the PSDKROs for the three sub-functions are $f_{00} = \overline{W}$, $f_{01} = Z \oplus \overline{W}$, and $f_{02} = Z$. Because the PSDKRO for f_{01} has the largest number of products, we use the type (1) expansion:

$$f_0 = f_{00} \oplus Y f_{02} = (\overline{W}) \oplus Y(Z).$$

Next for the expansion of f_1, the PSDKROs for sub-functions are $f_{10} = Z \oplus W$, $f_{11} = \overline{Z}$, and $f_{12} = \overline{W}$.

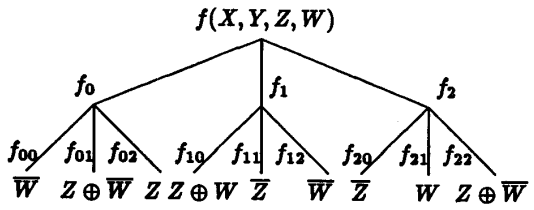

Figure 3.2: Ternary decision tree for 4-variable function

Because the PSDKRO for f_{10} has the largest number of products, we use the type (2) expansion:

$$f_1 = \overline{Y} f_{12} \oplus f_{11} = \overline{Y}(\overline{W}) \oplus (\overline{Z}).$$

Finally for the expansion of f_2, the PSDKROs for the sub-functions are $f_{20} = \overline{Z}$, $f_{21} = W$, and $f_{22} = Z \oplus \overline{W}$. Because the PSDKRO for f_{22} has the largest number of products, we use the type (3) expansion:

$$f_2 = \overline{Y} f_{20} \oplus \overline{Y} f_{21} = \overline{Y}(\overline{Z}) \oplus Y(W).$$

So, the numbers of the products in PSDKROs for f_0, f_1, and f_2 are all 2 in this case. If we use the type (3) expansion for f, we have the PSDKRO with 4 products:

$$\begin{aligned} f(X,Y,Z,W) &= \overline{X} f_0 \oplus X f_1 \\ &= \overline{X}(\overline{W} \oplus Y(Z)) \oplus X(\overline{Y}(\overline{W}) \oplus \overline{Z}) \\ &= \overline{X} \cdot \overline{W} \oplus \overline{X} \cdot Y \cdot Z \oplus X \cdot \overline{Y} \cdot \overline{W} \oplus X \cdot \overline{Z}. \end{aligned}$$

Note that in this case, $f_{00} = f_{12}$, $f_{01} = f_{22}$, and $f_{11} = f_{20}$. Because the same sub-functions have the isomorphic sub-trees, only one sub-tree is necessary for each sub-function to derive the minimum PSDKRO.

(End of Example)

3.3 Representation of functions by TDDs

Definition 3.4 A Reduced Ordered Ternary Decision Diagram *(ROTDD) of f is an acyclic directed graph obtained by reducing the isomorphic sub-trees from the complete ternary decision tree for f.*

ROTDDs can be generated by a similar algorithm to a ROBDD [5, 2, 11, 16]. When the number of the input variables is large, ROBDDs require less memory than truth tables or cube representations. Similarly, ROTDDs require less memory than extended truth tables.

The number of nodes in a complete ternary decision tree for an n-variable function is:

$$1 + 3^1 + 3^2 + \cdots + 3^{n-1} = (3^n - 1)/2.$$

However, in the ROTDD, only one sub-graph is realized for the same sub-functions. Thus, the number of nodes can be reduced.

Lemma 3.1 *All the functions of n or less variables are represented by a BDD with 2^{2^n} nodes.*

(Proof) The proof is done by mathematical induction. For $n = 0$, all the functions (constants 0 and 1) are represented by a BDD with 2 nodes. For $n = 1$, all the functions (constants $0, 1, X, \overline{X}$) are represented by a BDD with 4 nodes as shown in Fig.3.3. Suppose

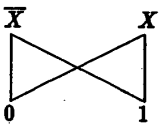

Figure 3.3: BDD for one-variable function

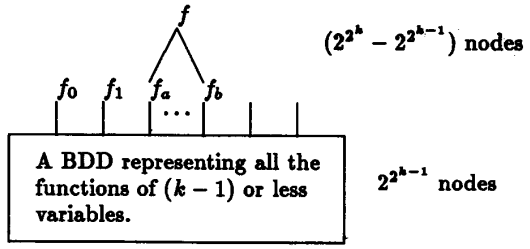

Figure 3.4: A BDD representing all the functions of k or less variables

that all the functions of (k-1) or less variables can be represented by a BDD with $2^{2^{k-1}}$ nodes.

