Ternary Decision Diagrams
– Survey –

Tsutomu Sasao
Department of Computer Science and Electronics
Kyushu Institute of Technology
Iizuka 820, Japan

Abstract

This paper surveys seven types of TDDs: General_TDD, SOP_TDD, ESOP_TDD, AND_TDD, prime_TDD, EXOR_TDD, and Kleene_TDD. We give new definitions for SOP_TDDs and ESOP_TDDs and introduce unifying terminology. After showing some theorems on complexities, we compare the sizes of these TDDs using benchmark functions. Finally, we review important works on TDDs.

1 Introduction

Various methods exist to represent logic functions. The truth table is the most straightforward method. A sum-of-products expression (SOP) is an another method; it can be converted directly into an AND-OR two-level logic network. A binary decision diagram (BDD) is suitable for representing a complex logic function with many variables [1, 3, 5, 11, 54].

This paper surveys ternary decision diagrams (TDDs). TDDs are similar to BDDs, except that each non-terminal node has three children. Because different TDDs were introduced by different people, there is a need for a unifying terminology applicable to all TDDs. This paper introduces such a terminology.

Section 2 defines seven types of TDDs: General_TDD, SOP_TDD, ESOP_TDD, AND_TDD, prime_TDD, EXOR_TDD and Kleene_TDD. A general_TDD represents an arbitrary ternary function; an SOP_TDD represents an SOP; an ESOP_TDD represents an ESOP; an AND_TDD represents the set of all implicants; an EXOR_TDD represents an extended truth vector, which is used in the optimization of AND-EXOR expressions; a prime_TDD represents the set of all the prime implicants; and a Kleene_TDD represents a Kleene function, which is useful for logic simulation in the presence of unknown inputs. Section 3 analyses the complexity of TDDs. Some theorems and experimental results show the complexities of various TDDs. Section 4 introduces our current research on TDDs. Finally, Section 5 reviews important works on TDDs.

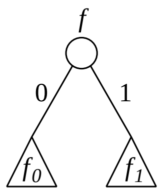


Figure 1. BDD.

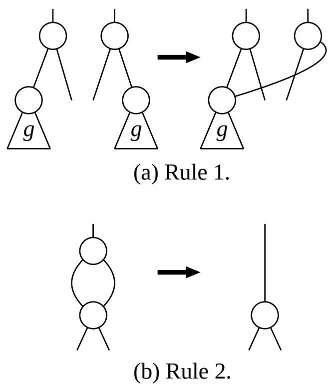


Figure 2. Reduction rules.

2 Various Decision Diagrams

2.1 BDDs

A binary decision diagram (BDD) represents a two-valued logic function f. Let $f = \bar{x}f_0 \vee xf_1$ be the Shannon expansion of f with respect to variable x. Then, the sub-graphs of the BDD represent f_0 and f_1, as shown in Fig. 1. Note that a path in the BDD from the root node to a terminal node represents an assignment of values to the variables. The value of the leaf node is the function value for that assignment. In this paper, we assume that the ordering of the input variables is the same for all paths from the root node to a leaf node, i.e., only ordered decision diagrams (DDs) are considered. We can reduce the DD, i.e., eliminate nodes, by using two rules:

Rule 1: Share equivalent sub-graphs (Fig. 2 (a)).

Rule 2: If descendent nodes of a node η are the same, then delete η and connect the incoming edges of the deleted node to the corresponding successor (Fig. 2 (b)).

Suppose that we have a complete binary decision tree. The BDD reduced by using only Rule 1 is a *quasi reduced ordered BDD* (*QROBDD*). The BDD reduced by using Rule 1 and Rule 2 is a *reduced ordered BDD*

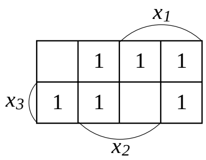


Figure 3. Example function.

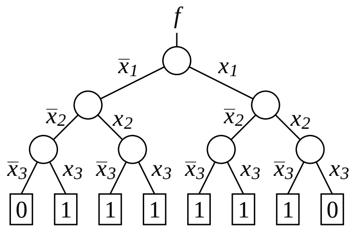


Figure 4. Complete binary decision tree.

(*ROBDD*). The QROBDD and ROBDD are canonical, i.e., unique QROBDD and ROBDD exists for a given function. In this paper, unless noted, both reduction rules are used in DDs. Fig. 3 shows the three-variable function that will be used as examples throughout the paper. Fig. 4 shows the complete binary decision tree for Fig. 3. After reduction, we have the QROBDD and ROBDD shown in Fig. 5 and Fig. 6, respectively.

A path from the root node to the constant 1 node is called a *1-path.* 1-paths in a QROBDD represent a sum-of-minterms expression (Fig. 7 (a)), while 1-paths of an ROBDD represent a disjoint sum-of-products expression (DSOP) (Fig. 7 (b)). In a QROBDD of an n-variable function, any path from the root node to the terminal nodes will visit exactly n non-terminal nodes. The SOP represented by a QROBDD is the same regardless of the order of the input variables, while the SOP represented by an ROBDD depends on order. The *size* of a DD is the number of nodes in the DD. In the case of a ROBDD, the size is $O(2^n/n)$ [29, 65].

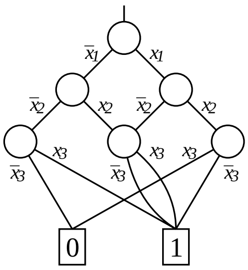


Figure 5. QROBDD.

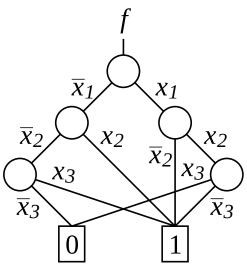


Figure 6. ROBDD.

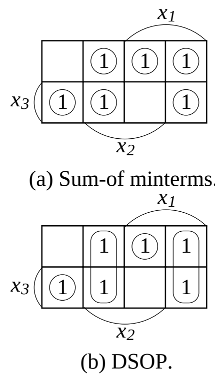


Figure 7. SOPs represented by QROBDD and ROBDD.

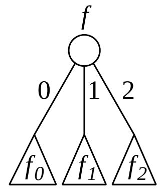


Figure 8. General_TDD.

