PAPER Special Section on VLSI Design and CAD Algorithms

Shared Multi-Terminal Binary Decision Diagrams for Multiple-Output Functions

Hafiz Md. HASAN BABU†, *Nonmember and* Tsutomu SASAO†, *Member*

SUMMARY This paper describes a method to represent m output functions using shared multi-terminal binary decision diagrams (SMTBDDs). The SMTBDD(k) consists of multi-terminal binary decision diagrams (MTBDDs), where each MTBDD represents k output functions. An SMTBDD(k) is the generalization of shared binary decision diagrams (SBDDs) and MTBDDs: for $k = 1$, it is an SBDD, and for $k = m$, it is an MTBDD. The size of a BDD is the total number of nodes. The features of SMTBDD(k)s are: 1) they are often smaller than SBDDs or MTBDDs; and 2) they evaluate k outputs simultaneously. We also propose an algorithm for grouping output functions to reduce the size of SMTBDD(k)s. Experimental results show the compactness of SMTBDD(k)s. An SMTBDD$_{min}$ denotes the smaller SMTBDD which is either an SMTBDD(2) or an SMTBDD(3) with fewer nodes. The average relative sizes for SBDDs, MTBDDs, and SMTBDDs are 1.00, 152.73, and 0.80, respectively.

key words: *binary decision diagram (BDD), multiple-output functions, clique cover, TDM realization, logic simulation*

1. Introduction

Efficient representations of logic functions are very important in logic design. Various methods exist to represent logic functions. Among them, graph-based representations such as BDDs (binary decision diagrams) are extensively used in logic synthesis, test, and verification [1],[4],[12],[13]. In logic simulation, the BDD-based methods offer orders-of-magnitude potential speedup over traditional logic simulation methods [2],[3],[5],[15]. In real life, many practical logic functions are multiple-output [12]. In this paper, we consider three different approaches to represent multiple-output functions using BDDs: shared binary decision diagrams (SBDDs), multi-terminal binary decision diagrams (MTBDDs), and shared multi-terminal binary decision diagrams (SMTBDDs). SMTBDDs are the generalization of SBDDs and MTBDDs. A general structure of an SMTBDD(k) with $k = 3$ is shown in Fig. 1. The evaluation of outputs using an SMTBDD(k) is k times faster than an SBDD, since it evaluates k outputs at the same time. For most functions, SMTBDD(k)s are smaller than the corresponding MTBDDs. In modern LSI, the reduction of the number of pins is not so easy, even though the integration of more gates may be possible. The time-division multiplexing (TDM) realizations of multiple-output networks from SMTBDDs are useful to reduce the number of pins as well as to reduce hardware [9]. SMTBDD(k)s are also helpful for look-up table type FPGA design [6], logic simulation [5],[14],[15], etc. In the previous paper [7], we considered SMTBDD(2)s. In this paper, we consider SMTBDD(3)s. SMTBDD(3)s are smaller than SMTBDD(2)s for many benchmark functions. In addition, the evaluation of outputs using an SMTBDD(3) is faster than an SMTBDD(2). The rest of the paper is organized as follows: Section 2 deals with definitions and properties of multiple-output functions and SMTBDD(k)s. Section 3 proposes an optimization algorithm for SMTBDD(3)s using clique cover, and Sect. 4 shows the experimental results for various benchmark functions.

Manuscript received March 13, 1998.
Manuscript revised June 5, 1998.

†The authors are with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka-shi, 820–8502 Japan.

2. Preliminaries

This section shows definitions and properties of multiple-output functions and SMTBDD(k)s [7],[8], [12]. The results of this section will be used in Sect. 3.

Definition 1: Let $B = \{0, 1\}$. A *multiple-output logic function* f with n input variables, $x_1, \ldots, x_n$, and m output variables $y_1, \ldots, y_m$, is a function

$$f : B^n \rightarrow B^m,$$

where $\boldsymbol{x} = (x_1, \ldots, x_n) \in B^n$ is an *input vector*, and $\boldsymbol{y} = (y_1, \ldots, y_m) \in B^m$ is an *output vector* of f.

Example 1: Table 1 shows a 2-input 6-output function. □

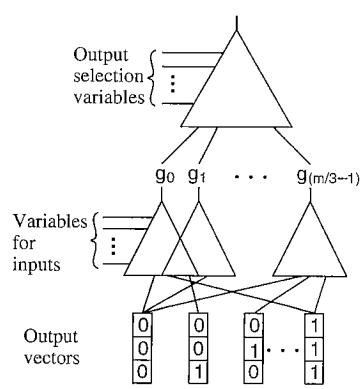


Fig. 1 General structure of an SMTBDD(k) with $k = 3$.

Definition 2: Let $F(\boldsymbol{a}) = (f_0(\boldsymbol{a}), f_1(\boldsymbol{a}), \ldots, f_{m-1}(\boldsymbol{a}))$ be the output vector of m functions for an input $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in B^n$. Two output vectors $F(\boldsymbol{a}_i)$ and $F(\boldsymbol{a}_j)$ are *distinct* iff $F(\boldsymbol{a}_i) \neq F(\boldsymbol{a}_j)$. Let r be the *number of distinct output vectors* in $F(\boldsymbol{a}) = (f_0(\boldsymbol{a}), f_1(\boldsymbol{a}), \ldots, f_{m-1}(\boldsymbol{a}))$.

