A Method to Represent Multiple-Output Switching Functions by Using Multi-Valued Decision Diagrams

Tsutomu Sasao

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

Jon T. Butler

Department of Electrical and
Computer Engineering
Naval Postgraduate School
Monterey, CA 93943-5121, U.S.A.

February 19, 1996

Abstract

Multiple-output switching functions can be simulated by multiple-valued decision diagrams (MDDs) at a significant reduction in computation time. We analyze the following approaches to the representation problem: shared multiple-valued decision diagrams (SMDDs), multi-terminal multiple-valued decision diagrams (MTMDDs), and shared multi-terminal multiple-valued decision diagrams (SMTMDDs). For example, we show that SMDDs tend to be compact, while SMTMDDs tend to be fast. We present an algorithm for grouping input variables and output functions in the MDDs.

1 Introduction

Various methods exist to represent discrete functions. Among them, graph based representations such as BDDs (binary decision diagrams) are extensively used in logic synthesis, test, and verification [4]. Multiple-valued decision diagrams (MDDs) are multiple-valued extensions of BDDs, and have been used to design logic networks [14, 8, 5, 9, 10]. Recently, McGeer *et al.* developed a logic simulator based on MDDs [12]. They showed that the MDD based simulator is orders-of-magnitude faster than a conventional one. Their method is summarized as follows:

1. Represent a given logic function by a BDD.
2. Group k variables into a single 2^k-valued variable forming an MDD from the BDD. Each node in the MDD has 2^k children. We assume that $n = rk$ is the number of input variables, where $k \geq 2$ is a constant.
3. Translate the MDD into a table on which function evaluation is performed by a sequence of address lookups.

An advantage of the MDD is a reduction in memory accesses needed to evaluate it compared with the BDD from which it was derived. Indeed, grouping k binary inputs together to form a single MDD variable reduces computation time by a factor of k. However, there is a tradeoff. Since each group of k binary variables can assume 2^k possible values, the size of the MDD tends to increase by a factor approaching 2^k. [12] claims that $k = 5$ gives the best performance for their prototype simulator. However, they did not show any theoretical or experimental justification.

To represent a logic function efficiently, it is essential to reduce the number of nodes in its MDD. Three methods exist.

1. Group binary variables to form multiple-valued variables.
2. Order the multiple-valued variables in the MDD.
3. Group the outputs.

In this paper, we consider a method of representation for multiple-output functions by MDDs, where $k = 2$, i.e., each MDD variable is 4-valued. Specifically, we consider a method for pairing input variables and pairing output functions to reduce the number of nodes.

We considered the case $k = 2$ for the following reason:

1. When $k = 2$, a node for a 4-valued MDD is realized by a "4 to 1 multiplexer," which is available in a CMOS gate array library. Its cost is 4 times that of a 2-input NAND gate [7]. Also, a node for a 4-valued MDD is realized by a 6-input LUT (look-up table) [3, 9]. FPGAs (field programmable gate arrays) with such LUTs are produced by AT&T [2].
2. $k = 2$ is the simplest case, and the design and analysis are easier than for the general case.

Strategies for grouping output functions are also useful for reducing the memory requirement of MTTDDs (Multi-Terminal Ternary Decision Diagrams) that are indispensable in the optimization of AND-EXOR expressions [8, 11]. Other methods to represent multiple-output functions using BDDs are developed for fast logic simulation [1].

2 Definitions and Basic Properties

Definition 2.1 *Let $P = \{0, 1, \ldots, p-1\}$ and $B = \{0, 1\}$. A mapping f: $B^n \rightarrow B$ is a switching function. A mapping f: $P^n \rightarrow P$ is a p-valued logic function. A mapping f: $P^n \rightarrow B$ is a p-valued input two-valued output function.*

Definition 2.2 *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 that is the AND of all literals that compose it. The expression

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

is a sum-of-products expression (SOP), where $\bigvee_{(S_1, S_2, \ldots, S_n)}$ denotes the inclusive-OR of products terms.