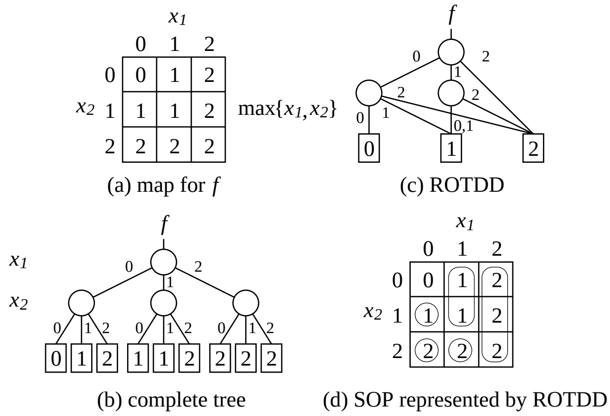


Figure 9. Representation of a three-valued function.

2.2 General_TDDs

A general_TDD is a natural extension of the BDD to the three-valued case. Let $f = x^0 f_0 \vee x^1 f_1 \vee x^2 f_2$ be the three-valued version of the Shannon expansion of an arbitrary three-valued function f: $T^n \to T, T = \{0, 1, 2\}$. Then, the sub-graphs of the general_TDD represent f_0, f_1 and f_2 as shown in Fig. 8. As with BDDs, TDDs are reduced to obtain an ROTDD (reduced ordered TDD). An ROTDD is unique for a given function, i.e., the representation is canonical for a given order of the input variables. Fig. 9 (a) shows a map for the max function of two ternary variables. Fig. 9 (b) shows the complete ternary decision tree; and Fig. 9 (c) shows an ROTDD; and Fig. 9 (d) shows the expressions represented by the 0-paths, 1-paths and 2-paths.

The size of the complete ternary decision tree is $O(3^n)$. But, after reduction, the size of the TDD become $O(3^n/n)$. A general_TDD is a special case of a multiple-valued decision diagram (MDD).

2.3 SOP_TDDs

An SOP_TDD represents a set of products in a sum-of-products expression (SOP). In this paper, a lower case f represents a two-valued logic function, an upper case F represents a three-valued input function, and the script $\mathcal{F}$ represents a logical expression. F is a mapping $T^n \to B$, where $T = \{0,1,2\}$ and $B = \{0,1\}$. Let $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, $\alpha_i \in T$ be a ternary vector. Then,

$$x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n} \qquad (1)$$

represents a product of an n-variable function, where

$$x^{\alpha} = \begin{cases} \bar{x} \text{ when } \alpha = 0, \\ x \text{ when } \alpha = 1, \\ 1 \text{ when } \alpha = 2. \end{cases}$$

$F(\boldsymbol{\alpha}) = 1$ iff the product is in the SOP. For example, $\boldsymbol{\alpha} = (1,0,2)$ corresponds to the product $x_1^1 x_2^0 x_3^2$ or $x_1\bar{x}_2$. There are 3^n different products, and the form (1) can represent any one.

For example, consider the SOP $\mathcal{F} = \bar{x}_1 x_3 \vee x_1\bar{x}_2 \vee \bar{x}_2 x_3$. To represent the set of products, we can use the following array of cubes:

x_1	x_2	x_3
0	2	1
1	0	2
2	0	1

In this array, 021 denotes $\bar{x}_1 x_3$, 102 denotes $x_1\bar{x}_2$, and 201 denotes $\bar{x}_2 x_3$. The complete ternary tree in Fig. 10 with the terminal values in the row of SOP in Table 1 represents F. There are 3^3 terminals each of which can be 0 or 1. A 1 shows that the product associated with a path from the root node to a terminal node.

In the case of the above example, there are three terminal nodes labeled 1: f_7, f_{11} and f_{19}. After reduction, we have the ROTDD in Fig. 11. Here only 1-paths are shown, i.e., the 0 terminal nodes are omitted. This RO SOP_TDD has only three 1-paths, and each 1-path corresponds to a product in the SOP. In general, we can say the following: For a function f, an RO SOP_TDD is not unique, since, in general, more than one SOPs exist for a function. However, for an SOP $\mathcal{F}$, the RO SOP_TDD is unique. Given an SOP $\mathcal{F} = \bar{x}\mathcal{F}_0 \vee x\mathcal{F}_1 \vee 1\mathcal{F}_2$, the SOP_TDD is constructed as shown in Fig. 12: The sub-graphs for $\mathcal{F}_0$, $\mathcal{F}_1$, and $\mathcal{F}_2$ are SOP_TDDs for $\mathcal{F}_0$, $\mathcal{F}_1$, and $\mathcal{F}_2$, respectively.

2.4 ESOP_TDDs

An ESOP_TDD represents a set of products in an ESOP. ESOPs are products combined with EXOR operator. On the average, ESOPs require fewer products than SOPs [44, 51]. An ESOP_TDD is similar in concept to an SOP_TDD. An ESOP_TDD represents a mapping F: $T^n \to B$, $F(\boldsymbol{\alpha}) = 1$ iff the product $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ is in the ESOP. Consider the ESOP:

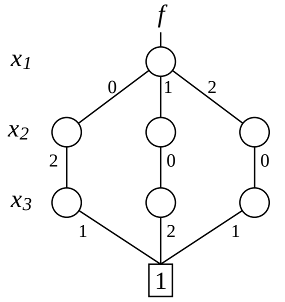


Figure 11. RO SOP_TDD.

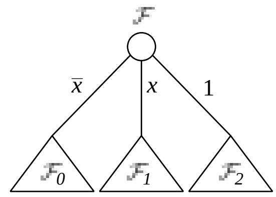


Figure 12. SOP_TDD and ESOP_TDD.

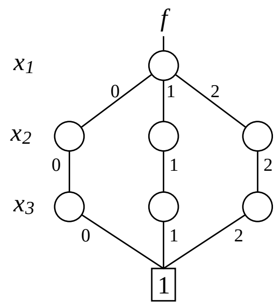


Figure 13. RO ESOP_TDD.

$\mathcal{F} = \bar{x}_1\bar{x}_2\bar{x}_3 \oplus x_1x_2x_3 \oplus 1$. This ESOP is represented by the set of cubes:

x_1	x_2	x_3
0	0	0
1	1	1
2	2	2

000 denotes $\bar{x}_1\bar{x}_2\bar{x}_3$, 111 denotes $x_1x_2x_3$, and 222 denotes the constant 1.

