PAPER *Special Issue on Multiple-Valued Logic and Its Applications*

Time-Division Multiplexing Realizations of Multiple-Output Functions Based on Shared Multi-Terminal Multiple-Valued Decision Diagrams

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

SUMMARY This paper considers methods to design multiple-output networks based on decision diagrams (DDs). TDM (time-division multiplexing) systems transmit several signals on a single line. These methods reduce: 1) hardware; 2) logic levels; and 3) pins. In the TDM realizations, we consider three types of DDs: shared binary decision digrams (SBDDs), shared multiple-valued decision diagrams (SMDDs), and shared multi-terminal multiple-valued decision diagrams (SMTMDDs). In the network, each non-terminal node of a DD is realized by a multiplexer (MUX). We propose heuristic algorithms to derive SMTMDDs from SBDDs. We compare the number of non-terminal nodes in SBDDs, SMDDs, and SMTMDDs. For nrm n, $\log n$, and for many other benchmark functions, SMTMDD-based realizations are more economical than other ones, where nrm n is a $(2n)$-input $(n+1)$-output function computing $\lfloor\sqrt{X^2+Y^2}+0.5\rfloor$, $\log n$ is an n-input n-output function computing $\lfloor\frac{(2^n-1)\log(x+1)}{n\log 2}\rfloor$, and $\lfloor a\rfloor$ denotes the largest integer not greater than a.

key words: *multiple-valued decision diagram (MDD), multiple-valued logic, multiple-output function, time-division multiplexing (TDM)*

1. Introduction

In modern LSIs, one of the most important issue is the "pin problem." The reduction of the number of pins in the LSIs is not so easy, even though the integration of more gates may be possible. To overcome the pin problem, the time-division multiplexing (TDM) systems are often used. In the TDM system, a single signal line represents several signals. For example, the Intel 8088 microprocessors used 8-bit buses to represent 16-bit data which made it possible to produce a large amount of microcomputers so quickly while the 16-bit peripheral LSIs were not so popular in the early 1980s. In this paper, we present a method to design multiple-output networks based on shared multi-terminal multiple-valued decision diagrams (SMTMDDs) by using TDMs. We propose heuristic algorithms to derive SMTMDDs from shared binary decision diagrams (SBDDs), and compare realizations based on SMTMDDs with the ones based on SBDDs and shared multiple-valued decision diagrams (SMDDs). Experimental results show the compactness of SMTMDDs over SBDDs and SMDDs.

Manuscript received September 10, 1998.
Manuscript revised November 11, 1998.

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

In the network, each non-terminal node of a decision diagram (DD) is realized by a multiplexer (MUX). The rest of the paper is organized as follows: Section 2 defines various DDs for multiple-output functions. Section 3 shows TDM realizations of multiple-output functions based on SBDDs, SMDDs, and SMTMDDs. Section 4 presents methods to derive SMTMDDs from SBDDs. Section 5 shows upper bounds on the size of an SBDD and an SMTMDD to represent an n-input m-output function. Experimental results for arithmetic functions and other benchmark functions are shown in Sect. 6.

2. Decision Diagrams for Multiple-Output Functions

In this section, we show three different decision diagrams to represent multiple-output functions.

2.1 Shared Binary Decision Diagrams

A shared binary decision diagram (SBDD) is a set of binary decision diagrams (BDDs) combined by a tree for output selection. Note that the definition of the SBDD in this paper is somewhat different from [16]. For example, Fig. 1 shows the SBDD for Table 1.

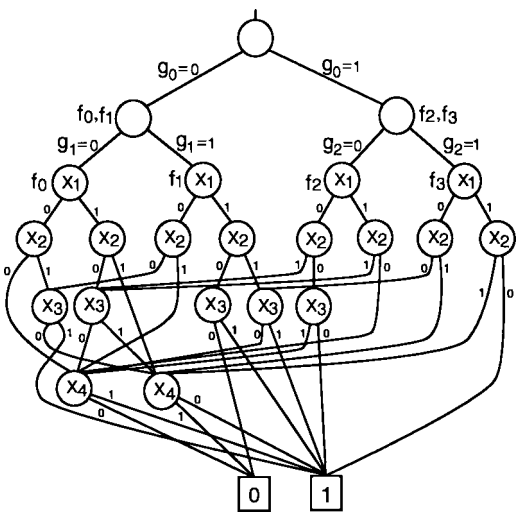


Fig. 1 SBDD for the function in Table 1.