From here, we consider the case of $n = k$.

An arbitrary k-variable function can be represented as $f = \overline{X} f_a \vee X f_b$, where f_a and f_b are functions of $(k-1)$ or less variables. Consider the BDD shown in Fig.3.4. An arbitrary n variable function can be realized in the upper part of Fig.3.4. Thus, the BDD in Fig.3.4 represents all the functions of k or less variables. Note that the number of k-variable functions to generate in the upper part of Fig.3.4 is $2^{2^k} - 2^{2^{k-1}}$, since $2^{2^{k-1}}$ functions are already generated in the lower block. Hence, all the functions of k or less variables are realized by the BDD shown in Fig.3.4. Note that the total number of nodes is 2^{2^k}. (Q.E.D.)

Lemma 3.2 *All the functions of n or less variables can be realized by a TDD with 2^{2^n} nodes.*

(Proof) Suppose that the TDD for all the functions of n or less variables is derived from the BDD shown in Fig.3.4. Since all the functions of k or less variables are already realized in the BDD, the numbers of nodes will not increase. (Q.E.D.)

Theorem 3.1 *An arbitrary n-variable function can be represented by a TDD with at most*

$$N(n) = \min_{k=1}^{n} \left(\tfrac{3^{k+1}-1}{2} + 2^{2^{n-k}} \right) \text{ nodes.}$$

(Proof) Consider the TDD in Fig.3.5, where the upper block is the complete ternary decision tree of k variables, and the lower block generates all the functions of $(n - k)$ or less variables. The complete ternary decision tree for a k-variable function has

$$1 + 3^1 + 3^2 + \cdots + 3^k = (3^{k+1} - 1)/2 \text{ nodes.}$$

By Lemma 3.2, the lower block has $2^{2^{n-k}}$ nodes. Hence, we have the theorem. (Q.E.D.)

Corollary 3.1 *An arbitrary n-variable function can be represented by a ROTDD with $O(3^n/n)$ nodes.*

(Proof) Set $k = n - \log_3 n$ in Theorem 3.1. (Q.E.D.)

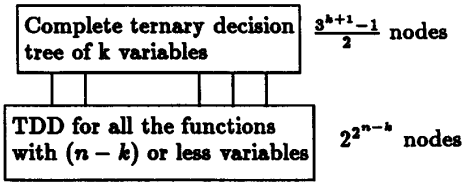

Figure 3.5: Representation of an n-variable function by TDD

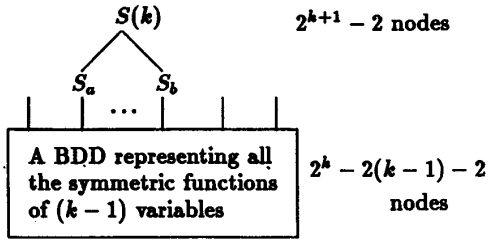

Figure 3.6: BDD representing all the k-variable symmetric function

3.4 Symmetric functions and TDD

Definition 3.5 *An n-variable* fundamental symmetric function $S(n,k)$ *takes the value 1 iff the number of 1's in the inputs is exactly* k $(k=0,1,2,\cdots,n)$.

Lemma 3.3 *All the symmetric functions of n variables can be represented by a BDD with* $2^{n+2}-2n-2$ *nodes.*

(Proof) The proof is done by mathematical induction. For $n=0$, all the symmetric functions (constants 0 and 1) are realized by a BDD with 2 nodes. For $n=1$, all the symmetric functions (constants 0 and 1, X, $\overline{X}$) are realized by a BDD with 4 nodes as shown in Fig.3.3. Suppose that all the symmetric functions of $(k-1)$ variables can be represented by a BDD with $2^{k+2}-2k-2$ nodes.

There are $2^{k+1}-2$ different symmetric functions of k variables and each of them can be represented as

$$S(k)=\overline{X}_k\cdot S_a(k-1)\vee X_k\cdot S_b(k-1),$$

where $S_a(k-1)$ and $S_b(k-1)$ are symmetric functions of $(k-1)$ variables. Therefore, $S(k)$ can be represented by the BDD in Fig.3.6. Note that the total number of nodes is

$$2^{k+1}-2+2^{k+1}-2(k-1)-2=2^{k+2}-2k-2.$$

(Q.E.D.)