The tree in Fig. 10 with the terminal values in the row of ESOP in Table 1 represents F. It has three non-zero terminals: f_0, f_{13}, and f_{26}. Fig. 13 is the RO ESOP_TDD, where only 1-paths are shown. There are three 1-paths, and each 1-path corresponds to a product in the ESOP. In general, we can say the following: For a given function, the RO ESOP_TDD is not unique, since many ESOPs may exist for a function. However, for a given ESOP, the RO SOP_TDD is unique. Given an ESOP $\mathcal{F} = \bar{x}\mathcal{F}_0 \oplus x\mathcal{F}_1 \oplus 1\mathcal{F}_2$, ESOP_TDD is constructed as shown in Fig. 12: The sub-graphs for $\mathcal{F}_0$, $\mathcal{F}_1$, and $\mathcal{F}_2$ are ESOP_TDDs for $\mathcal{F}_0$, $\mathcal{F}_1$, and $\mathcal{F}_2$, respectively.

2.5 AND_TDDs

An AND_TDD represents the set of all the implicants of a two-valued logic function. An AND_TDD represents a mapping F: $T^n \to B$, where $F(\boldsymbol{\alpha}) = 1$ iff the product $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ is an implicant of f. Since

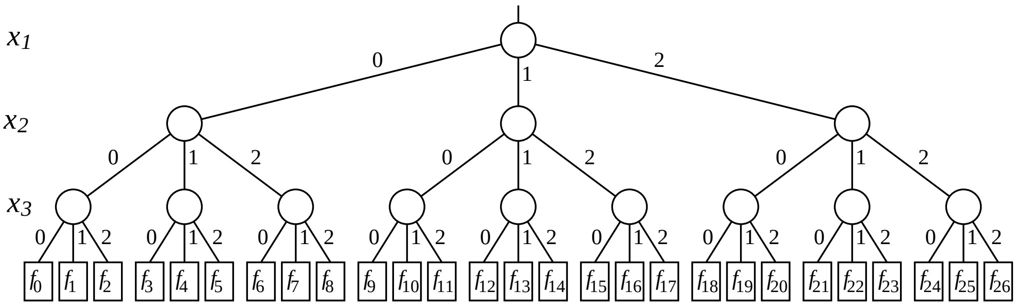


Figure 10. Complete ternary tree for $n = 3$.

Table 1. Functions represented by F.

x_1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2
x_2	0	0	0	1	1	1	2	2	2	0	0	0	1	1	1	2	2	2	0	0	0	1	1	1	2	2	2
x_3	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2
	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}	f_{21}	f_{22}	f_{23}	f_{24}	f_{25}	f_{26}
SOP	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
ESOP	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
AND	0	1	0	1	1	1	0	1	0	1	1	1	1	0	0	1	0	0	0	1	0	1	0	0	0	0	0
Prime	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0
EXOR	0	1	1	1	1	0	1	0	1	1	1	0	1	0	1	0	1	1	1	0	1	0	1	1	1	1	0
Kleene	0	1	2	1	1	1	2	1	2	1	1	1	1	0	2	1	2	2	2	1	2	1	2	2	2	2	2

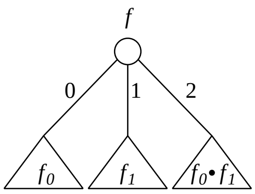


Figure 14. AND_TDD.

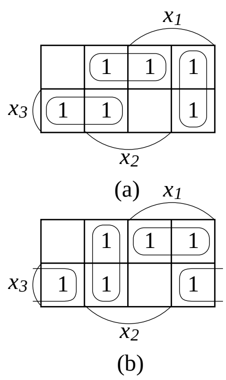


Figure 15. Prime implicants

an AND_TDD represents the set of all the implicants, it is a special case of an SOP_TDD. The RO AND_TDD is unique for f. An AND_TDD is constructed as shown in Fig. 14. Here the rightmost sub-graph represents the logical AND function of f_0 and f_1. An AND_TDD is used to produce a prime_TDD, which is explained later. For example, consider the function of three variables in Fig. 3. There are 6 minterms (Fig. 7(a)), and 6 prime implicants (Fig. 15(a) and (b)). So, in total, there are 12 implicants. The tree in Fig. 10 and terminal values in the row of AND in Table 1 show the set of all the implicants. Fig. 16 shows the Quasi-Reduced AND_TDD (QR AND_TDD). A QR TDD is obtained from a complete ternary tree by using only the reduction Rule 1. In the QR TDD of an n-variable function, all the paths from the root node to the terminal nodes visit exactly n non-terminal nodes. In an AND_TDD, each 1-path corresponds to an implicant of f. In general, RO AND_TDDs do not represent all the implicants, while the QR AND_TDDs represent all the implicants.

2.6 Prime_TDDs

A prime_TDD represents all the prime implicants (PIs) [4, 13, 39, 42] of a two-valued logic function f. A prime_TDD represents F: $T^n \rightarrow B$, where $F(\boldsymbol{\alpha}) = 1$ iff the product $x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$ is a PI of f. A prime_TDD is a special case of SOP_TDDs and is unique for each f. By using prime_TDDs, we can efficiently generate all the PIs. The prime_TDD can be derived from the AND_TDD. Consider the function in Fig. 3. There are 6 PIs as shown in Fig. 15(a) and (b). The tree in Fig. 10 with the terminal values in the row of Prime in Table 1 shows the set of PIs. For example, the path for (012) reaches to a constant 1. This shows that $\bar{x}_1 x_2$ is a PI of the function. Fig. 17 shows the RO prime_TDD with all 0-paths omitted. There are 6 paths from the root node to the constant node, and each corresponds to a PI.

2.7 EXOR_TDDs

An EXOR_TDD represents the extended truth vector of a two-valued logic function f. The extended truth vector $EXT(f : \boldsymbol{\alpha})$ for an n-variable function consists of 3^n elements, and is useful for optimization of AND-EXOR expressions [9, 12, 44, 46, 48, 54, 58]. An EXOR_TDD represents a mapping $F : T^N \rightarrow B$, $F(\boldsymbol{\alpha}) = 1$ iff $EXT(f : \boldsymbol{\alpha}) = 1$. The RO EXOR_TDD

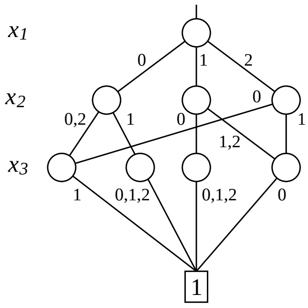


Figure 16.
QR AND_TDD.