Example 2: Consider the 2-input 6-output function in Table 1. The distinct output vectors are $(0,1,0,0,1,1)$, $(1,1,1,0,1,0)$, and $(0,1,1,1,0,1)$. Therefore, the number of distinct output vectors is three, i.e. $r = 3$. □

Lemma 1: Let $f_0, f_1, \ldots,$ and f_{m-1} $(f_i \neq 0)$ be disjoint each other, i.e. $f_i \cdot f_j = 0$, and $i \neq j$. Then, the number of distinct output vectors for $F(\boldsymbol{x}) = (f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_{m-1}(\boldsymbol{x}))$ is m or $m+1$.

Proof: Since $f_0, f_1, \ldots,$ and f_{m-1} are disjoint, in the vector $F(\boldsymbol{x}) = (f_0(\boldsymbol{x}), f_1(\boldsymbol{x}), ..., f_{m-1}(\boldsymbol{x}))$ at most one output $f_i(\boldsymbol{x})(i = 0, 1, \ldots m-1)$ is one and others are zero. Therefore, the number of distinct output vectors is at least m. On the other hand, when there is the output vectors with all zero's, the number of distinct output vectors is $m+1$. □

Example 3: Consider the 2-input m-output function in Table 2, where $m = 3$. The distinct output vectors for the functions f_0, f_1, and f_2 are $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$. So, the number of distinct output vectors is m. Now, consider the 2-input 3-output function in Table 3. The distinct output vectors in (f_0, f_1, f_2) are $(1,0,0)$, $(0,1,0)$, $(0,0,1)$, and $(0,0,0)$. In this case, the number of distinct output vectors is $m+1$. □

Definition 3: Let f be a function. The set of input variables on which f depends is the *support* of f, and is denoted by $support(f)$. The *size of the support* is the number of variables in the $support(f)$.

Example 4: Table 1 shows a 2-input 6-output function. An SMTBDD(3) can be constructed with the groupings $[f_0, f_1, f_2]$ and $[f_3, f_4, f_5]$. $support(f_0, f_1, f_2) = \{x_1, x_2\}$, and $support(f_3, f_4, f_5) = \{x_1, x_2\}$. Thus, the sizes of the supports are 2. □

Definition 4: The *size of the BDD*, denoted by size(BDD), is the total number of terminal and non-terminal nodes. In the case of SBDDs and SMTBDD(k)s, the sizes include the nodes for the output selection variables.

Example 5: The size of the SMTBDD(3) in Fig. 3 is 9. □

Table 1 2-input 6-output function.

Input		Output					
x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5
0	0	0	1	0	0	1	1
0	1	0	1	0	0	1	1
1	0	1	1	1	0	1	0
1	1	0	1	1	1	0	1

Table 2 2-input 3-output function with three distinct output vectors.

Input		Output		
x_1	x_2	f_0	f_1	f_2
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	0	1	0

Table 3 2-input 3-output function with four distinct output vectors.

Input		Output		
x_1	x_2	f_0	f_1	f_2
0	0	1	0	0
0	1	0	1	0
1	0	0	0	1
1	1	0	0	0

2.1 Shared Multi-Terminal Binary Decision Diagrams

Shared binary decision diagrams (SBDDs), multi-terminal binary decision diagrams (MTBDDs), and shared multi-terminal binary decision diagrams (SMTBDDs) represent multiple-output functions. SMTBDDs consist of MTBDDs. An SMTBDD(k) is the generalization of SBDDs and MTBDDs: for $k = 1$, it is an SBDD, and for $k = m$, it is an MTBDD, where m is the number of output functions. Figures 2 and 3 show an SMTBDD(2) and an SMTBDD(3) for Table 1, respectively. In Fig. 3, the SMTBDD(3) has two groups: $[f_0, f_1, f_2]$ and $[f_3, f_4, f_5]$, and g_0 is the output selection variable which selects a group of outputs. In this paper, "[ ]" denotes a group of output functions that consists of two or more outputs.

We use the following two techniques to reduce the number of nodes in the SMTBDD(k)s:
Let $[f_i, f_j]$ be a pair of two output functions, where $i \neq j$.

- In general, an MTBDD for two outputs has four terminal nodes $[0,0]$, $[0,1]$, $[1,0]$, and $[1,1]$. However, if $f_i f_j = 0$, then $[1,1]$ never appers as a terminal node in the MTBDD of an SMTBDD(2). Thus, this pairing of output functions tends to produce a

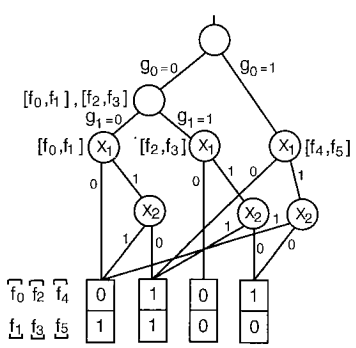


Fig. 2 SMTBDD(2) with groupings $[f_0, f_1]$, $[f_2, f_3]$, and $[f_4, f_5]$ for the functions in Table 1.

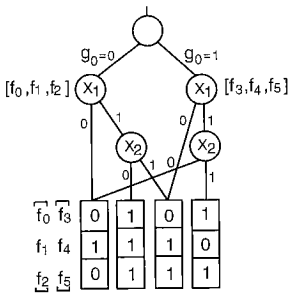


Fig. 3 SMTBDD(3) with groupings $[f_0, f_1, f_2]$ and $[f_3, f_4, f_5]$ for the functions in Table 1.

smaller BDD, since the number of terminal nodes is at most three. Similarly, if $\bar{f}_i f_j = 0$, $f_i \bar{f}_j = 0$, or $\bar{f}_i \bar{f}_j = 0$, then $[f_i, f_j]$ is also a candidate of a pair.