Lemma 3.4 *All the symmetric functions of n variables can be represented by a TDD with* $2^{n+2}-2n-2$ *nodes.*

(Proof) Similar to the proof of Lemma 3.2. (Q.E.D.)

Theorem 3.2 *An arbitrary n-variable symmetric function can be represented by a BDD with*

$BN(n)=\min_k\{\frac{(k+1)(k+2)}{2}+2^{n-k+2}-2(n-k)-2\}$ *nodes.*

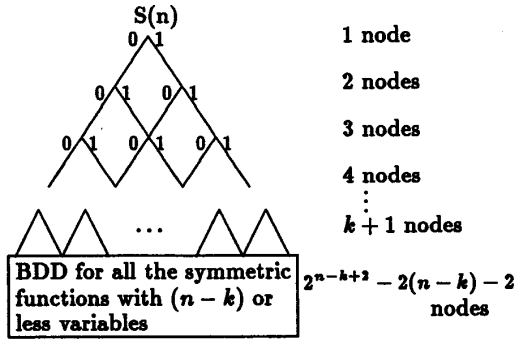

Figure 3.7: BDD for symmetric function of n variables.

(Proof) Consider the BDD shown in Fig.3.7. The nodes in the lowest level of the upper block correspond to fundamental symmetric functions $S(n,k)$, $(k=0,1,\cdots,n)$. Fig.3.7 shows that the total number of nodes is $BN(n)$. (Q.E.D.)

Theorem 3.3 *An arbitrary n-variable symmetric function can be represented by a TDD with*

$$TN(n)=\min_k\{\tfrac{k(k+1)(k+5)}{6}+(k+1)+2^{n-k+2}-2(n-k)-2\}$$

nodes.

(Proof) Consider the complete ternary decision tree of k variables. The different number of functions generated in the lowest level of the tree is derived as follows:

1. Because f is completely symmetric, the permutation of the subscripts of f will not change the function: i.e., f_{021120} is equal to f_{001122}. So the different number of k-variable functions generated by the complete ternary decision tree is equal to "the number of ways to select k objects from 3 distinct objects".
2. "The number of ways to select k objects from p distinct objects" is $C(p+k-1,k)$ [14] .
3. So the number of the different symmetric functions is $(k+1)(k+2)/2$.

The total number of nodes in the ROTDD is $\frac{1}{2}\sum_{i=0}^{k}(i+1)(i+2)=\frac{k(k+1)(k+5)}{6}+(k+1)$. Consider the TDD shown in Fig.3.9, which generates all the symmetric functions of $(n-k)$ variables. By combining the ROTDD derived from Fig.3.8, and the TDD in Fig.3.9, we can derive a TDD which represents an arbitrary symmetric function of n variables. Hence the theorem. (Q.E.D.)

Corollary 3.2 *An arbitrary symmetric function of n variables can be represented by a TDD with* $O(n^3)$ *nodes.*

(Proof) Set $k=n$ in Theorem 3.3. (Q.E.D.)

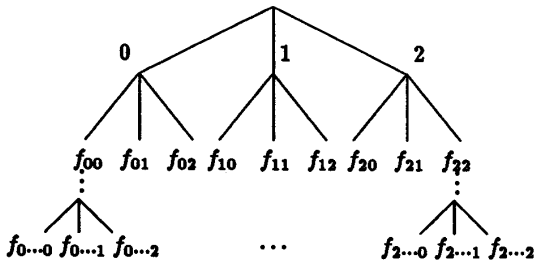

Figure 3.8: Complete ternary decision tree for k-variable function.

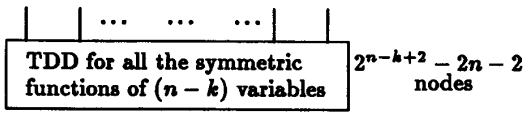

Figure 3.9: TDD for all the symmetric functions of $(n-k)$ variables.

3.5 Minimization Algorithm ($p=2$)

A minimum PSDKRO for a given function can be recursively obtained from the minimum PSDKROs for all the sub-functions.