Lemma 2.1 *An arbitrary p-valued input two-valued output function can be expanded as*

$$\begin{aligned} & f(X_1, X_2, \ldots, X_r) \\ &= X_1^0 f(0, X_2, \ldots, X_r) \vee X_1^1 f(1, X_2, \ldots, X_r) \vee \\ &\quad \cdots \vee X_1^{p-1} f(p-1, X_2, \ldots, X_r). \end{aligned}$$

This is the Shannon expansion with respect to X_1.

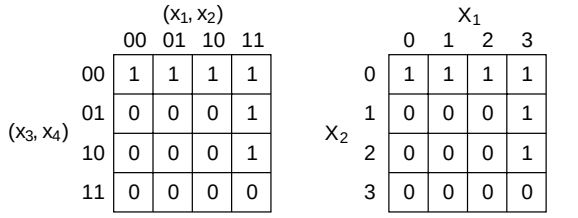
In the derivations that follow, it is convenient to let x_i represent a two-valued variable and to let X_i represent a 2^k valued variable corresponding to k two-valued variables. For example, consider a switching function $f(x_1, x_2, \ldots, x_{2r})$, which can be written as $f(X_1, X_2, \ldots, X_r)$, where X_i is either 0, 1, 2, or 3 if $(x_{2i-1}, x_{2i}) = (0,0)$, $(0,1)$, $(1,0)$ and $(1,1)$, respectively. We use the notation $X_i = (x_{2i-1}, x_{2i})$ to denote the relationship between X_i and x_{2i-1} and x_{2i}.

Example 2.1 *The switching function shown in Fig. 2.1(a) can be converted into a four-valued input two-valued output function as shown in Fig. 2.1(b), where $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4)$.*
(End of Example)

Definition 2.3 *Let f be a function. The set of input variables on which f depends is the support of f, and is denoted as $Support(f)$.*

Example 2.2 *Let $f(x_1, x_2, x_3, x_4) = x_1 x_2 \vee x_3 \bar{x}_4 \vee x_3 x_4$. Then, $Support(f) = \{x_1, x_2, x_3\}$, since f is also represented as $f = x_1 x_2 \vee x_3$. (End of Example)*

Definition 2.4 *A multi-valued decision diagram (MDD) is a generalization of a binary decision diagram (BDD). Specifically, an internal node of an MDD may have more than two children (Fig. 2.2). An MDD having more than two kinds of terminal nodes (e.g., $0, 1, \cdots, p-1$) is called a multi-terminal MDD (MTMDD).*



(a) A switching function. (b) A four-valued input two-valued output function.

Figure 2.1: Example 2.1.

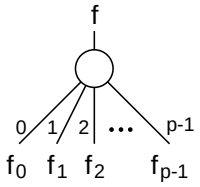


Figure 2.2: Node for an MDD.

For a given BDD, we can group the binary variables, and obtain an MDD.

Example 2.3 *Fig. 2.3 shows a BDD for $f = x_1 \oplus x_2 \oplus x_3 \oplus x_4$. Let $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4)$ represent logic value in the range $\{0, 1, 2, 3\}$. By this, we have a 4-valued input two-valued output function*

$$F(X_1, X_2) = X_1^{\{0,3\}} X_2^{\{1,2\}} \vee X_1^{\{1,2\}} X_2^{\{0,3\}}.$$

Fig. 2.4 shows an MDD for F. *(End of Example)*

3 Grouping Input Variables

The number of nodes in an MDD depends on the method of grouping input variables as well as the ordering of the multiple-valued variables.

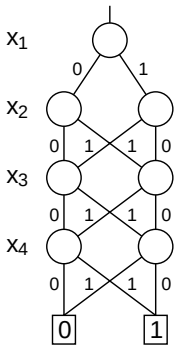


Figure 2.3: BDD for $f = x_1 \oplus x_2 \oplus x_3 \oplus x_4$.

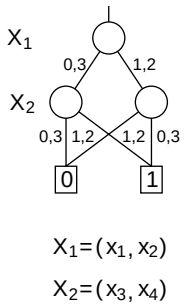


Figure 2.4: MDD for $X_1^{\{0,3\}}X_2^{\{1,2\}} \vee X_1^{\{1,2\}}X_2^{\{0,3\}}$.