- If $support(f_i) \cap support(f_j) \neq \phi$, then $[f_i, f_j]$ is a candidate of a pair, otherwise, they should be represented by the seperate BDDs.

Note that these two techniques are also applicable to SMTBDD(k)s with $k \geq 3$.

Definition 5: Let an SMTBDD(k) consist of two MTBDDs: MTBDD1 and MTBDD2. MTBDD1 and MTBDD2 are ***disjoint*** iff they do not share any non-terminal nodes each other in the SMTBDD(k).

Example 6: In Fig. 3, there are two disjoint MTBDDs for groupings $[f_0, f_1, f_2]$ and $[f_3, f_4, f_5]$. □

Property 1: Let SMTBDD1 and SMTBDD2 be SMTBDD(k)s. Let SMTBDD1 consist of MTBDD1 and MTBDD2, and let SMTBDD2 consist of MTBDD3 and MTBDD4. If all the MTBDDs are disjoint each other and $size(MTBDD1) = size(MTBDD3)$, and $size(MTBDD2) = size(MTBDD4)$, then $size(SMTBDD1) = size(SMTBDD2)$. □

Lemma 2: The numbers of terminal nodes in an SMTBDD(2) and an SMTBDD(3) are the same iff the numbers of distinct output vectors are also the same.

Proof: Since the number of terminal nodes in an SMTBDD(k) is equal to the number of distinct output vectors, an SMTBDD(2) and an SMTBDD(3) have the same number of terminal nodes iff the numbers of distinct output vectors in the both SMTBDDs are also the same. □

Example 7: Figures 2 and 3 show an SMTBDD(2) and an SMTBDD(3) for the functions in Table 1, respectively. The numbers of terminal nodes in the both SMTBDDs are the same, since the numbers of distinct output vectors are also the same, i.e. 4. □

Lemma 3: All the functions $\{0,1\}^n \rightarrow \{0,1,\ldots,r-1\}$ can be represented by an MTBDD with r^{2^n} nodes.

Proof: No more than r^{2^n} nodes are needed. Else two nodes represent the same function and can be combined. No less than r^{2^n} nodes can be used because there are this many functions. □

The constructive proof of this lemma is shown in [7].

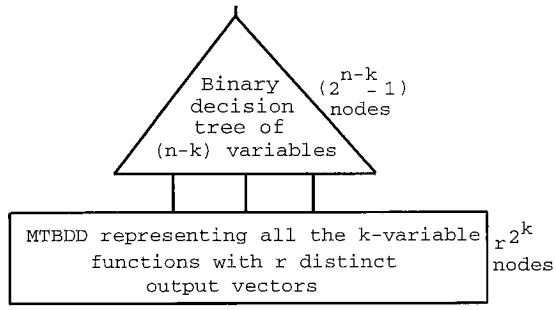


Fig. 4 Representation of an n-input m-output function by an MTBDD.

Theorem 1: Let r be the number of distinct output vectors of an n-input m-output function. Then, the size of the MTBDD can be at most $\min_{k=1}^{n}\{2^{n-k} - 1 + r^{2^k}\}$.

Proof: Consider the MTBDD in Fig. 4, where the upper block is a binary decision tree of $(n-k)$ variables, and the lower block generates all the functions of k or fewer variables. The binary decision tree of $(n-k)$ variables has

$$1 + 2 + 4 + \cdots + 2^{n-k-1} = 2^{n-k} - 1 \text{ nodes.}$$

By Lemma 3, the MTBDD of k-variable functions with r distinct output vectors has r^{2^k} nodes. Thus, the size of the MTBDD can be at most

$$\min_{k=1}^{n}\{2^{n-k} - 1 + r^{2^k}\}. \quad \square$$

Note that the upper bound on the size of the MTBDD is used in Algorithm 1.

Lemma 4: Let m_1 be the total number of groups of m distinct output functions. Then, the number of nodes for the output selection variables in the SMTBDD is $m_1 - 1$.

Example 8: Consider the SMTBDD in Fig. 3, where the total number of groups is two, i.e. $[f_0, f_1, f_2]$ and $[f_3, f_4, f_5]$. Therefore, the number of nodes for the output selection variables in the SMTBDD is one. □

Theorem 2: Let m_1 be the total number of groups of m distinct output functions and $f : \{0,1\}^n \rightarrow \{0,1,\ldots,r-1\}^m$. Then, f can be represented by an SMTBDD with at most $\min_{k=1}^{n}\{m_1 \cdot 2^{n-k} - 1 + r^{2^k}\}$ nodes.