Table 1 2-valued 4-input 4-output function.

Input				Output			
x_1	x_2	x_3	x_4	f_0	f_1	f_2	f_3
0	0	0	0	0	1	1	0
0	0	0	1	1	0	1	1
0	0	1	0	0	1	0	1
0	0	1	1	1	1	1	1
0	1	0	0	1	0	0	1
0	1	0	1	0	1	1	0
0	1	1	0	1	0	1	1
0	1	1	1	1	1	0	0
1	0	0	0	0	0	0	1
1	0	0	1	1	0	1	1
1	0	1	0	1	1	0	1
1	0	1	1	0	1	1	1
1	1	0	0	1	0	0	1
1	1	0	1	0	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	0	1	0	0

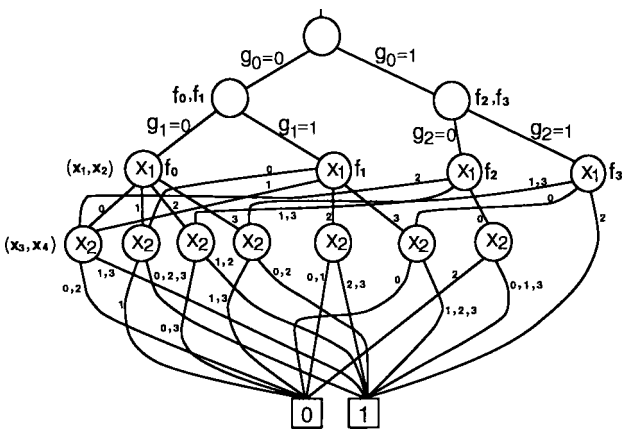


Fig. 2 SMDD for the function in Table 1.

Multi-terminal binary decision diagrams (MTBDDs) are the extended BDDs with multiple terminal nodes, where the terminals are m-bit binary vectors for m output functions. A shared multi-terminal binary decision diagram (SMTBDD) is a set of MTBDDs combined by a tree for output selection [1]–[3], [6], [7], [9], [16].

2.2 Shared Multiple-Valued Decision Diagrams

A shared multiple-valued decision diagram (SMDD) is a set of multiple-valued decision diagrams (MDDs) combined by a tree for output selection [4]. Figure 2 shows the SMDD for Table 1, where g_0, g_1, and g_2 are the output selection variables, and X_1 and X_2 are the pairs of binary inputs.

2.3 Shared Multi-Terminal Multiple-Valued Decision Diagrams

Shared multi-terminal multiple-valued decision diagrams (SMTMDDs) are another representations of multiple-valued multiple-output logic functions [4], [5],

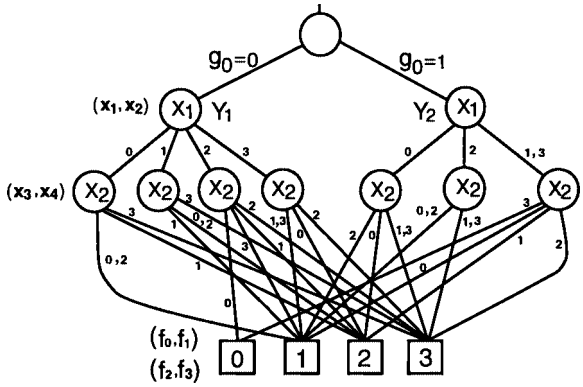


Fig. 3 SMTMDD for the function in Table 2.

Table 2 4-valued 2-input 2-output function.