Example 3.1 *Consider the function* $f = x_1x_2(x_3 \oplus x_4) \vee \bar{x}_3\bar{x}_4$.

1) *When the input variables are grouped as* $X_1 = (x_1, x_2)$ *and* $X_2 = (x_3, x_4)$, *the MDD for* f *has three non-terminal nodes as shown in Fig. 3.1(a).*
2) *When the input variables are grouped as* $X_1 = (x_3, x_4)$ *and* $X_2 = (x_1, x_2)$, *the MDD for* f *has only two non-terminal nodes as shown in Fig. 3.1(b).*
3) *When the input variables are grouped as* $X_1 = (x_1, x_3)$ *and* $X_2 = (x_2, x_4)$, *the MDD for* f *has four non-terminal nodes as shown in Fig. 3.1(c).*

Thus, for this function, the grouping of 2) is the best among the three. *(End of Example)*

Suppose that the input variables are partitioned into X_1 and X_2, where $X_1 = (x_i, x_j)$. By renaming the input variables, we can represent f as $f(X) = f(X_1, X_2)$. Next, consider the sub-functions: $f(0, 0, X_2)$, $f(0, 1, X_2)$, $f(1, 0, X_2)$, and $f(1, 1, X_2)$. Let μ be the number of distinct non-constant functions among these four functions. Let the BDD representing f have x_i as the highest variable and x_j as the second highest. Since f depends on x_i, the root node representing f has two children nodes representing $f_{x_i=0}$ and $f_{x_i=1}$. Since there are μ non-constant subfunctions associated with assignments of values to x_i and x_j, the number of non-terminal nodes in the lower level is μ. As an example, consider $f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$, and let $(x_1, x_2) = (x_i, x_j)$. Among the four subfunctions, $f(0, 0, x_3, x_4)$, $f(0, 1, x_3, x_4)$, $f(1, 0, x_3, x_4)$, and $f(1, 1, x_3, x_4)$, are two distinct ones, $x_3 \oplus x_4$ and $x_3 \oplus x_4 \oplus 1$. Thus, $\mu = 2$, and there are two non-terminal nodes at the second level. This is shown in Fig. 2.3. Further, in the multiple-valued version of a function f, represented as $f(X_1, X_2, \ldots, X_r)$, where $X_1 = (x_i, x_j)$, there will also be μ nodes. In our running example, there are two nodes at the X_2 level in Fig. 2.4, which is an MDD representation of the BDD in Fig. 2.3. This shows that a grouping in the BDD that reduces nodes tends to reduce nodes in the equivalent MDD. If $\mu \leq 3$, then (x_i, x_j) is a candidate pair. It is clear that if f is partially symmetric with respect to x_i and x_j, then $\mu \leq 3$.

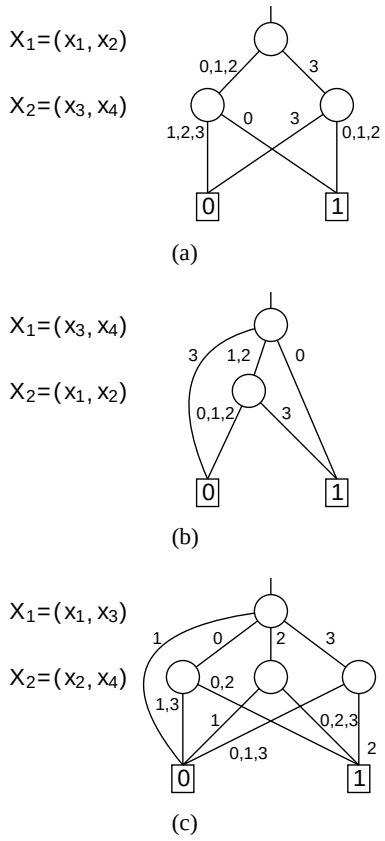


Figure 3.1: MDD for $f = x_1x_2(x_3 \oplus x_4) \vee \bar{x}_3\bar{x}_4$.

4 Representation of Multiple-Output Functions

Practical logic networks have, usually, many outputs. In this section, we consider various methods to represent multiple-output functions by MDDs.