Proof: Consider the mapping $f : \{0,1\}^n \rightarrow \{0,1,\ldots,r-1\}^m$, where r is the number of terminal nodes. In the SMTBDD in Fig. 5, the upper block selects m_1 groups for m output functions, the middle block constitutes binary decision trees of $(n-k)$ variables, and the lower block generates all the functions of k variables by an MTBDD with r terminal nodes. By Lemma 4, the upper block requires $(m_1 - 1)$ nodes to select m_1 groups. By Lemma 3, the lower block requires r^{2^k} nodes. Now, we consider the middle block. Each binary decision tree of $(n-k)$ variables has

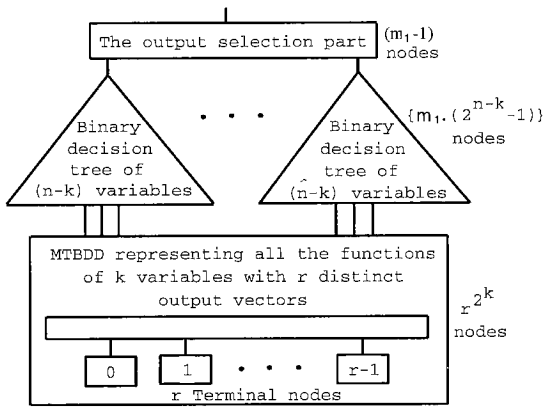


Fig. 5 Representation of an n-input m-output function by an SMTBDD.

$$1+2+4+\cdots+2^{n-k-1}=2^{n-k}-1 \text{ nodes.}$$

Since the number of binary decision trees is m_1, the total number of nodes for m_1 binary decision trees is $m_1 \cdot (2^{n-k}-1)$. Therefore, the number of nodes in the SMTBDD for m output functions can be at most

$$\min_{k=1}^{n}\{(m_1-1)+m_1\cdot(2^{n-k}-1)+r^{2^k}\}$$
$$=\min_{k=1}^{n}\{m_1\cdot 2^{n-k}-1+r^{2^k}\}. \quad \square$$

Theorem 3: Let an SMTBDD(m) represent m functions $f_i = x_i$ $(i = 0,1,\ldots,m-1)$, and let $[f_0, f_1, \ldots, f_{m-1}]$ be the group of an m-output function. Then, the size of the SMTBDD(m) for grouping $[f_0, f_1, \ldots, f_{m-1}]$ is $2^{m+1}-1$ [8]. □

3. An Optimization Algorithm for SMTBDD(k)s

In this section, we will show an algorithm for deriving small SMTBDD(k)s using clique covers. Note that this algorithm can be used for SMTBDD(k)s with $k \geqq 3$. A similar algorithm for SMTBDD(2)s was considered in [7]. The clique covering is one of the NP-hard problems of graph optimization [16]. Usually, the edge or vertex weighted graph for this problem is considered [10]. For the optimization of SMTBDD(k)s, we will use a clique weighted graph, i.e. each group of vertices has a weight.

Definition 6: A *clique* of a graph is a set of vertices such that every pair is connected by an edge.

Example 9: Each of c_0 and c_1 in Fig. 6 is the clique. □

Definition 7: Let $G=(V,E)$ be a graph, where V and E denote a set of vertices and a set of edges, respectively. A *clique cover* of G is a partition of V such that each set in the partition is a clique.

Example 10: In Fig. 6, a clique cover is formed by the cliques c_0 and c_1. □

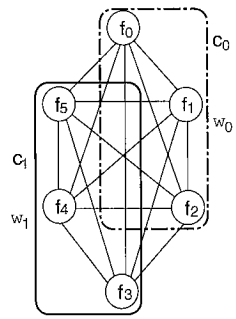


Fig. 6 The clique weighted graph.

Definition 8: Let $G=(V,E)$ be a graph. Then, G is a *clique weighted graph* iff each subset of vertices of G has a weight and each vertex pair is connected by an edge.

Example 11: Figure 6 is an example of a clique weighted graph. For simplicity, we show only two cliques with their weights. The weights of the cliques c_0 and c_1 are w_0 and w_1, respectively. □

Problem 1: Given a clique weighted graph $G=(V,E)$, find the clique cover such that the sum of weights of the cliques in the clique cover is minimum. □

Note that the weight corresponds to the upper bound on the size of the MTBDD, and the minimum weighted clique cover corresponds to the groupings of outputs that have small size, though sometimes they are not minimum.

3.1 The Weight Calculation Procedure

Each clique in the clique weighted graph has a weight. In this part, we show a method for calculating the weights of the cliques. From here, we assume that the size of a clique is three, i.e. each clique has three vertices.

Definition 9: Let $F=\{f_0, f_1, \ldots, f_{m-1}\}$ be a set of m output functions. Let F_i $(i=1,2,3,\ldots,s)$ be subsets of F. $\{F_1, F_2, \ldots, F_s\}$ is called a *partition* of F if

$$\bigcup_{i=1}^{s} F_i = F, \text{ and } F_i \cap F_j = \phi, \text{ where}$$

$i \neq j$, and $F_i \neq \phi$ for every i. Henceforth, each of $F_1, F_2, \ldots, F_s$ is called a *group* of output functions. Note that each vertex in the clique weighted graph represents an output function, where each group and each partition of output functions are denoted as a *clique* and a *clique cover*, respectively.