Input		Output	
X_1	X_2	Y_1	Y_2
0	0	1	2
0	1	2	3
0	2	1	1
0	3	3	3
1	0	2	1
1	1	1	2
1	2	2	3
1	3	3	0
2	0	0	1
2	1	2	3
2	2	3	1
2	3	1	3
3	0	2	1
3	1	1	2
3	2	3	3
3	3	1	0

[8], [10]–[15]. An SMTMDD is a set of multiple-valued decision diagrams (MDDs) with multiple terminal nodes combined by a tree for output selection. The number of MDDs in the SMTMDD is equal to the number of groups of output functions. Figure 3 shows the SMTMDD for Table 2, where Y_1 and Y_2 are the pairs of binary outputs, and X_1 and X_2 are the pairs of binary inputs. The advantage of SMTMDDs is that they can evaluate several output functions simultaneously. Moreover, the good grouping of output functions and good grouping of input variables produce compact SMTMDDs.

Definition 1: The **size$_n$ of the DD** denoted by size$_n$(DD), is the total number of non-terminal nodes excluding the nodes for output selection variables.

Example 1: The size$_n$s of the SBDD, SMDD, and SMTMDD in Figs. 1, 2, and 3 are 19, 11, and 9, respectively. Note that g_0, g_1, and g_2 are the output selection variables in the SBDD and SMDD, and g_0 is the output selection variable in the SMTMDD. □

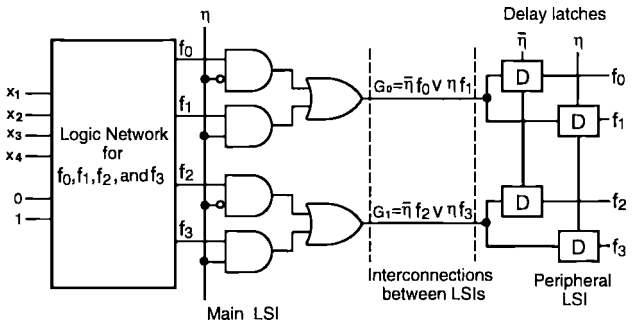


Fig. 4 TDM realization based on the SBDD.

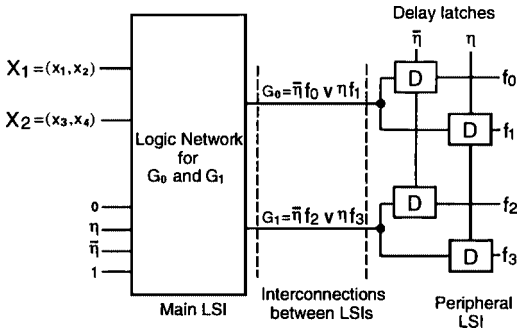


Fig. 5 TDM realization based on the SMTMDD.

3. TDM Realizations

The TDM realization uses clock pulse to reduce the number of input and output pins. On the other hand, the non-TDM realization means a conventional combinational network without using clock pulse. In this section, we will show TDM realizations of multiple-output functions based on SBDDs, SMDDs, and SMTMDDs.

3.1 TDM Realizations Based on SBDDs

In this part, we introduce a method to realize multiple-output functions by using TDM. To illustrate it, we use an example of the 4-input 4-output function shown in Table 1. Figure 4 shows a TDM realization. In the main LSI, pairs of logic functions are multiplexed by the clock pulse η. The output signals of the main LSI denote the functions as follows:

$$G_0 = \bar{\eta} f_0 \vee \eta f_1, \text{ and}$$
$$G_1 = \bar{\eta} f_2 \vee \eta f_3.$$

These mean when $\eta = 0$, G_0 and G_1 represent f_0 and f_2, respectively. On the other hand, when $\eta = 1$, G_0 and G_1 represent f_1 and f_3, respectively. In this realization, we need the hardware for the functions f_0, f_1, f_2, and f_3, as well as the hardware for multiplexing. By using this technique, we can reduce the number of output pins into a half. Note that in this example, only two lines are necessary between the main LSI and the peripheral LSI. In the peripheral LSI, we need delay latches. When $\eta = 0$, the values for f_0 and f_2 are transferred to the first and the third latches, respectively. On the other hand, when $\eta = 1$, the values for f_1 and f_3 are transferred to the second and fourth latches, respectively. To realize the multiple-output function, we use an SBDD. By replacing each non-terminal node of an SBDD by a multiplexer, we obtain a network for the multiple-output function. In this case, the amount of hardware for the network is easily estimated by the $size_n$ of the SBDD, and the design of the network is quite easy.