Algorithm 3.1 (Minimisation of PSDKRO for $p=2$)

1. *Construct the ROTDD for the given function. Let the node i represent the function f_i.*
2. *Compute the cost of each node. $COST(f_i)$ is defined as follows:*
   $COST$(the constant function 0) $= 0$;
   $COST$(the constant function 1) $= 1$; and
   $COST(f_i)$
   $= \sum_{j=0}^{2} COST(f_i{:}j) - \max_{j=0}^{2} COST(f_i{:}j)$,
   where
   $COST(f_i{:}0)$, $COST(f_i{:}1)$, and $COST(f_i{:}2)$ denote the COSTs of PSDKRO for the sub-functions $f_i(0)$, $f_i(1)$ and $fi(0) \oplus fi(0)$, respectively.
3. *For each node, delete a sub-tree with the maximum COST among the three.*
4. *Expand the remaining decision diagrams, and obtain the PSDKRO.*

An m-output function is represented by a function whose values takes m-bit binary vectors: A 2-valued input 2^m-valued output function.

IV Extension to multiple-valued cases

4.1 PSDKROs with multiple-valued inputs

Definition 4.1 *Let $P=\{0,1,\cdots,p-1\}$, $p \geq 2$ and $B=\{0,1\}$. $f{:}P^n \to B$ is a* multi-valued input two-valued output function. *From here, the term* function *means a multiple-input two-valued output function.*

Definition 4.2 *Let $S \subseteq P$. X^S is a* literal *of X, where*

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

When S contains only one element, $X^{\{i\}}$ is denoted by X^i. A product of literals $X_1^{S_1}X_2^{S_2}\cdots X_n^{S_n}$ is a product term. *A sum of products*

$$\bigvee_{(S_1,S_2,\cdots,S_n)} X_1^{S_1}X_2^{S_2}\cdots X_n^{S_n}$$

is a sum-of-products expression *(SOP). Where $\vee_{(S_1,S_2,\cdots,S_n)}$ denotes the inclusive-OR of some tuples of $(S_1,S_2,\cdots,S_n)$. An exclusive-OR of products*

$$\sum\oplus_{(S_1,S_2,\cdots,S_n)} X_1^{S_1}X_2^{S_2}\cdots X_n^{S_n}$$

is an exclusive-OR sum-of-products expression *(ESOP).*

Definition 4.3 *An SOP for f is* a minimum *SOP (MSOP) for f, if it has the minimum number of products. An ESOP is a* minimum ESOP *(MESOP) if it has the minimum number of the products.*

Lemma 4.1 *An arbitrary n-variable function $f(X_1, X_2, \cdots, X_n)$ can be uniquely represented as*

$$f = \bigvee_{(a_1,a_2,\cdots,a_n)} f(a_1,a_2,\cdots,a_n)X_1^{a_1}X_2^{a_2}\cdots X_n^{a_n},$$

where $\vee_{(a_1,a_2,\cdots,a_n)}$ represents the inclusive-OR for all the combinations such that $a_i \in P$, and $f(a_1,a_2,\cdots,a_n) = 0$ or $= 1$.

Lemma 4.2 *An arbitrary n-variable function $f(X_1, X_2, \cdots, X_n)$ can be uniquely represented by an expression:*

$$f = \sum\oplus_{(a_1,a_2,\cdots,a_n)} f(a_1,a_2,\cdots,a_n)X_1^{a_1}X_2^{a_2}\cdots X_n^{a_n},$$

where $\sum\oplus_{(a_1,a_2,\cdots,a_n)}$ represents exclusive-or for all the combinations of $a_i \in P$.

Definition 4.4 *Let S be subsets of $P=\{0,1,\cdots,p-1\}$. Let*

$$a_i = \begin{cases} 0 & (i \notin S) \\ 1 & (i \in S) \end{cases},$$

then, $\vec{a}=(a_0,a_1,\cdots,a_{p-1})$ is called a characteristic vector *of S.*

Lemma 4.3 *An arbitrary n-variable function $f(X_1, X_2, \cdots, X_n)$ can be uniquely represented as*

$$f = \sum\oplus_{j=0}^{p-1} X_1^j \cdot f(j,X_2,\cdots,X_n).$$

This is a multiple-valued version of Shannon's expansion theorem.

Theorem 4.1 (Expansion Theorem) *An arbitrary function $P^n \to B$ can be uniquely represented in the form*

$$f = X^{S_0}\cdot h_0 \oplus X^{S_1}\cdot h_1 \oplus \cdots \oplus X^{S_{p-1}}\cdot h_{p-1} \quad (4.1)$$

if and only if M is non-singular, where $M = \begin{bmatrix} \vec{a}_0 \\ \vec{a}_1 \\ \vdots \\ \vec{a}_{p-1} \end{bmatrix}$, and $\vec{a}_i (i=0,1,\cdots,p-1)$ are the characteristic vectors of S_i.