4.1 Shared MDD

Fig. 4.1 shows the general structure of a shared MDD (SMDD), where each function f_i $(i = 0, 1, \ldots, m-1)$ has its corresponding MDD. These MDDs may share sub-graphs. Above the root nodes, the output selection variables are used. The advantage of this data structure for use in a simulator is that the sizes of MDDs are relatively small. The disadvantage is that we need to evaluate MDDs m times. This is because the MDD produces only one output for each assignment of values to the output selection variables. Its use in a simulation package requires the production of all outputs sequentially, and this results in a slow response time.

Example 4.1 *Fig. 4.2 shows the shared BDD (SBDD) for the two-input two-output function in Table 4.1.* *(End of Example)*

4.2 Multi-Terminal MDD

Fig. 4.3 shows the general structure of a multi-terminal MDD (MTMDD), where the terminals are

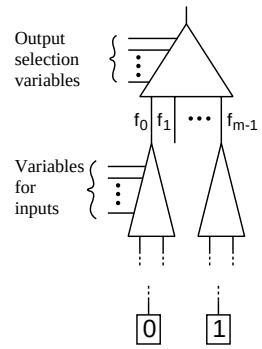


Figure 4.1: General structure of an SMDD.

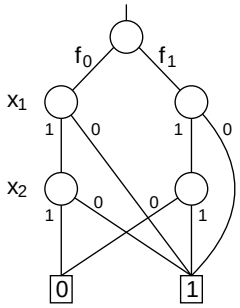


Figure 4.2: SBDD for 2-input 2-output function.

m-bit binary vectors. In a conventional MDD, an assignment of values to the variables corresponds to a path that terminates on a single binary value. This value is the function value for that assignment. However, in an MTMDD, a path terminates on a vector of m-bits corresponding to m functions. The merit of this data structure is that the values of all the outputs can be evaluated simultaneously. This is because the MDD produces all the output values for each assignment of values to the input variables. The demerit is that the size of the MTMDDs tends to be larger than that of SMDDs, when m is large.

Example 4.2 *Fig. 4.4 shows an MTBDD for the two-input two-output function in Table 4.1.*
(End of Example)

The above example shows a case where an MTBDD represents functions efficiently. However, the next example shows a case where an MTBDD is quite inefficient.

Table 4.1:

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

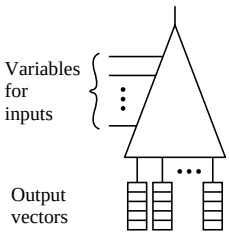


Figure 4.3: General structure of an MTMDD.

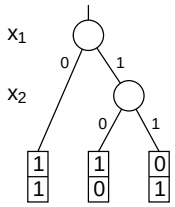


Figure 4.4: MTBDD for 2-input 2-output function.

Example 4.3 *Consider the three-output function: $f_0 = x_1$, $f_1 = x_2$, and $f_2 = x_3$. As shown in Fig. 4.5, the SBDD is simple, but the MTBDD is the complete tree for three variables.* *(End of Example)*

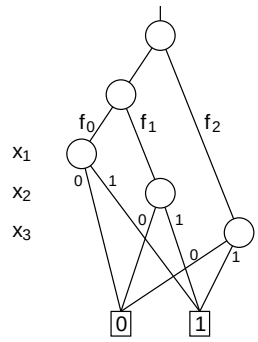
The above example suggests that a conversion from an SBDD to an MTBDD may increase the size of the structure exponentially. Note that, on the other hand, a conversion from an MTBDD to an SBDD can increase nodes by at most m times, where m is the number of outputs. Experimental results show that, in many cases, when m is large, the size of an MTBDD is larger than the size of the equivalent SBDD. The number of nodes tends to be larger than the equivalent SMDD.

4.3 MDD with Output Selection Variables

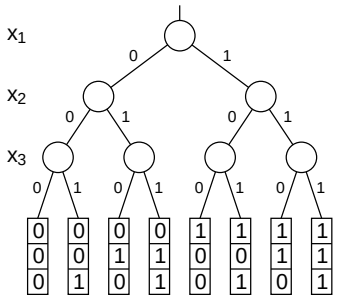
Fig. 4.6 shows the general structure of an MDD with output selection variables. This data structure is the same as the SMDD, except that the output selection variables are interchanged with the input variables. Also, in this case, we need additional time to select the output functions. However, the time to evaluate m outputs can be less than that of the equivalent SMDD, if we use a special technique in a simulation program.