3.2 TDM Realizations Based on SMDDs

In Fig. 4, if we realize the multiple-output function by an SMDD, we have the TDM realization based on SMDDs. Each non-terminal node of an SMDD is realized by a 2-MUX in Fig. 6. Figure 7 shows the literal generator whose inputs are a pair of 2-valued variables (details will be shown in Sect. 3.3). In this method, the input variables are partitioned into pairs to make 4-valued variables, and we consider the realization of a 4-valued input 2-valued output function: $Q^n \rightarrow B^m$, where $Q = \{00, 01, 10, 11\}$ and $B = \{0, 1\}$. The numbers of nodes in an SMDD can be reduced as follows:

- By finding the best pairing of the input variables to make 4-valued variables.
- By finding the best ordering of the 4-valued variables.

3.3 TDM Realizations Based on SMTMDDs

The TDM realization based on SMTMDDs is shown in Fig. 5. In this method, 4-valued logic is used instead of 2-valued logic. Consider a 2-valued multiple-output function. First, partition the input variables into pairs. For example, the input variables $\{x_1, x_2, x_3, x_3\}$ in Table 1 are partitioned into $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4)$. Second, partition the output functions into pairs. For example, the output functions $\{f_0, f_1, f_2, f_3\}$ in Table 1 are partitioned into $G_0 = (f_0, f_1)$ and $G_1 = (f_2, f_3)$. Then, we have a 4-valued logic function: $Q^2 \rightarrow Q$, where $Q = \{0, 1, 2, 3\}$, as shown in Table 2. The output functions Y_1 and Y_2 in Table 2 correspond to G_0 and G_1, respectively. In general, a 4-valued n-input m-output function: $Q^n \rightarrow Q^m$ is represented by an SMTMDD. Next, consider the hardware realization of an SMTMDD. Each non-terminal node of an SMTMDD is realized by a 2-MUX shown in Fig. 6. It is a 4-way multiplexer. Figure 7 shows the literal generator whose inputs are a pair of 2-valued variables (x_1, x_2), and outputs are X^0, X^1, X^2, and X^3 that control the 2-MUX. Note that

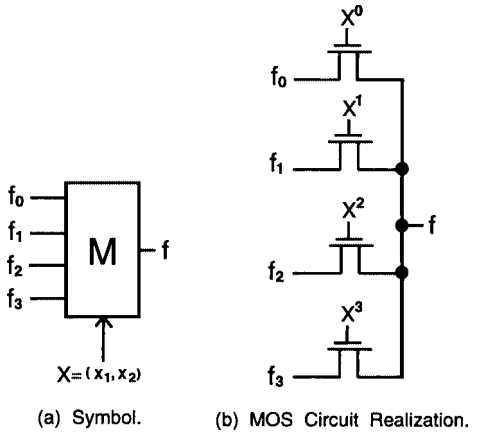


Fig. 6 2-MUX.

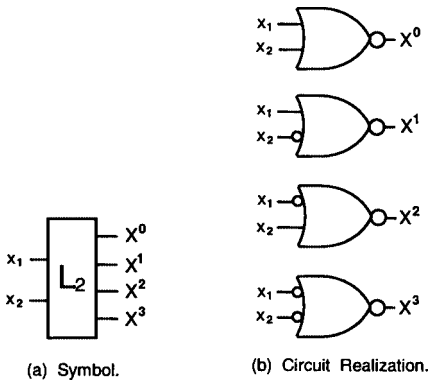


Fig. 7 Literal generator.

$$X^i = \begin{cases} 0 & \text{if } X \neq i, \\ 1 & \text{if } X = i. \end{cases}$$

A signal in the terminal node is represented by a pair of bits (c_0, c_1) as follows:

When $\eta = 0$, the signal represents c_0.
When $\eta = 1$, the signal represents c_1.