(Proof) By rewriting (4.1), we have

$$f = \sum\oplus_{j=0}^{p-1} X^{S_j}\cdot h_j.$$

On the other hand, by Shannon expansion of f, we have

$$f = \sum \bigoplus_{j=0}^{p-1} X^j \cdot f_j.$$

Let I be the unit $p \times p$ matrix. The relation of the above two equations can be written as

$$[h_0, h_1, \cdots, h_{p-1}] \cdot M = [f_0, f_1, \cdots, f_{p-1}] \cdot I$$

When M is non-singular, the inverse matrix M^{-1} exists, and the function can be uniquely represented as

$$[h_0, h_1, \cdots, h_{p-1}] = [f_0, f_1, \cdots, f_{p-1}] \cdot M^{-1}.$$

When M is singular, $[h_0, h_1, \cdots, h_{p-1}]$ cannot be uniquely represented. (Q.E.D.)

Definition 4.5 Pseudo-Kronecker expression *(PSDKROs) of n-variable p-valued functions are defined recursively as follows:*

1) *Constants 0 and 1 are PSDKROs.*
2) *A literal $X_n^S (S \subseteq P)$ is a PSDKRO.*
3) *Let $G_j(X_{k+1}, X_{k+2}, \cdots, X_n), (j = 0, 1, 2, \cdots, p-1)$ be PSDKROs, then*
$$f = \sum \bigoplus_{j=0}^{p-1} X_k^{S_j} \cdot G_j(X_{k+1}, X_{k+2}, \cdots, X_n)$$
is also PSDKRO if the matrix M defined in Theorem 4.1 is non-singular.
4) *The only expressions given by 1), 2) or 3) are PSDKROs.*

A PSDKRO for the function f is said to be minimum *if it contains the minimum number of the products.*

4.2 Minimization algorithm ($p = 4$)

From here, we consider the case where $p = 4$.

Theorem 4.2 (Expansion Theorem for $p = 4$) *Let $A, B, C, D, A', B', C', D' \subseteq P$, and $P = \{0, 1, 2, 3\}$. Let $\vec{a}, \vec{b}, \vec{c}, \vec{d}, \vec{a'}, \vec{b'}, \vec{c'}$ and $\vec{d'}$ be the characteristic vectors of A, B, C, D, A', B', C' and D', respectively.*

Let $M = \begin{bmatrix} \vec{a} \\ \vec{b} \\ \vec{c} \\ \vec{d} \end{bmatrix}$, $R = [\vec{a'}^t, \vec{b'}^t, \vec{c'}^t, \vec{d'}^t]$, *and $M \cdot R = I$ (unit matrix). Then, an arbitrary 4-valued input two-valued output function f can be uniquely represented in the forms*

$$f = X^{\{0\}} \cdot f_{\{0\}} \oplus X^{\{1\}} \cdot f_{\{1\}} \oplus X^{\{2\}} \cdot f_{\{2\}} \oplus X^{\{3\}} \cdot f_{\{3\}}$$

or

$$= X^A \cdot f_{A'} \oplus X^B \cdot f_{B'} \oplus X^C \cdot f_{C'} \oplus X^D \cdot f_{D'},$$

where

$$f_{A'} = \sum \bigoplus_{i \in A'} f_{\{i\}}, \quad f_{B'} = \sum \bigoplus_{i \in B'} f_{\{i\}},$$
$$f_{C'} = \sum \bigoplus_{i \in C'} f_{\{i\}}, \quad f_{D'} = \sum \bigoplus_{i \in D'} f_{\{i\}}.$$

(Proof) Obvious from the proof of Theorem 4.1. (Q.E.D.).

Now, we consider the simplification method for PSDKRO. In the case of $p = 2$, we considered three sub-functions. In the p-valued case, we have to consider $(2^p - 1)$ sub-functions. Especially for $p = 4$, we have to consider the following 15 sub-functions:

$f_{\{0\}}, f_{\{1\}}, f_{\{2\}}, f_{\{3\}}$,
$f_{\{01\}} = f_{\{0\}} \oplus f_{\{1\}}, f_{\{02\}} = f_{\{0\}} \oplus f_{\{2\}}, f_{\{03\}} = f_{\{0\}} \oplus f_{\{3\}}$,
$f_{\{12\}} = f_{\{1\}} \oplus f_{\{2\}}, f_{\{13\}} = f_{\{1\}} \oplus f_{\{3\}}, f_{\{23\}} = f_{\{2\}} \oplus f_{\{3\}}$,
$f_{\{012\}} = f_{\{0\}} \oplus f_{\{1\}} \oplus f_{\{2\}}, f_{\{013\}} = f_{\{0\}} \oplus f_{\{1\}} \oplus f_{\{3\}}$,

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

Figure 4.1: Example of a 4-valued input function

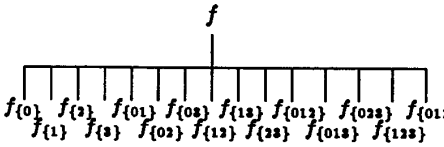

Figure 4.2: Expansion tree having 15 branches.