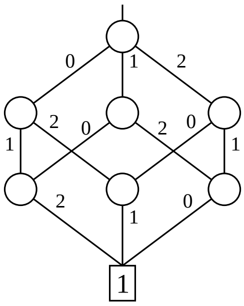


Figure 17.
RO Prime_TDD.

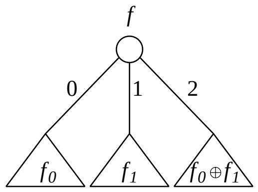


Figure 18.
EXOR_TDD.

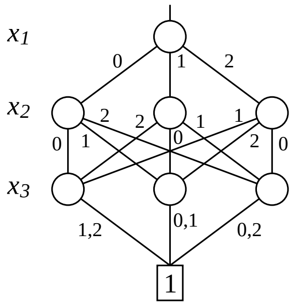


Figure 19.
RO EXOR_TDD.

is unique for f. The EXOR_TDD is constructed as shown in Fig. 18, where the rightmost sub-graph represents the EXOR of f_0 and f_1. The tree in Fig. 10 with the terminal values in the row of EXOR in Table 1 shows the extended truth vector for the three-variable function. Fig. 19 shows the RO EXOR_TDD for the three-variable function.

2.8 Kleene_TDDs

A Kleene_TDD represents the *Kleene function* F: $T^n \to T$, $T = \{0, 1, 2\}$ of a two-valued logic function f: $B^n \to B$. Let $\boldsymbol{\alpha}$ be a ternary vector. $A(\boldsymbol{\alpha})$ is a set of all the binary vectors that are obtained by replacing all the 2's with 0 or 1.

$$F(\boldsymbol{\alpha}) = \begin{cases} 0 \text{ if } f(A(\boldsymbol{\alpha})) = \{0\} \\ 1 \text{ if } f(A(\boldsymbol{\alpha})) = \{1\} \\ 2 \text{ if } f(A(\boldsymbol{\alpha})) = \{0, 1\} \end{cases}$$

In other words, if all the vectors in $A(\boldsymbol{\alpha})$ are mapped to 0, then $F(\boldsymbol{\alpha}) = 0$; if all the vectors are mapped to 1, then $F(\boldsymbol{\alpha}) = 1$; and if some vectors are mapped to 0 and others are mapped to 1, then $F(\boldsymbol{\alpha}) = 2$. In this case, 2 denotes unknown input values or output values, and is often represented by u (unknown). The Kleene function represents the behavior of logic function in the presence of unknown input values. For a given two-valued logic function, the Kleene function is unique.

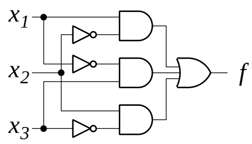


Figure 20. AND-OR network for Fig. 15(a)

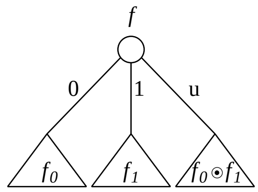


Figure 21. Kleene_TDD.

To explain an application of Kleene_TDDs, consider the three-variable function in Fig. 3. When $(x_1, x_2, x_3) = (0, 0, 0)$, the value of f is zero. When $(x_1, x_2, x_3) = (1, 0, u)$, the value of f is one, since $x_1\bar{x}_2$ is an implicant. However, when $(x_1, x_2, x_3) = (1, 1, u)$, the value of f is u. When $(x_1, x_2, x_3) = (1, u, 0)$, the value of f is 1. A problem arises when we apply a naive ternary logic simulator to the circuit in Fig. 20. For the input $(x_1, x_2, x_3) = (1, u, 0)$, a naive logic simulator produces u at the output; the correct output is 1. However, if we use the Kleene function, such a problem will not occur. A Kleene_TDD is easy to construct as shown in Fig. 21. The rightmost sub-graph represents the alignment of f_0 and f_1, where

$$alignment(x, y) = \begin{cases} x \text{ if } x = y \\ u \text{ if } x \neq y. \end{cases}$$

Alignment is the 3-valued operator defined by Kleene[27]. The tree in Fig. 10 with the terminal values in the row of Kleene in Table 1 shows the Kleene function. This is the only TDD that has three different terminal nodes $\{0, 1, u\}$.

2.9 Relations among TDDs

Table 2 compares the properties of various DDs. The last column shows whether the DD is canonical or not. SOP_TDDs and ESOP_TDDs are not canonical, since many expressions may exist for a function.

1-paths of the Kleene_TDD and the AND_TDD represent sets of all the implicants. 2-paths of the Kleene_TDD represent the set of input values that make unknown output values. 1-paths of the prime_TDD represent the set of all the PIs.

3 Complexity of TDDs

Even if the TDDs have useful properties, they become impractical when they are too large to construct. A complexity analysis in this section reveals the limit of the approach.

3.1 Theoretical analysis

For the DSOP $\mathcal{F}$ represented by the BDD for f, consider an SOP_TDD and an ESOP_TDD that represent $\mathcal{F}$. In this case, the size of the SOP_TDD and the ESOP_TDD are not greater than that of BDD. Thus, the sizes of minimum SOP_TDD and ESOP_TDD are not greater than that of BDD. Table 3 compares the

Table 2. Comparison of various DDs.

	1-paths represent	Applications	
ROBDD	Disjoint SOP	Representation of logic functions	C
QR AND_TDD	Complete sum-of-implicants	Generation of implicants	C
Prime_TDD	Complete sum-of-prime implicants	Generation of prime implicants	C
QR Kleene_TDD	Set of implicants	Logic simulation in the presence of unknown inputs	C
SOP_TDD	SOP	Representation of SOPs	N
ESOP_TDD	ESOP	Representation of ESOPs	N

C: Canonical
N: Non-canonical

Table 3. Size of DDs.