Thus,

$(c_0, c_1) = (0, 0)$ corresponds to a constant 0.
$(c_0, c_1) = (0, 1)$ corresponds to η.
$(c_0, c_1) = (1, 0)$ corresponds to $\bar{\eta}$.
$(c_0, c_1) = (1, 1)$ corresponds to a constant 1.

Figure 8 shows the SMTMDD-based TDM realization for the function in Table 2. In the inputs, a pair of 2-valued variables $X = (x_1, x_2)$ represents a 4-valued signal $\{00, 01, 10, 11\}$ or $\{0, 1, 2, 3\}$. On the other hand, 0, η, $\bar{\eta}$, and 1, represent $(0, 0)$, $(0, 1)$, $(1, 0)$, and $(1, 1)$, respectively. Note that $\{0, \eta, \bar{\eta}, 1\}$ constitutes the 4-element Boolean algebra. If we replace $\{0, \eta, \bar{\eta}, 1\}$ by $\{0, 1, 2, 3\}$, then we have the 4-valued function in Table 2. An arbitrary 4-valued function is represented by an SMTMDD. The amount of hardware for the network is estimated by the $size_n$ of the SMTMDD. The

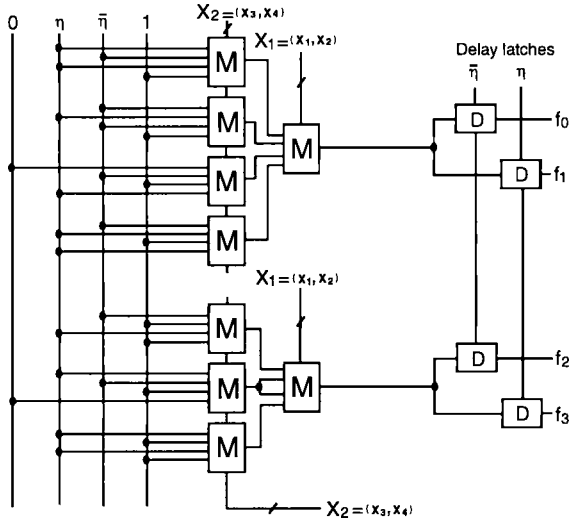


Fig. 8 TDM realization of a 4-output function based on the SMTMDD.

number of nodes in an SMTMDD can be reduced as follows:

- By finding the best pairing of the input variables to make 4-valued variables.
- By finding the best pairing of the output functions to make 4-valued functions.

3.4 Comparison of TDM Realizations

In this part, we compare the DD-based TDM realizations of an n-input m-output function F. In the hardware realization, each non-terminal node of a BDD is realized with two MOS transistors, while each non-terminal node of an MDD is realized with four MOS transistors. So, if we ignore the cost of literal generators, then the cost of a non-terminal node of an MDD is twice the cost of a non-terminal node of the BDD. Therefore, when $(2size_n(MDD : F) < size_n(BDD : F))$, the MDD-based realizations are more economical than BDD-based ones. In addition, in the case of an n-variable function, a BDD-based realization requires n levels, while an MDD-based realization requires only $\frac{n}{2}$ levels. In the FPGAs, the delay of interconnections between the modules is often greater than the delay of logic modules. Thus, the reduction of logic level is important. So, MDD-based realizations can be faster and require smaller amount of hardware than BDD-based ones.