Example 12: Let $F=\{f_0, f_1, f_2, f_3, f_4, f_5\}$ be a set of 6-output function. Then, the partitions of F into groups are as follows: $\{[f_0,f_1,f_2], [f_3,f_4,f_5]\}$, $\{[f_0,f_1,f_3], [f_2,f_4,f_5]\}$, $\{[f_0,f_1,f_4], [f_2,f_3,f_5]\}$, $\{[f_0,f_1,f_5], [f_2,f_3,f_4]\}$, $\{[f_0,f_2,f_3], [f_1,f_4,f_5]\}$, $\{[f_0,f_2,f_4], [f_1,f_3,f_5]\}$, $\{[f_0,f_2,f_5], [f_1,f_3,f_4]\}$, $\{[f_0,f_3,f_4], [f_1,f_2,f_5]\}$, $\{[f_0,f_3,f_5], [f_1,f_2,f_4]\}$, and $\{[f_0,f_4,f_5], [f_1,f_2,f_3]\}$. □

Definition 10: Let F be an n-input m-output function. The *dependency matrix* $B = (b_{ij})$ for F is a *0-1* matrix with n columns and m rows. $b_{ij} = 1$ iff f_i depends on x_j, and $b_{ij} = 0$ otherwise, where $i = 0, 1, ..., m-1$, and $j = 1, 2, ..., n$.

Example 13: Consider the 4-input 6-output function:

$$\begin{aligned} f_0(x_1, x_2, x_3, x_4) &= x_2 x_3, \\ f_1(x_1, x_2, x_3, x_4) &= x_1 x_4 \vee x_2, \\ f_2(x_1, x_2, x_3, x_4) &= x_1 \vee x_3, \\ f_3(x_1, x_2, x_3, x_4) &= x_3, \\ f_4(x_1, x_2, x_3, x_4) &= x_1 \vee x_3 x_4, \text{ and} \\ f_5(x_1, x_2, x_3, x_4) &= x_4. \end{aligned}$$

The dependency matrix is

$$B = \begin{array}{c} \\ f_0 \\ f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}.$$

□

Definition 11: Let F be an n-input m-output function, and let $[f_i, f_j, f_k]$ be a group of output functions. The *group-dependency matrix* $A = (a_{ij})$ for F is a *0-1* matrix with n columns and $\frac{m(m-1)(m-2)}{6}$ rows. $a_{ij} = 1$ iff at least one of the outputs depends on x_j, and $a_{ij} = 0$, otherwise.

Example 14: Consider the 6-output function in Example 13. The group-dependency matrix A is given as follows:

$$A = \begin{array}{c} \\ [f_0, f_1, f_2] \\ [f_0, f_1, f_3] \\ [f_0, f_1, f_4] \\ [f_0, f_1, f_5] \\ [f_0, f_2, f_3] \\ [f_0, f_2, f_4] \\ [f_0, f_2, f_5] \\ [f_0, f_3, f_4] \\ [f_0, f_3, f_5] \\ [f_0, f_4, f_5] \\ [f_1, f_2, f_3] \\ [f_1, f_2, f_4] \\ [f_1, f_2, f_5] \\ [f_1, f_3, f_4] \\ [f_1, f_3, f_5] \\ [f_1, f_4, f_5] \\ [f_2, f_3, f_4] \\ [f_2, f_3, f_5] \\ [f_2, f_4, f_5] \\ [f_3, f_4, f_5] \end{array} \begin{array}{c} \begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \end{array}.$$

□

Note that the row for $[f_i, f_j, f_k]$ in A is equal to bit-wise OR of rows for f_i, f_j, and f_k in the dependency matrix B.

Definition 12: Let $r[f_i, f_j, f_k]$ be the *number of distinct output vectors* for the group of outputs $[f_i, f_j, f_k]$. Note that $1 \leqq r[f_i, f_j, f_k] \leqq 8$. $r[f_i, f_j, f_k]$ is equal to the number of non-zero functions in $f_i f_j f_k$, $\bar{f}_i \bar{f}_j f_k$, $\bar{f}_i f_j \bar{f}_k$, $\bar{f}_i f_j f_k$, $f_i \bar{f}_j \bar{f}_k$, $f_i \bar{f}_j f_k$, $f_i f_j \bar{f}_k$, and $\bar{f}_i \bar{f}_j \bar{f}_k$.

Example 15: Consider the 6-output function in Example 13. There are 20 groups of output functions. The number of distinct output vectors $r[f_i, f_j, f_k]$ for each group $[f_i, f_j, f_k]$ is calculated as follows:
For $r[f_0, f_2, f_3]$:

$$\begin{aligned} f_0 f_2 f_3 &= x_1 x_2 x_3 \vee x_2 x_3, \\ f_0 f_2 \bar{f}_3 &= 0, \\ f_0 \bar{f}_2 f_3 &= 0, \\ f_0 \bar{f}_2 \bar{f}_3 &= 0, \\ \bar{f}_0 f_2 f_3 &= x_1 \bar{x}_2 x_3 \vee \bar{x}_2 x_3, \\ \bar{f}_0 f_2 \bar{f}_3 &= x_1 \bar{x}_2 \bar{x}_3 \vee x_1 \bar{x}_3, \\ \bar{f}_0 \bar{f}_2 f_3 &= 0, \text{ and} \\ \bar{f}_0 \bar{f}_2 \bar{f}_3 &= \bar{x}_1 \bar{x}_2 \bar{x}_3 \vee \bar{x}_3. \end{aligned}$$

Since the number of non-zero functions is four, we have $r[f_0, f_2, f_3] = 4$. Similarly, we can calculate the number of distinct output vectors for other groups of output functions. □

Definition 13: Let $s(i, j, k)$ be the *size of the support* for a group of output functions $[f_i, f_j, f_k]$. The *weight* $w(i, j, k)$ for $[f_i, f_j, f_k]$ is $\min_{t=0}^{n-1}\{2^{s(i,j,k)-t} - 1 + (r[f_i, f_j, f_k])^{2^t}\}$. This will be the *weight of the clique* in the clique weighted graph.