	BDD	TDD type	
		AND, EXOR Prime, Kleene	SOP, ESOP
General function	$O(2^n/n)$	$O(3^n/n)$	$O(2^n/n)$
Symmetric function	$O(n^2)$	$O(n^3)$	$O(n^2)$

size of BDDs and TDDs. $N(BDD : f)$ denotes the size of the BDD for the function f. For TDDs, similar notations are used. As for the size of selected DDs, we have the following:

Theorem 3.1

$$N(BDD : f) = N(BDD : \bar{f}),$$
$$N(EXOR\_TDD : f) = N(EXOR\_TDD : \bar{f}),$$
$$N(Kleene\_TDD : F) = N(Kleene\_TDD : \bar{F}).$$

However, in general,

$$N(AND\_TDD : f) \neq N(AND\_TDD : \bar{f}),$$
$$N(prime\_TDD : f) \neq N(prime\_TDD : \bar{f}).$$

Theorem 3.2

$$\begin{aligned} N(BDD : f) &\leq N(AND\_TDD : f) \\ &\leq N(EXOR\_TDD : f) \\ &\leq N(Kleene\_TDD : f) \\ N(AND\_TDD : f) &\leq N(Kleen\_TDD : f). \end{aligned}$$

Theorem 3.3 *If f is a parity function, then*

$$\begin{aligned} N(BDD : f) &= N(AND\_TDD : f) \\ &= N(EXOR\_TDD : f) \\ &= N(Kleene\_TDD : f) \\ &= N(prime\_TDD : f). \end{aligned}$$

If f is a unate function, then

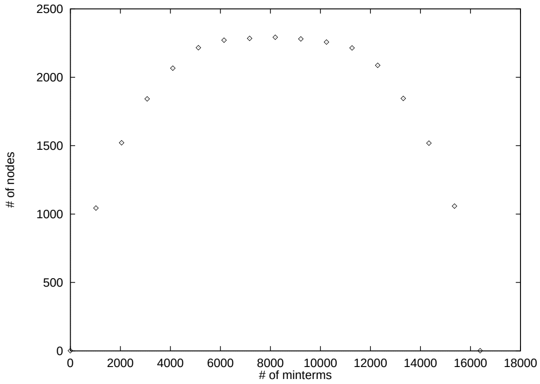
$$N(BDD : f) = N(AND\_TDD : f).$$

For most functions f, $N(BDD : f) < N(prime\_TDD : f)$. However, for some cases, $N(BDD : f) > N(prime\_TDD : f)$.

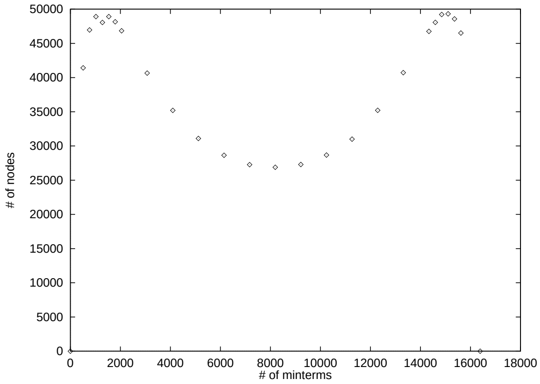
Example 3.1 *For $f = x_1x_{p+1} \vee x_2x_{p+2} \vee \cdots \vee x_px_{2p}$, $N(BDD : f) = 2^{p+1}$, but $N(prime\_TDD : f) = 2p^2 + 2$.* *(End of Example)*

3.2 Experimental results

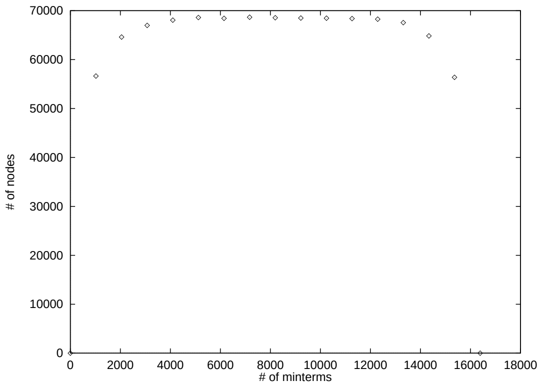
Fig. 22(a), (b), and (c) show the sizes of BDDs, Kleene_TDDs and EXOR_TDDs for randomly generated functions with 14 variables, respectively. For a given number of true minterms, we generated one logic function randomly. In each graph, the horizontal axis denotes the number of true minterms of f, and the vertical axis denotes the size of DDs. The graphs are approximately symmetric with respect to the center, which is supported by Theorem 3.1. For BDDs and EXOR_TDDs, size is largest when the number of true minterms is 2^{n-1}. On the other hand, Kleene_TDDs has a local minimum when the number of true minterms is 2^{n-1}, and have their maximum sizes for two points, either side of the central minimum. This is a very interesting property of Kleene_TDDs. Fig. 23(a) and (b) show the sizes of AND_TDDs and prime_TDDs for randomly generated functions with 14 variables, respectively. In these cases, the plots are not symmetric with respect to the center lines. The sizes of these TDDs take their maximum when the number of true minterms is near to 2^n. It is known that the number of implicants or PIs reaches their maximum value when the number of true minterms is near to 2^n [7, 33]. Thus, the sizes of TDDs are large for these points. Table 4 compares the sizes for various DDs of benchmark functions [66]. As shown in Theorem 3.2, BDDs are not larger than AND_TDDs, EXOR_TDDs, and Kleene_TDDs. However, prime_TDDs can be smaller than corresponding BDDs. For example, the prime_TDD for apex2 is smaller than the corresponding BDD. For 9sym and rd84, which are symmetric functions, the DDs are relatively small. For xor5, which is a parity function, the sizes of all the DDs are the same, which is verified by Theorem 3.3. Note that BDDs, AND_TDDs, EXOR_TDDs, and Kleene_TDDs are represented as shared DD [32], while prime_TDDs are represented as multi-terminal DDs [6]. Also, the sizes of prime_TDDs include the constant nodes, while sizes of other DDs do not. Orderings of the input variables were obtained by a heuristic algorithm that reduces the sizes of BDDs.



(a) BDD



(b) Kleene_TDD



(c) EXOR_TDD

Figure 22. Size of BDDs and TDDs.