4.4 Shared MTMDD

Fig. 4.7 shows the general structure of a shared MTMDD (SMTMDD). This data structure is a combination of an SMDD and an MTMDD. In an SMTMDD, the output selection variables select a set of outputs (in this example, two), and each MDD for g_i ($i = 0, 1, \ldots, m/2$) represents the set of outputs. Its



(a) SBDD



(b) MTBDD

Figure 4.5: SBDD and MTBDD.

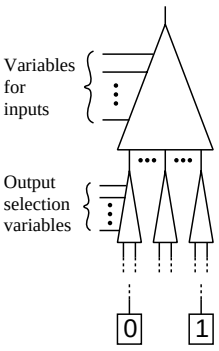


Figure 4.6: General structure of an MDD with output selection variable.

merit is that several (in this example, two) functions are evaluated simultaneously. Also, the functions can be grouped so that the size of the resulting MDD remains moderate.

4.5 Strategies for Grouping Output Functions

Consider an MTMDD for two-output function (f_i, f_j). In general, the MTMDD has $(0,0)$, $(0,1)$, $(1,0)$, and $(1,1)$ as terminal nodes. However if $f_i f_j = 0$, then $(1,1)$ never appears as a terminal node in the MTMDD for (f_i, f_j). Thus, this grouping of the output functions tends to produce a smaller MDD, since the number of terminal nodes is at most three. Similarly, if $f_i \bar{f}_j = 0$, $\bar{f}_i f_j = 0$, or $\bar{f}_i \bar{f}_j = 0$, then f_i and f_j is also a candidate pair. On the contrary, if f_i and f_j have disjoint supports, then they should not be paired, but represented by separate MDDs.

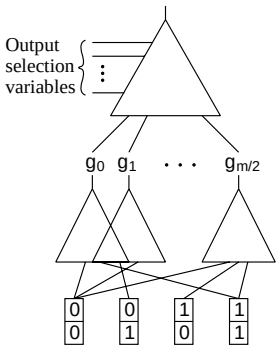


Figure 4.7: General structure of an SMTMDD.

$$\begin{array}{rcc} & x_2 & x_1\ x_0 \\ +) & y_2 & y_1\ y_0 \\ \hline & \underbrace{z_3\ z_2}_{G_1} & \underbrace{z_1\ z_0}_{G_0} \end{array}$$

Figure 4.8: Three-bit adder (adr3).

Example 4.4 *Consider the three-bit adder (adr3) shown in Fig. 4.8. Assume that the input variables and output functions are paired as follows:* $X_1 = (x_2, y_2)$, $X_2 = (x_1, y_1)$, $X_3 = (x_0, y_0)$, $G_1 = (z_3, z_2)$, $G_0 = (z_1, z_0)$. *Note that*

$$\begin{aligned} Support(z_0) &= \{x_0, y_0\}, \\ Support(z_1) &= \{x_0, y_0, x_1, y_1\}, \\ Support(z_2) &= \{x_0, y_0, x_1, y_1, x_2, y_2\}, \text{ and} \\ Support(z_3) &= \{x_0, y_0, x_1, y_1, x_2, y_2\}. \end{aligned}$$

Fig. 4.9 shows the SMDD: Four MDDs for z_3, z_2, z_1, *and* z_0 *are realized. In this case, the number of nodes is small. However, the simulation is slow since we have to evaluate MDDs four times, once for each of the four output functions. Fig. 4.10 shows the MTMDD: A single MDD with 4-bit terminal nodes. In this case, the number of nodes is large. However, the simulation is fast, since four outputs are evaluated simultaneously. Fig. 4.11 shows the SMTMDD, a pair of MDDs with 2-bit terminal nodes. In this case, the number of nodes is moderate. The simulation time is one-half that of the SMDD, since two outputs are evaluated simultaneously.* *(End of Example)*

5 Optimization Algorithm for MDDs

For simplicity, we assume that $n = 2r$, and m is an even number, where n denotes the number of in-

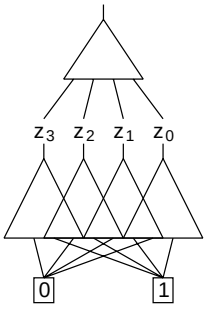


Figure 4.9: SMDD.