Example 16: Consider the 6-output function in Example 13. $w(0, 2, 3)$ is calculated as follows: From Example 13, we know that $r[f_0, f_2, f_3] = 4$, and $s(0, 2, 3) = 3$. $w(0, 2, 3)$ takes its minimum when $t = 0$. Therefore, $w(0, 2, 3) = 2^3 - 1 + 4 = 11$. Similarly, we can calculate the weights for other groups. Since $w(i, j, k)$ is an upper bound on the size of the MTBDD for $[f_i, f_j, f_k]$ (Theorem 1), the MTBDD with the minimum weight is relatively small. □

3.2 Optimization of SMTBDD(3)s

To find the minimum weighted clique cover is an NP-hard problem of graph optimization [16]. So, we use a heuristic algorithm for finding a clique cover with small weight.

Algorithm 1: (Optimization of SMTBDD(3)s)

Input: A graph $G = (V, E)$.

Output: A clique cover K of G whose sum of weights of the cliques is relatively small.

Method: Firstly, calculate the weights of all cliques in

```
procedure Weightedclique(V, E) {
C ← set of cliques in which each clique consists of a
triple of vertices;
for each c ∈ C do {
/* c is a clique with a triple of vertices */
 w(c) ← min_{t=0}^{n-1} {2^{s(i,j,k)-t} - 1 + (r[f_i, f_j, f_k])^{2^t}};
/* "w(c)" denotes the upper bound on the size
   of an MTBDD */
   }
 }
procedure Min Weightclique cover(V, E) {
   Weightedclique(V, E);
 Make a list of weights, W sorted in ascending order;
 while C ≠ φ do {
    Select c ∈ C with the smallest weight w(c)
     from W;
    K ← K ∪ {c};
    Eliminate the cliques that contain the
    vertices in c from C;
    Update W and C;
     }
 return K;
 }
```

Fig. 7 Pseudocode for optimizing SMTBDD(3)s.

the graph G as shown in procedure "Weightedclique" in Fig. 7. Secondly, use procedure "MinWeightclique-cover" in Fig. 7 to find the clique cover with small weight, where "W" is the list of sorted weights of C, C is the set of cliques, and $w(c)$ is the weight of the clique c.

Since Algorithm 1 is greedy one, it may not obtain the optimal solutions, but we can expect good solutions.

4. Experimental Results

We implemented C programs for constructing SBDDs and MTBDDs. SMTBDDs were constructed from SBDDs. We developed a C program for Algorithm 1 and built SMTBDD(3)s. The SMTBDD(2)s were obtained by the method in [7]. In [7], we also showed that, in most cases, our algorithm obtains the best partition of output functions for SMTBDD(2)s. We used dynamic ordering [11] for the input variables that minimized the sizes of SBDDs, MTBDDs, and SMTBDDs. The numbers of terminal nodes in the SBDD, MTBDD, SMTBDD(2), and SMTBDD(3) are at most 2, 2^m, 4, and 8, respectively, where m is the number of output functions.

Table 4 compares the sizes of BDDs for small benchmark functions. For most benchmark functions in Table 4, MTBDDs are larger than SMTBDD(2)s and SMTBDD(3)s. Sometimes MTBDDs are much larger than SMTBDDs. For example, the size of the MTBDD for ts10 is 589837, while the size of the SMTBDD(2) for ts10 is 121. For many functions, SMTBDD(3)s are smaller than SMTBDD(2)s. The *ratios* show the relative sizes of SMTBDD(3)s to SMTBDD(2)s. An SMTBDD_{min} denotes either an SMTBDD(2) or an SMTBDD(3), with fewer nodes. In Table 4, 15 SMTBDD(3)s are the smallest among four types of BDDs, while 9 SMTBDD(2)s are the smallest among four types of BDDs. Note that the smallest BDD is the BDD with the fewest nodes. For some functions, MTBDDs are the smallest, e.g. nrm4. The *average relative sizes* for SBDDs, MTBDDs, and SMTBDDs are 1.00, 152.73, and 0.80, respectively. Furthermore, the columns 7 and 9 in Table 4 show the construction times for SMTBDD(2)s and SMTBDD(3)s, respectively. The maximum construction time in Table 4 was 1.2 seconds that required for ts10 to construct the SMTBDD(3) from the SBDD.

Table 5 compares the sizes of BDDs for larger benchmark functions. In Table 5, the SMTBDD denotes either an SMTBDD(2) or an SMTBDD(3). For many benchmark functions, MTBDDs were impossible to construct due to their excessive sizes, while other BDDs were possible, e.g. cps. Sometimes SMTBDDs are much smaller than SBDDs, e.g. the size of the SMTBDD for apex5 is 847, while the size of the SBDD for apex5 is 1167. In Table 5, c880 was one of the most complex benchmark functions, and our program required 10.7 seconds to construct the SMTBDD(2) from the SBDD on a JU1/170 with 160 MB of main memory (Sun Ultra1-170 compatible workstation). Note that in most cases, the constructions of SMTBDDs are not so time consuming as MTBDDs. For example, the construction time of the SMTBDD(2) from the BDD for ts10 was 2.2 seconds, while the construction time of the MTBDD from the BDD for ts10 was 30.2 seconds (these were construction times for both BDDs without using variable ordering).