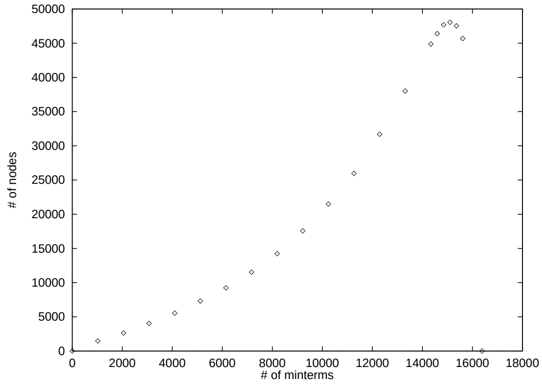
4 Ongoing Research

4.1 SOP_TDD

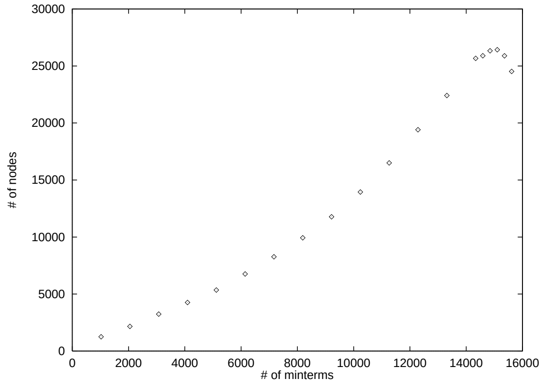
For a given function f, there exists an SOP_TDD, which is not greater than the corresponding BDD. For many functions, we can generate SOP_TDDs that are smaller than the corresponding BDDs. Table. 5 compares the sizes of BDDs and SOP_TDDs, where the orderings of the input variables are not optimized.

4.2 Kleene_TDD

We have developed a Kleene_TDD package. Unfortunately, Kleene_TDDs are much larger than cor-



(a) AND_TDD



(b) Prime_TDD

Figure 23. Size of TDDs.

Table 4. Size of various DDs.

function	in	out	BDD	AND TDD	EXOR TDD	Kleene TDD	Prime TDD
9sym	9	1	33	60	70	94	62
apex1	45	45	1332	6249	47814	18401	15210
apex2	39	3	410	542	3575	3500	215
apex3	54	50	935	3161	34574	7119	—
apex5	117	88	1078	3039	3282	4204	—
cordic	23	2	75	83	153	271	210
cps	24	109	987	1457	4808	5653	—
duke2	22	29	336	522	2176	2555	3800
e64	65	65	1379	1379	1379	2693	1371
ex5	8	63	278	381	444	628	3599
misex1	8	7	36	45	70	92	64
misex2	25	18	81	81	138	204	253
misex3	14	14	542	1219	3644	3262	10632
pdc	16	40	560	1024	2321	3031	48837
rd84	8	4	59	79	72	121	87
sao2	10	4	85	114	216	305	170
seq	41	35	1248	3873	67414	18745	—
spla	16	46	581	717	2237	2315	30024
t481	16	1	32	48	43	66	357
vg2	25	8	194	399	865	961	1340
xor5	5	1	9	9	9	9	11
z5xp1	7	10	68	79	75	158	721

— represents memory over flow.

Table 5. Size of BDDs and SOP_TDDs.

	in	out	BDD	SOP_TDD
accpla	50	69	5632	1566
apex1	45	45	5024	1433
apex2	39	3	652	311
b2	16	17	4472	939
clip	9	5	260	196
duke2	22	29	1006	523
ex4	128	28	1319	679
in1	16	17	4472	939
in2	19	10	2405	414
in4	32	20	1310	535
in6	33	23	540	299
mainpla	27	54	3360	2694
misex3	14	14	1316	907
p1	8	18	354	193
signet	39	8	2965	293
ti	47	72	6260	828
tial	14	8	1363	964
ts10	22	16	4408	183

responding BDDs. We can decompose a logic function into two such that the corresponding decomposition of Kleene function will produce the correct result as shown below [18] .

Theorem 4.4 *Let a function f be represented as $f(X) = h(g(X_1), X_2)$. F, the Kleene function for f, is represented as $F(X_1, X_2) = H(G(X_1), X_2)$, where G and H are Kleene functions for g and h, respectively.*

A bi-decomposition is a special case of a functional decomposition, having form $f(X) = h(g_1(X_1), g_2(X_2))$. The detection of a bi-decomposition is quite easy [57].

5 Various Works on TDDs

Higuchi-Kameyama [16, 23, 24] considered the realization of ternary logic functions $F\colon T^n \rightarrow T$ by using T-gates. A T-gate [28] is a three-valued multiplexer, and corresponds to a node in a general_TDD. They showed optimization methods for ROTDDs and free TDDs to simplify T-gate networks. In the free-TDDs, ordering of input variables may be different for each path.

Thayse-Davio-Deschamps [63] presented the concept of MDDs in 1978, calling them "multiple-valued decision algorithms." They used MDDs to realize multi-valued logic function using multiplexers, to realize sequential circuits using multiple-valued ROMs and multiplexers, and to transform and to optimize microprograms.

Mukaidono [37] defined B-ternary logic function, which is the same as the Kleene function. He found a canonical representation of Kleene functions. Later, he used the Kleene functions for evaluation of logic functions in the presence of unknown input values [38].

Papakonstantinou [40] used EXOR ternary decision trees to minimize ESOPs with up to four variables.

Srinivasan-Kam-Malik-Brayton [62] developed algorithm to manipulate MDDs. They also showed applications of MDDs to channel and switch box routing as well as hardware resource scheduling.

Sasao [45, 50] used EXOR_TDDs to minimize pseudo-Kronecker expression, a class of AND-EXOR two-level expressions. Prime_TDDs were used to generate all the prime implicants [49, 61]. This method is much faster than conventional ones [64, 43], although yet faster methods exist [8]. The program in [49] generated thousands of PIs within 10 seconds. At the same time, he presented the concept of AND_TDDs and SOP_TDDs, and analyzed their complexities. Later his group successfully minimized FPRM, a class of AND-EXOR two-level logic expressions, with more than 90 input variables by using EXOR_TDDs [60].