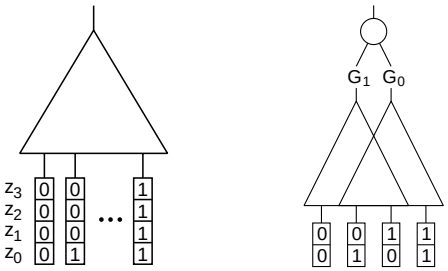


Figure 4.10: MTMDD. Figure 4.11: SMTMDD.

put variables, and m denotes the number of output functions.

5.1 Grouping and Ordering of Input Variables

Algorithm 5.1 *(Group the Input Variables)*

1. *Construct an SBDD from the given specification.*
2. *Find a good initial ordering of the input variables for the BDD, by interchanging adjacent variables.*
3. *Count the nodes in the MDD. This is done in a BDD data structure: If the BDD contains disjoint subgraphs, Fig. 5.1(a), (b), or (c), then the numbers of the corresponding nodes in the MDD are counted as one, two, or three, respectively. In Fig. 5.1(a), three BDD nodes are replaced by one MDD node. However, in Fig. 5.1(b), the BDD nodes are replaced by two MDD nodes. In Fig. 5.1(c), we need three MDD nodes. Thus, the number of nodes in an MDD is more than one third of that for the corresponding BDD.*

Algorithm 5.2 *(Order Input Variables in the MDD)*

1. *Construct an SMDD by Algorithm 5.1.*
2. *Reduce the number of nodes in the MDD by interchanging the adjacent variables.*

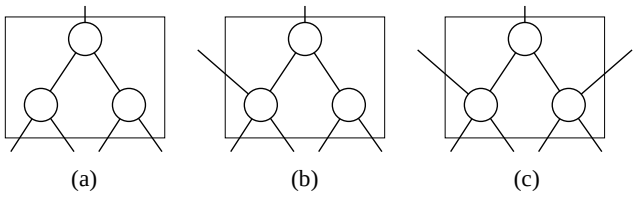


Figure 5.1: Enumeration of MDD nodes.

Table 6.1: Number of MDD nodes of represent adder.

Name	in	out	SBDD	SMDD	MTMDD	SMTMDD
adr3	6	4	25	11	26	15
adr5	10	6	46	18	120	26
adr7	14	8	65	25	502	37
adr9	18	10	87	31	2036	49

5.2 Grouping of Output Functions

Algorithm 5.3 *(Group the Output Functions)*
Let $f_0, f_1, \ldots, f_{m-1}$ be the output functions. Let m be even.

1. *Construct an SBDD from the given specification.*
2. *For $(0 \leq i \leq m-1)$, construct the BDD representing f_i. Calculate w_i, the number of nodes.*
3. *For $(0 \leq i < j \leq m-1)$, construct an SBDD representing f_i and f_j. Count w_{ij}, the number of nodes.*
4. *Let $W_{ij} \leftarrow w_i + w_j - w_{ij}$. Note that W_{ij} is the reduction in nodes achived by pairing f_i and f_j.*
5. *Consider the complete graph G with m nodes whose edges have weight W_{ij}. Obtain the maximum matching for G. That is, choose a set E of $m/2$ edges such that each node is incident to exactly one edge, and $\sum_{i<j} W_{ij}$ is maximum, where the sum is over all possible edges in E. We use a branch and bound algorithm.*
6. *Group the output functions according to the maximum matching of G.*

6 Experimental Results

[12] did not show the number of MDD nodes, so the comparison with their method is not made.