5. Conclusions and Comments

In this paper, we proposed a method to represent multiple-output functions using SMTBDDs. We also compared three types of BDDs. SMTBDD(k)s are not so large as MTBDDs, and the evaluation speed is k times faster than SBDDs, since k outputs are evaluated simultaneously. In addition, for large benchmark functions, MTBDDs are impossible to construct due to their excessive sizes, while SMTBDDs are possible. We presented an algorithm for grouping output functions to reduce the size of SMTBDD(k)s. Experimental results showed the compactness of SMTBDD(k)s. By combining an SMTBDD(2) and an SMTBDD(3), we also proposed a compact representation for the SMTBDD which denotes either an SMTBDD(2) or an SMTBDD(3) with fewer nodes. The average relative sizes for SBDDs, MTBDDs, and SMTBDDs are 1.00, 152.73, and 0.80, respectively. Thus, SMTBDDs compactly represent many multiple-output functions, and

Table 4 Number of nodes in the BDDs to represent various benchmark functions.

Function name	In	Out	SBDD	MTBDD	SMT-BDD(2)	time[1] (sec)	SMT-BDD(3)	time[2] (sec)	ratio[3]	SMT-BDD$_{min}$
alu1	12	8	29	580	22	0.2	130	0.5	5.90	22*
bench1	9	9	688	691	732	0.6	675	0.4	0.92	675*
clip	9	5	111	139	117	0.3	167	0.5	1.42	117
exp	8	18	212	153	219	0.4	186	0.3	0.84	186†
ex1010	10	10	1421	1548	1402	0.7	1361	0.5	0.97	1361*
gary	15	11	322	228	334	0.5	210	0.3	0.62	210*
inc	7	9	80	85	76	0.3	72	0.2	0.94	72*
in0	15	11	290	570	294	0.7	238	0.4	0.80	238*
log8	8	8	205	303	187	0.4	246	0.6	1.31	187*
m2	8	16	133	140	133	0.9	85	0.8	0.63	85*
max128	7	24	189	133	220	0.7	100	0.4	0.45	100*
nrm4	8	5	158	130	147	0.3	195	0.6	1.32	147†
prom2	9	21	958	578	766	0.7	466	0.3	0.60	466*
p1	8	18	354	370	388	0.5	220	0.3	0.56	220*
p3	8	14	222	247	241	0.7	191	0.5	0.79	191*
rd53	5	3	27	21	26	0.2	21	0.1	0.80	21†
radd	8	5	45	98	39	0.3	62	0.5	1.58	39*
sao2	10	4	90	69	74	0.2	62	0.1	0.83	62*
sex	9	14	68	140	62	0.6	59	0.5	0.95	59*
sqr4	4	8	27	31	25	0.4	33	0.5	1.32	25*
sqr6	6	12	84	127	76	0.7	104	0.9	1.36	76*
ts10	22	16	163	589837	121	0.9	225	1.2	1.85	121*
test1	8	10	406	452	428	0.6	344	0.4	0.80	344*
5xp1	7	10	93	255	76	0.5	99	0.7	1.30	76*
average[4] relative size			1.00	152.73						0.80

In: number of inputs; Out: number of outputs.
*SMTBDD with the fewest nodes among four types of BDDs.
†Number of nodes in the SMTBDD is less than the SBDD.

1. CPU time in sec on a JU1/170 for constructing the SMTBDD(2) from the SBDD.
2. CPU time in sec on a JU1/170 for constructing the SMTBDD(3) from the SBDD.
3. ratio $= \frac{size(SMTBDD(3))}{size(SMTBDD(2))}$.
4. average relative size $= \frac{1}{N}\sum_{i=1}^{N} \frac{size\ of\ the\ BDD\ for\ function\ i}{size\ of\ the\ SBDD\ for\ function\ i}$,

   where N is the total number of functions.

Table 5 Number of nodes in the BDDs to represent large benchmark functions.

Function name	In	Out	SBDD	MTBDD	SMTBDD‡	time[1] (sec)
apex1	45	45	1321	–	1619	1.7
apex2	39	3	67	62	79	0.1
apex5	117	88	1167	–	847	2.7
al2	16	47	145	4707	115	0.3
alcom	15	38	118	5033	76	0.3
cps	24	109	1095	–	1127	2.1
c880	60	26	4168	–	15634	10.7
pdc	16	40	609	18451	494	0.6
misex3	14	14	557	2910	522	0.9
prom1	9	40	2012	852	1710	1.0
soar	83	94	632	–	918	2.4
x1	51	35	586	–	938	1.5
x3	135	99	715	–	1131	2.8
xparc	41	73	1949	3861	2277	1.9

" – " indicates memory overflow.
‡The SMTBDD denotes either an SMTBDD(2) or an SMTBDD(3).

1. CPU time in sec on a JU1/170 for constructing the SMTBDD from the SBDD.

are useful for TDM (time-division multiplexing) realizations of multiple-output networks, look-up table type FPGA design, and logic simulation.