$f_{\{023\}} = f_{\{0\}} \oplus f_{\{2\}} \oplus f_{\{3\}}, f_{\{123\}} = f_{\{1\}} \oplus f_{\{2\}} \oplus f_{\{3\}}$,
$f_{\{0123\}} = f_{\{0\}} \oplus f_{\{1\}} \oplus f_{\{2\}} \oplus f_{\{3\}}$.

Also in the multiple-valued cases, a minimum PSDKRO for a given function can be recursively obtained from the minimum PSDKROs for all the sub-functions. The following examples illustrate the method.

Example 4.1 *Consider the 4-valued input function f shown in Fig.4.1, which was obtained by pairing the variables in Fig.3.1. Fig.4.2 shows the expansion tree for the function f with respect to X_1. Note that this tree has 15 branches. Table 4.1 shows the sub-functions, and the numbers of products to represent in PSDKROs. The numbers of the products to represent the sub-functions are 0 for $f_{\{023\}}$, and 1 for other functions. The characteristic vectors for the 4 sub-functions $f_{\{023\}}, f_{\{1\}}, f_{\{2\}}$ and $f_{\{3\}}$ are (1011),(0100),(0010), and (0001), respectively. Note that these vectors are linearly independent from each other. Hence, f can be uniquely represented by these 4 sub-functions. Because*

$$R = \begin{bmatrix} 1000 \\ 0100 \\ 1010 \\ 1001 \end{bmatrix} \text{ and } M = R^{-1} = \begin{bmatrix} 1000 \\ 0100 \\ 1010 \\ 1001 \end{bmatrix},$$

f can be represented as

$$f = X_1^A \cdot f_{A'} \oplus X_1^B \cdot f_{B'} \oplus X_1^C \cdot f_{C'} \oplus X_1^D \cdot f_{D'}$$
$$= X_1^{\{1\}} \cdot X_2^{\{0,3\}} \oplus X_1^{\{0,2\}} \cdot X_2^{\{1,2\}} \oplus X_1^{\{0,3\}} \cdot X_2^{\{0,1\}}.$$

(End of Example)

Definition 4.6 Reduced Ordered Penta-decimal Decision Diagram *(ROPDD) of f is an acyclic directed graph obtained by reducing the isomorphic sub-trees from the complete penta-decimal decision tree for f.*

Table 4.1: Sub-functions in Example 4.1.

	f_0	f_1	f_2	f_3	f_{01}	f_{02}	f_{03}	f_{12}	f_{13}	f_{23}	f_{012}	f_{013}	f_{023}	f_{123}	f_{0123}
0	1	1	0	1	0	1	0	1	0	1	0	1	0	0	1
1	0	0	1	1	0	1	1	1	1	0	1	1	0	0	0
2	0	1	0	0	1	0	1	1	1	0	0	1	0	1	1
3	1	0	1	0	1	0	0	1	0	1	0	1	0	1	0
C	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

Table 5.1: Number of products to realize arithmetic functions

Data Name	$\|f\|$	2-valued SOP	2-valued PSD KRO	2-valued ESOP	4-valued SOP	4-valued PSD KRO	4-valued ESOP
adr4	255	75	34	31	17	14	11
log8	255	123	128	96	98	115	94
mlp4	225	121	81	61	85	63	52
nrm4	255	120	105	71	70	72	56
rdm8	255	76	41	31	52	30	26
rot8	255	57	44	35	38	32	28
sqr8	255	180	146	112	147	125	112
wgt8	255	255	107	54	54	34	25

Algorithm 4.1 (Minimisation of PSDKRO with $p = 4$)

1. *Form the ROPDD of the given function f. Let node i denote the function f_i.*
2. *Let $COST(f_i)$ be the cost of the function f_i recursively defined as follows:*
   $COST($*the constant function* $0) = 0$ *and*
   $COST($*the constant function* $1) = 1$.
   $COST(f_i) = COST(f_i{:}A) + COST(f_i{:}B) + COST(f_i{:}C) + COST(f_i{:}D)$,
   where $COST(f_i{:}A)$ denotes the cost of sub-function f_{iA}, etc., and let the sets A, B, C and D satisfy the conditions of Theorem 4.2, and the value of the $COST(f_i)$ be minimum.
3. *For each node, delete the redundant sub-tree.*
4. *Expand the remaining decision diagram, and obtain the PSDKRO.*

V Experimental Results

We developed minimization programs described in III and IV, and optimized various functions. The programs are coded in C language and run on a SPARC station 1+.