6.1 Representation of Adders

n-bit adders (adr n) for n =3, 5, 7, and 9 are represented by SBDDs (shared BDDs), SMDDs, MTMDDs, and SMTMDDs. Table 6.1 compares the number of nodes. Note that the number of terminal nodes for SBDDs, SMDDs, MTMDDs, and SMTMDDs are 2, 2, $2^{n+1}-1$, and 4, respectively. SMDDs require the fewest nodes among the three types of MDDs, and MTMDDs require the most number of nodes. The number of nodes in SMTMDDs is not so large, but the simulation time is half that of SMDDs, since two functions are evaluated simultaneously.

Table 6.2: Number of MDD nodes of represent various functions.

Name	in	out	SBDD	SMDD	MTMDD	SMTMDD
alu2	10	8	82	44	174	63
apla	10	12	123	67	85	90
bc0	26	11	624	363	209	378
clip	10	5	138	56	95	91
dc2	8	7	74	41	136	41
dk17	10	11	82	44	54	67
dk27	9	9	37	22	39	25
f51m	8	8	83	43	341	51
in1	16	17	580	322	188	437
in2	19	10	301	164	189	226
inc	7	9	89	46	44	48
misex1	8	7	45	26	19	33
misex3	14	14	580	322	1773	351
misj	35	14	59	35	3492	54
mlp4	8	8	151	79	170	92
radd	8	5	35	15	57	21
rd53	5	3	27	14	15	15
rd73	7	3	47	24	24	24
rd84	8	4	64	33	25	31
rot8	8	5	85	42	47	39
sao2	10	4	99	49	36	52
sex	9	14	63	40	105	50
sqr8	8	16	272	132	341	150
tial	14	8	774	380	388	624
ts10	22	16	177	73	240297	79
x6dn	39	5	263	149	148	188
z5xp1	5	10	82	41	213	55

6.2 Representation of Other Functions

Other benchmark functions are represented by SBDDs, SMDDs, MTMDDs, and SMTMDDs. Table 6.2 compares the number of nodes. Each MDD (BDD) was optimized by a heuristic program independently. Thus, MDDs (BDDs) for the same function may use different ordering of the input variables. In many cases, SMDDs required the fewest nodes among three MDDs. MTMDDs sometimes required excessive number of nodes, although they sometimes require the fewest nodes. SMTMDDs were not as large as MTMDDs, but were usually larger than SMDDs.

7 Conclusion and Comments

In this paper, we considered various methods to represent multiple-output functions by using MDDs. We presented algorithms for grouping input variables and output functions to reduce the number of nodes in MDDs. These methods are useful for fast logic simulation. The technique in 5.2 can be also used as a BDD data structure for representing multiple-output logic functions.

Acknowledgments

This work was supported in part by a Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan, and by a JSPS Fellowship. Mr. M. Matsuura developed the program and did the experiments.

References

[1] P. Ashar and S. Malik, "Fast functional simulation using branching programs," *in Proc. of the International Conference on Computer-Aided Design*, Nov. 1995.

[2] AT&T, *AT&T ORCA (Optimized Reconfigurable Cell Array) Series FPGA*, Product Brief, April. 1993.

[3] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field Programmable Gate Arrays*, Kluwer Academic Publishers, Boston 1992.

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

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

[6] S. Minato, "Graph-based representation of discrete functions," in T. Sasao and M. Fujita, (e.d.) *Representations of Discrete Functions*, Kluwer Academic Publishers, 1996 (to be published).

[7] NEC Corporation, *CMOS-8 Family Ver. 3.0 Block Libaray, User's Manual*, 1994.

[8] T. Sasao (ed.), *Logic Synthesis and Optimization*, Kluwer Academic Publishers (1993-01).

[9] 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 25-27, 1994, pp. 97-106.

[10] 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.

[11] T. Sasao and F. Izuhara, "Exact minimization of fixed polarity Reed-Muller expressions using multi-terminal EXOR ternary decision diagram," in T. Sasao and M. Fujita, (e.d.) *Representations of Discrete Functions*, Kluwer Academic Publishers, 1996 (to be published).

[12] 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 Thahoe, May, 1995, pp. 6_1-6_9. Also, *in Proc. of the International Conference on Computer-Aided Design*, pp. 402-407, Nov. 1995.

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

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