A multiple-output function can also be represented by a BDD for characteristic functions (CFs) [2],[3]. In a BDD for CFs, if the input variables appear before the output variables in the variable ordering, then an n-input m-output function can be evaluated in $O(n+m)$ time. However, in most cases, BDDs for CFs are much larger than the corresponding SBDDs. Moreover, if all the output functions depends on all the input variables, then the size of the BDD for CFs is greater than the corresponding MTBDD. By dropping the above ordering restriction, we can reduce the size of the BDD, but we can not guarantee the time of $O(n+m)$ to evaluate a multiple-output function. Furthermore, in most cases, the sizes of such BDDs are still larger than the corresponding SBDDs. For example, the size of the BDD for CFs of c880 in [3] is about 90 times larger than the corresponding SBDD, while the size of the SMTBDD for c880 is about 4 times larger than the corresponding SBDD. In many cases, the sizes of SMTBDDs are not so large as BDDs for CFs. Also, the evaluation speed of an SMTBDD is k times faster than the corresponding SBDD.

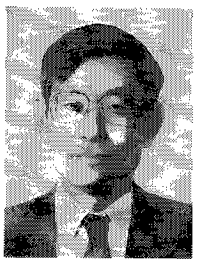
Acknowledgements

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. This paper is based on [7],[8].

References

[1] S.B. Akers, "Binary decision diagrams," IEEE Trans. Comput., vol.C-27, no.6, pp.509–516, June 1978.
[2] P. Ashar and S. Malik, "Fast functional simulation using branching programs," Proc. International Conference on Computer-Aided Design, pp.408–412, Nov. 1995.
[3] C. Scholl, R. Drechsler, and B. Becker, "Functional simulation using binary decision diagrams," Proc. International Conference on Computer-Aided Design, pp.8–12, Nov. 1997.
[4] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Trans. Comput., vol.C-35, no.8, pp.677–691, Aug. 1986.
[5] T. Sasao and J.T. Butler, "A method to represent multiple-output switching functions by using multi-valued decision diagrams," Proc. International Symposium on Multiple-Valued Logic, pp.248–254, May 1996.
[6] T. Sasao and J.T. Butler, "A design method for look-up table type FPGA by pseudo-Kronecker expansion," Proc. International Symposium on Multiple Valued Logic, pp.97–106, May 1994.
[7] 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, pp.212–217, Nov. 1996.
[8] Hafiz Md. Hasan Babu and T. Sasao, "Representations of multiple-output logic functions using shared multi-terminal binary decision diagrams," the Seventh Workshop on Synthesis And System Integration of MIxed Technologies (SASIMI'97), Osaka, Japan, pp.25–32, Dec. 1997.
[9] Hafiz Md. Hasan Babu and T. Sasao, "Design of multiple-output networks using time domain multiplexing and shared multi-terminal multiple-valued decision diagrams," IEEE International Symposium on Multiple Valued Logic (ISMVL-98), pp.45–51, May 1998.
[10] E. Balas and C.S. Yu, "Finding a maximum clique in an arbitrary graph," SIAM J. Comput., vol.15, pp.1054–1068, 1986.
[11] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," Proc. International Conference on Computer-Aided Design, pp.42–47, Nov. 1993.
[12] T. Sasao, ed., "Logic Synthesis and Optimization," Kluwer Academic Publishers, Boston, 1993.
[13] S. Minato, "Graph-based representation of discrete functions," in Representations of Discrete Functions, eds. T. Sasao and M. Fujita, Kluwer Academic Publishers, Boston, 1996.
[14] A. Srinivasan, T. Kam, S. Malik, and R.K. Brayton, "Algorithm for discrete functions manipulation," Proc. International Conference on Computer-Aided Design, pp.92–95, Nov. 1990.
[15] P.C. McGeer, K.L. McMillan, A. Saldanha, A.L. Sangiovanni-Vincentelli, and P. Scaglia, "Fast discrete function evaluation using decision diagrams," International Workshop on Logic Synthesis, pp.6.1–6.9, May 1995. Also, in Proc. International Conference on Computer-Aided Design, pp.402–407, Nov. 1995.
[16] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.

Hafiz Md. Hasan Babu was born in Bangladesh. He received the M.Sc. degree in Computer Science and Engineering from the Technical University of Brno, Czechoslovakia, in 1992. He is with the Department of Computer Science and Engineering, Khulna University, Bangladesh since 1993. In 1995, he was at the Asian Institute of Technology (AIT), Thailand under the DAAD Fellowship from the Federal Republic of Germany. He is currently working towards the Ph.D. degree with the Japanese Government Scholarship at the Kyushu Institute of Technology, Iizuka, Japan. His research interests include logic synthesis and representations of logic functions.



Tsutomu Sasao received the B.E., M.E., and Ph.D. degrees in Electronic Engineering from Osaka University, Osaka Japan, in 1972, 1974, and 1977, respectively. He was with Osaka University Japan, IBM T. J. Watson Research Center and Naval Postgraduate School in Monterey, California. Now, he is a Professor of Kyushu Institute of Technology, Iizuka, Japan. His research areas include logic design and switching theory, representations of logic functions, and multiple-valued logic. He has published seven books on logic design including, "Logic Synthesis and Optimization," and "Representation of Discrete Functions" Kluwer Academic Publishers 1993, and 1996, respectively. He has served Program Chairman for the IEEE International Symposium on Multiple-Valued Logic (ISMVL) many times. Also, he was the Symposium Chairman for the ISMVL-98 held in Fukuoka, Japan in 1998. He received the NIWA Memorial Award in 1979, and Distinctive Contribution Awards from IEEE Computer Society MVL-TC in 1987 and 1996. Now, he is an associate editor of IEEE Transactions on Computers. He is a Fellow of IEEE.