5.1 Number of products in PSDKROs

Table 5.1 compares the numbers of the products to represent arithmetic circuits, where $|f|$ denotes number of the products in the original data. The numbers of the products tend to decrease in the following order: $|f|$, SOP, PSDKRO, ESOP. These arithmetic functions were generated by a computer program. They also appear ESPRESSO [3] or MCNC [30] benchmarks. But, some are renamed as follows: nrm4=dist, rdm8=f51m, rot8=root, and wgt8=rd84. Table 5.2 compares the number of the products for other benchmark functions. In this tables, only the functions whose ESOP realizations require fewer products than SOPs are shown. We could not minimize some functions when $p = 4$ because of memory overflow. In these experiments, SOPs are simplified by QM or MINI2 [24], ESOPs are simplified by EXMIN2 [29], and 4-valued input functions are generated by a heuristic algorithm for pairing input variables [24].

5.2 Ordering of the input variables

The ordering of the input variables in the expansion influences the number of the products in PSDKROs. To obtain the minimum PSDKRO for all the orderings, we have to consider $n!$ different combinations.

Table 5.2: Number of products to realize arithmetic functions

Data	IN	OUT	Products 2-valued input SOP	PSD KRO	ESOP	4-valued input SOP	PSD KRO	ESOP
5xp1	7	10	75	47	34	47	34	27
add6	12	7	355	132	127	37	34	23
bc0	26	11	177	180	168	143	-	140
co14	14	1	14	14	14	14	7	7
duke2	22	29	86	108	81	76	-	72
in2	19	10	134	117	113	84	-	71
in7	26	10	54	42	35	44	-	35
inc	7	9	29	31	30	28	27	24
misex3	14	14	690	754	585	457	-	454
rd53	5	3	31	20	15	12	10	9
rd73	7	3	127	63	42	37	25	18
sao2	10	4	58	41	29	38	28	25
t481	16	1	481	13	13	32	9	8
tial	14	8	579	939	506	282	-	190
x6dn	39	5	81	104	95	63	-	75

- : Memory overflow

Table 5.3: Distribution of the number of products in PSDKRO for function (LOG8) when the order of the input variables is changed.

PT	COUN	PT	COUN	PT	COUN	PT	COUN
119	10	126	1446	133	3598	140	908
120	32	127	1824	134	3402	141	626
121	118	128	2166	135	3190	142	388
122	190	129	2410	136	2586	143	266
123	504	130	2816	137	2188	144	152
124	692	131	3112	138	1744	145	58
125	1036	132	3592	139	1256	146	10

PT : Number of the products
COUN : Number of the combinations

For example, LOG8 (logarithm function of 8 bits) has 8 inputs, and the number of the combinations to consider is $8! = 40320$. Table 5.3 shows the distribution of the number of the products. In this function, the number of the products is between 119 and 146.

VI Conclusion and Comments

In this paper, we presented minimization algorithms for PSDKROs by using MDD. We simplified expressions for various functions, and compared the number of products. PSDKROs require more products than ESOPs, but they are easier to minimize than ESOPs. For $p = 2$, the memory requirement for the optimization by TDD is $O(3^n/n)$. Minimal PSDKROs can be used as initial solutions for ESOPs. The experimental results for the functions up to 39-variable functions are shown.
The remaining problems are:

1. Simplification method of ESOPs by using TDDs.
2. Ordering of the input variables which minimizes the number of the products in a PSDKRO.
3. Pairing of the input variables which minimizes the number of the products in a PSDKRO.

VII Acknowledgements

This work was supported in part by a Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan. Prof. Ph. W. Besslich read the manuscript; and Prof. M. Perkowski, Prof. J. Muzio, Prof. Mukerjee and Prof. M. David send me related papers. Mr. T. Amada developed the program.