Heap-Rogers-Mercer [15] used EXOR_TDD to simplify ESOPs. Miller [34] implemented an MDD reduction algorithm, where he considered "unary cycling operations" to simplify MDDs [35]. McGeer-McMillan-Saldanha-Sangiovanni-Vincentelli-Scaglia [31] used MDD in cycle-based logic simulation. They grouped k binary input variables to form a single 2^k-valued variable. By this, the number of memory access to evaluate an MDD is reduced by a factor of k. They showed that BDD-based logic simulation is much faster than conventional ones. The extensions are in [14, 56].

Yasuoka [67, 68] developed algorithms for manipulating ESOP_TDDs. Miller-Thomson-Bradbeer [36] used ESOP_TDDs to implement multi-level networks. Jennings [20] presented a Kleene_TDD for logic simulation in the presence of unknown input values. The extensions are in [19, 21, 30, 22]. Perkowski-Chrzanowska-Jeske-Sarabi-Schafer [41] defined various canonical and non-canonical TDDs. Their work shows many other different TDDs exist. Kamiura-Satoh-Hata-Yamato [25, 26] used general_TDDs to implement ternary cellular arrays. Iguchi-Sasao-Matsuura [17] compared the complexities of Kleene_TDDs with BDDs, AND_TDDs and EXOR_TDDs.

Acknowledgments

This work was supported in part by a Grant in Aid for Scientific Research of the Ministry of Education, Science Culture and Sports of Japan. Prof. J. T. Butler carefully reviewed the manuscript. Mr. Matsuura worked for drawing pictures and edited the Latex files.

References

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

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

[3] 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.

[4] F. M. Brown, *Boolean Reasoning: The logic of Boolean equations*, Kluwer Academic Publishers, Boston, 1990.

[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] E. M. Clarke, M. Fujita, and X. Zhao, "Multi-terminal binary decision diagrams and hybrid decision diagrams," Chapter 4 in [54].

[7] A. Cobham, R. Fridshal, and J. H. North, "A statistical study of the minimization of Boolean functions using integer linear programming," *IBM Research Report RC-756*, June 1962.

[8] O. Coudert, "Implicit and incremental computation of primes and essential primes of Boolean functions," in *Proc. of DAC'92*, June 1992. Also, in Sasao (ed.), *Logic Synthesis and Optimization*, Kluwer Academic Publisher, pp. 33-57, 1992.

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

[10] R. Drechsler and B. Becker, "OKFDD: Algorithms, applications and extensions," Chapter 7 in [54].

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

[13] M. A. Harrison, *Introduction to Switching and Automata Theory*, McGraw-Hill, 1965.

[14] Hafiz Md. Hasan Babu and T. Sasao, "A method to represent multiple-output switching functions by using binary decision diagrams," *the Sixth Workshop on Synthesis And System Integration of MIxed Technologies (SASIMI'96)*, Fukuoka, Japan, Nov. 1996, pp. 212-217.

[15] M. A. Heap, W. A. Rogers, and M. R. Mercer, "A synthesis algorithm for two-level XOR based circuits," *IEEE 1992 International Conference on Computer Design: VLSI in Computers and Processors. ICCD '92*, pp. 459-462, 1992.

[16] T. Higuchi and M. Kameyama, "Ternary logic system based on T-gate," *Proc. International Symposium on Multiple-valued logic*, pp. 290-304, May 1975.

[17] Y. Iguchi, T. Sasao and M. Matsuura, "On properties of Kleene TDDs," *Asia and South Pacific Design Automation Conference, ASP-DAC'97*, Jan. 1997, pp. 473-476.

[18] Y. Iguchi, T. Sasao and M. Matsuura, "A logic simulation by using Kleene_TDD and its evaluation," (in Japanese), *FTC Workshop*, Jan. 23, 1997.

[19] G. Jennings, J. Isaksson, and P. Lindgren, "Ordered ternary decision diagrams and the multivalued compiled simulation of unmapped logic," *Proc. IEEE 27th Annual Simulation Symposium*, pp. 99-105, 1994

[20] G. Jennings, "Symbolic incompletely specified functions for correct evaluation in the presence of indeterminate input values," *Proceedings of the 28th Hawaii International Conference on System Sciences*, pp. 23-31 Vol. 1, 1995.

[21] G. Jennings, "Accurate ternary-valued compiled logic simulation of complex logic networks by OTDD composition," *Proceedings of the 28th Annual Simulation Symposium*, 303-310, 1995.

[22] G. Jennings, "Using redundancy to side-step the OBDD variable ordering problem: K-feasible decomposition by Kleenean-strong ternary decision diagram," *the Sixth Workshop on Synthesis And System Integration of MIxed Technologies (SASIMI'96)*, Fukuoka, Japan, Nov. 25-26, 1996, pp. 205-211.

[23] M. Kameyama and T. Higuchi, "Synthesis of multiple-valued logic based on tree-type universal logic module," *IEEE Trans. Comput.*, Vol. C-26, pp. 1297-1302, Dec. 1977.

[24] M. Kameyama and T. Higuchi, "Synthesis of optimal T-gate networks in multiple-valued logic," *Proc. International Symposium on Multiple-valued logic*, pp. 190-195, May 1979.

[25] N. Kamiura, H. Satoh, Y. Hata, and K. Yamato, "Design in fault isolating of ternary cellular arrays using ternary decision diagrams," *Proceedings of the IEEE Third Asian Test Symposium*, pp. 201-206, 1994.

[26] N. Kamiura, H. Satoh, Y. Hata, and K. Yamato, "On ternary cellular arrays designed from ternary decision diagrams," *IEICE Trans. Inf. Syst.* (Japan), Vol. E78-D, No. 4, 326-335, April 1995.

[27] S. C. Kleene, *Introduction to Metamathematics*, Wolters-Noordhoff, North-Holland Publishing, 1952.

[28] C. Y. Lee and W. H. Chen, "Several-valued logic combinational switching circuits," *Trans. AIEE*, Vol. 75, No. 1, 1956, pp. 278-283.

[29] H-T. Liaw and C-S. Lin, "On the OBDD representation of generalized Boolean functions," *IEEE Transactions on Comput.*, Vol. 41, No. 6, June 1992, pp. 661-664.

[30] P. Lindgren, "Improved computational methods and lazy evaluation of the ordered ternary decision diagram," *Asia and South Pacific Design Automation Conference*, pp. 379-384, Aug. 1995.