4. Reduction of SMTMDDs

The reduction of $size_n$ for SMTMDDs is important to design compact logic networks. We consider the following methods for reduction: 1) pairing of output functions; 2) pairing of input variables; and 3) ordering of group of input variables. The SMTBDDs

are derived from the SBDDs by pairing the output functions, and the SMTMDDs are derived from the SMTBDDs by pairing the input variables. Since an SMTBDD consists of MTBDDs, and each MTBDD represents a pair of output functions, we used the following heuristics to pair the outputs: Pair output functions so that the upper bounds on the size of the MTBDD are minimized [9]. The MDD nodes for each pair of input variables are counted from SMTBDDs as follows: Subgraphs shown in Figs. 9 (a), (b), or (c), correspond to one, two, or three MDD nodes, respectively. In Fig. 9 (a), three SMTBDD nodes are replaced by one MDD node. However, in Fig. 9 (b), the SMTBDD nodes correspond to two MDD nodes. In Fig. 9 (c), the SMTBDD nodes are replaced by three MDD nodes. Finally, SMTMDDs are optimized by using sifting algorithm [17].

5. Upper Bounds on the Size$_n$ of DDs

In the design of multiple-output networks, we often have to estimate the number of MUXs to realize functions. This section shows upper bounds on the number of non-terminal nodes to represent an n-input m-output function by an SBDD and an SMTMDD. Since each non-terminal node of a DD corresponds to an MUX, the $size_n$ of the DD estimates the amount of hardware.

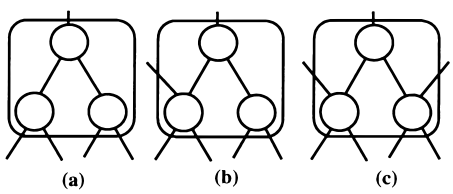


Fig. 9 A method replacing SMTBDD nodes by MDD nodes.

Theorem 1: Consider an n-input m-output function F. Then, $size_n(SBDD) \leq \min_{k=1}^{n}\{m\cdot(2^{n-k}-1)+2^{2^k}-2\}$.

Theorem 2: Consider a function $F : \{0,1,\ldots, p-1\}^N \rightarrow \{0,1,\ldots,r-1\}^m$. Let m_1 be the number of groups of 2-valued functions. Then, $size_n(SMTMDD) \leq \min_{k=1}^{N}\{m_1 \cdot \frac{p^{N-k}-1}{(p-1)} + r^{p^k} - r\}$.

Example 2: Let $n = 18$, $m = 20$, and $p = r = 4$. For such a function, $size_n(SBDD) \leq 2621422$, and $size_n(SMTMDD) \leq 218702$. □

Note that these theorems are used in the heuristic algorithm in Sect. 4.

6. Experimental Results

We developed C programs to build SBDDs and SMTBDDs. SMTMDDs were constructed from SMTBDDs using the method in Sect. 4, while SMDDs were constructed from SBDDs using the similar technique to Fig. 9 [4]. Tables 3, 4, and 5 compare the size$_n$s of SBDDs, SMDDs, and SMTMDDs for nrm n, $\log n$, and n-bit adders, respectively, where nrm n is a $(2n)$-input $(n+1)$-output function computing $\lfloor\sqrt{X^2+Y^2}+0.5\rfloor$, $\log n$ is an n-input n-output function computing $\lfloor\frac{(2^n-1)\log(x+1)}{n\log 2}\rfloor$, and $\lfloor a\rfloor$ denotes the largest integer not greater than a [7]. Table 6 compares the size$_n$s of SBDDs, SMDDs, and SMTMDDs for other benchmark functions. The symbol " * " in Table 6 denotes the function with ***don't cares***, where the ***don't cares*** were set to zero during the experiment. The ***ratio***[1]s showing $size_n$s of SMDDs to SBDDs are in the column 7 of Tables 3–6, while the ***ratio***[2]s showing $size_n$s of SMTMDDs to SBDDs are in the column 8 of Tables 3–6. The ratios in these tables show the relative

Table 3 Numbers of non-terminal nodes in SBDDs, SMDDs, and SMTMDDs to represent nrm n.