References

[1] S. B. Aker, "Binary decision diagrams", IEEE Trans. Comput., Vol.C-27. No.6, June 1978, pp.509-516.

[2] K. S. Brace, R. L. Rudell and R. E. Bryant, "Efficient implementation of a BDD package", Proc. 27th Design Automation Conference, June 1990, pp.40-45.

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

[4] D. Brand and T. Sasao, "On the minimization of AND-EXOR expressions", International Workshop on Logic Synthesis, Research Triangle Park, NC, May 1991.

[5] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation", IEEE Trans. Comput. Vol. C-35, No.8, Aug. 1986, pp.677-691.

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

[7] Ph. W. Besslich, "Spectral processing of switching functions using signal-flow transformations", in M. Karpovsky (Ed.), Spectral Techniques and Fault Detection, Orlando, FL:Academic Press, 1985, pp.91-141.

[8] M. Davio, J-P Deschamps, and A. Thayse, "Discrete and switching functions", McGraw-Hill International, 1978.

[9] S. Even, I. Kohavi and A. Paz, "On minimal modulo-2 sum of products for switching functions", IEEE Trans. on Electron Computers, Vol. EC-16,pp.671-674, Oct. 1967.

[10] H. Fleisher, M. Tarvel, and J. Yeager, "A computer algorithm for minimizing Reed-Muller canonical forms", IEEE Trans. on Computers Vol.C-36, No.2 Feb. 1987.

[11] M. Fujita, H. Fujisawa, and N. Kawato, "Evaluation and implementation of Boolean Comparison method base on binary decision diagrams", ICCAD-88, Nov.1988, pp.6-9.

[12] D. H. Green and I. S. Taylor, "Multiple-valued switching circuit design by means of generalized Reed-Muller expansions", Digital Processes,vol.2, pp.63-81, 1976.

[13] M. Helliwell and M. Perkowski, "A fast algorithm to minimize multi-output mixed-polarity generalized Reed-Muller forms", Proc. of the 25th Design Automation Conference, pp.427-432, June 1988.

[14] C. L. Liu, "Elements of discreate mathematics", (second edition), McGraw-Hill, New York, 1985.

[15] P. K. Lui and J. Muzio, "Boolean matrix transforms for the parity spectrum and the minimization of modulo-2 canonical expansions", Department of Computer Science, University of Victoria, DCS-135-IR, July 1990.

[16] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manupulation", Proc. 27th ACM/IEEE Design Automation Conf., June 1990, pp.52-57.

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

[18] M. R. Mukerjee, "Minimization of ring-sum expansion of mixed polarity", AMSE Sympo. on Modelling and Simulation, Greensboro, Oct. 1990.

[19] G. Papakonstantinou, "Minimization of modulo-2 sum of products", IEEE Trans. Comput., C-28, pp.163-167, 1979.

[20] M. Perkowski, M. Helliwell, and P. Wu, "Minimization of multiple-valued input multi-output mixed-radix exclusive sum of products for incompletely specified Boolean functions", Proc. of the 19 th International Symposium on Multiple-valued Logic, pp.256-263, May 1989.

[21] M. Perkowski and M. Chrzanowska-Jeske, "An exact algorithm to minimize mixed-radix exclusive sums of products for incompletely specified Boolean functions", Proc. ISCAS, pp.1652-1655, June 1990.

[22] J. P. Robinson and Chia-Lung Yeh, "A method for modulo-2 minimization", IEEE Trans. Comput., C-31, pp.800-801, 1982.

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

[24] T. Sasao, "Input variable assignment and output phase optimization of PLA's", IEEE Trans. on Comput. vol C-33, No.10, pp.879-894, Oct. 1984.

[25] T. Sasao and P. Besslich, "On the complexity of MOD-2 Sum PLA's", IEEE Trans. on Comput. Vol.39. No.2, pp.262-266, Feb.1990.

[26] T. Sasao, "EXMIN: A simplification algorithm for Exclusive-OR-Sum-of-Products Expressions for multiple-Valued input two-valued output functions", ISMVL-90, May 1990, pp.128-135.

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

[28] T. Sasao, "On the complexity of some classes of AND-EXOR expressions", IEICE Technical Report FTS91-39, October 1991.

[29] T. Sasao, "EXMIN2: A tool for EXOR logic synthesis", Proceedings of the Synthesis and Simulation Meeting and International Interchange, April 1992.

[30] S. Yang, "Logic synthesis and optimization benchmark user guide, Version 3.0", MCNC, Jan. 1991.