[31] P. C. McGeer, K. L. McMillan, A. Saldanha, A. L. Sangiovanni-Vincentelli, P. Scaglia, "Fast discrete function evaluation using decision diagrams," *International Workshop on Logic Synthesis*, Lake Tahoe, May, 1995, pp. 6_1-6_9. Also in *International Conf. on Computer Aided Design*, Nov. 1995, pp. 402-407.

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

[33] F. Mileto and G. Putzolu, "Average values of quantities appearing in Boolean function minimization," *IEEE Trans. Elec. Comput.*, Vol. EC-13, No. 4, April 1964, pp. 87-92.

[34] D. M. Miller, "Multiple-valued logic design tools," *Proc. of International Symposium on Multiple Valued Logic*, May 1993, pp. 2-11.

[35] D. M. Miller, "Spectral transformation of multiple-valued decision diagrams," *Proc. 24th International Symposium on Multiple-Valued Logic*, Boston, pp. 89-96, May 1994.

[36] J. F. Miller, P. Thomson, P. V. G. Bradbeer, "Ternary decision diagram optimisation of Reed-Muller logic functions using a genetic algorithm for variable and simplification rule ordering," *Evolutionary Computing. AISB Workshop. Selected Papers*, 181-90, 1995

[37] M. Mukaidono, "On the B-ternay logic function: A ternary logic considering ambiguity," *Trans. IECE Japan*, (in Japanese), Vol. 55-D, No. 6, pp. 355-362, June 1972.

[38] M. Mukaidono, "Evaluation methods of logic functions for uncertain input values," (in Japanese), *Information Processing Soceity of Japan*, DA-18, 1983.

[39] S. Muroga, *Logic Design and Switching Theory*, John Wiley & Sons, 1979.

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

[41] M. A. Perkowski, M. Chrzanowska-Jeske, A. Sarabi, I. Schafer, "Multi-level logic synthesis based on Kronecker decision diagrams and Boolean ternary decision diagrams for incompletely specified functions," *VLSI Des. (Switzerland)*, Vol. 3, No. 3-4, 301-313, 1995.

[42] W. V. Quine, "A way to simplify truth functions," *Amer. Math. Mon.*, Vol. 62, Nov. 1955, pp. 627-631.

[43] B. Reusch, "Generation of prime implicant from subfunctions and a unifying approach to the covering problem," *IEEE Trans. Comput.*, Vol. C-24, No. 9, Sept. 1975, pp. 924-930.

[44] 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.

[45] T. Sasao, "Optimization of multiple-valued AND-EXOR expressions using multiple-place decision diagrams," *ISMVL-92*, May 1992, pp. 451-458. Also, in Sasao (ed.), *Logic Synthesis and Optimization*, Kluwer Academic Publisher, pp. 33-57, 1992.

[46] T. Sasao (ed.), *Logic Synthesis and Optimization*, Kluwer Academic Publishers (1993).

[47] T. Sasao, "Logic synthesis using EXOR logic gates," Chapter 12 in [46].

[48] T. Sasao, "AND-EXOR expressions and their optimization," Chapter 13 in [46].

[49] T. Sasao, "Ternary decision diagram and their applications," *International Workshop on Logic Synthesis*, Lake Tahoe, May 1993, pp. 6C-1–6C-11.

[50] T. Sasao, "Optimization of pseudo-Kronecker expressions using multiple-place decision diagrams," *IEICE Transactions on Information and Systems*, Vol. E76-D, No. 5, May 1993, pp. 562-570.

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

[52] T. Sasao and J. T. Butler, "A design method for look-up table type FPGA by pseudo-Kronecker expansion," *Proc. of International Symposium on Multiple Valued Logic*, Boston, MA, May 1994, pp. 97-106.

[53] T. Sasao and J. T. Butler, "Planar multiple-valued decision diagrams," *Proc. of International Symposium on Multiple Valued Logic*, Bloomington, Indiana, May 23-25, 1995, pp. 28-35.

[54] T. Sasao and M. Fujita (ed.), *Representations of Discrete Functions*, Kluwer Academic Publishers (1996).

[55] T. Sasao and J. T. Butler, "Planar multiple-valued decision diagrams," *Multiple-valued Journal*, Vol. 1, No. 1, pp. 39-64, 1996.

[56] T. Sasao and J. T. Butler, "A method to represent multiple-output switching functions by using multi-valued decision diagrams," *IEEE International Symposium on Multiple-Valued Logic*, Santiago de Compostela, Spain, May 29-31, 1996, pp. 248-254.

[57] T. Sasao and J. T. Butler, "Bi-decompositions of logic functions and their applications," *IPSJ SIG Note*, Vol. 96, Information Processing Society of Japan, Dec. 1996.

[58] T. Sasao, "Representation of logic functions using EXOR operators," Chapter 2 in [54].

[59] S. Stankovic, T. Sasao, and C. Moraga, "Spectral transforms decision diagrams," Chapter 3 in [54].

[60] T. Sasao and F. Izuhara, "Exact minimization of FPRMs using multi-terminal EXOR TDDs," Chapter 8 in [54].

[61] T. Sasao, "Ternary decision diagrams and their applications," Chapter 12 in [54].

[62] A. Srinivasan, T. Kam, S. Malik, and R. K. Brayton, "Algorithm for discrete functions manipulation," *Proc. ICCAD-90*, pp. 92-95, Nov. 11-15, 1990, Stanta Clara, CA.

[63] A. Thayse, M. Davio, and J.-P. Deschamps, "Optimization of multiple-valued decision diagrams," *ISMVL-79*, Rosemont, IL. May 1978, pp. 171-177.

[64] P. Tison, "Generalization of consensus theory and application to the minimization of Boolean functions," *IEEE Trans. Electron Comput.*, Vol. EC-16, August 1967, pp. 446-456.

[65] I. Wegener, "The size of reduced OBDD's and optimal read-once branching program for almost all Boolean functions," *IEEE Trans. on Comput.*, Vol. C-43, No. 11, pp. 1262-1269, Nov. 1994.

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

[67] K. Yasuoka, "Ternary decision diagrams to represent ringsum-of-products forms," *IFIP WG. 10.5 Workshop on Applications of the Reed-Muller Expansions in Circuit Design*, August 1995.

[68] K. Yasuoka, "Ternary decision diagrams," *Proc. SPIE - Int. Soc. Opt. Eng.*, Vol. 2644, pp. 489-496, 1996.