Function name	In	Out	SBDD	SMDD	SMT-MDD	ratio[1]	ratio[2]
nrm3	6	4	45	25	21	0.55	0.46
nrm4	8	5	152	81	70	0.53	0.46
nrm5	10	6	471	241	214	0.51	0.45
nrm6	12	7	1345	684	618	0.50	0.45
nrm7	14	8	3859	1590	1463	0.41	0.37
average[3] relative size$_n$			1.00			0.50	0.43

In: number of inputs.
Out: number of outputs.

1. ratio = $\frac{size_n(SMDD)}{size_n(SBDD)}$;

2. ratio = $\frac{size_n(SMTMDD)}{size_n(SBDD)}$;

3. average relative size$_n$ = $\frac{1}{N_1}\sum_{i=1}^{N_1}\frac{size_n \text{ of the MDD for function } i}{size_n \text{ of the SBDD for function } i}$,

where N_1 is the total number of functions.

Table 4 Numbers of non-terminal nodes in SBDDs, SMDDs, and SMTMDDs to represent $\log n$.

Function name	In	Out	SBDD	SMDD	SMT-MDD	ratio[1]	ratio[2]
log 6	6	6	56	34	33	0.60	0.58
log 8	8	8	196	108	106	0.55	0.54
log 10	10	10	585	296	277	0.50	0.47
log 12	12	12	1697	851	780	0.50	0.45
log 14	14	14	4833	2365	2147	0.48	0.44
average[3] relative size$_n$			1.00			0.52	0.49

Table 5 Numbers of non-terminal nodes in SBDDs, SMDDs, and SMTMDDs to represent n-bit adders.

Function name	In	Out	SBDD	SMDD	SMT-MDD	ratio[1]	ratio[2]
adr3	6	4	20	8	9	0.40	0.45
adr4	8	5	29	11	14	0.37	0.48
adr5	10	6	38	14	19	0.36	0.50
adr6	12	7	47	17	24	0.36	0.51
adr7	14	8	56	20	29	0.35	0.51
average[3] relative size$_n$			1.00			0.36	0.49

Table 6 Numbers of non-terminal nodes in SBDDs, SMDDs, and SMTMDDs to represent various benchmark functions.

Function name	In	Out	SBDD	SMDD	SMT-MDD	ratio[1]	ratio[2]
al2	16	47	97	81	77	0.83	0.79
amd	14	28	256	154	187	0.60	0.73
apex4	9	19	970	465	516	0.47	0.53
apex5	117	88	1078	590	563	0.54	0.52
apla*	10	12	102	61	66	0.59	0.64
bc0	26	11	578	357	435	0.61	0.75
cordic	23	2	75	41	30	0.54	0.40
cps	24	109	985	678	792	0.68	0.80
c432	36	7	1298	749	731	0.57	0.56
dk17*	10	11	83	38	48	0.45	0.57
exp*	8	18	193	127	105	0.65	0.54
ex1010*	10	10	1410	811	648	0.57	0.45
in0	15	11	278	137	169	0.49	0.60
max1024	10	6	301	159	145	0.52	0.48
misex1	8	7	36	22	24	0.61	0.66
misex3	14	14	542	313	290	0.57	0.53
pdc	16	40	568	350	326	0.61	0.57
prom1	9	40	1971	961	928	0.48	0.47
prom2	9	21	936	483	465	0.51	0.49
rd84	8	4	59	24	24	0.40	0.40
rot10	10	6	159	79	55	0.49	0.34
sao2	10	4	85	46	35	0.54	0.41
shift	19	16	61	45	47	0.73	0.77
sqr6	6	12	71	40	33	0.56	0.46
sqr8	8	16	233	123	130	0.52	0.55
ts10	22	16	146	67	58	0.45	0.39
x4	94	71	370	395	506	1.06	1.36
average[3] relative size$_n$			1.00			0.57	0.58

*function with ***don't cares***

sizes of SMDDs and SMTMDDs to SBDDs. Note that the numbers of terminal nodes in SBDDs, SMDDs, and SMTMDDs are at most 2, 2, and 4, respectively. We used sifting algorithm for input variables to reduce the size$_n$s of DDs [17]. In addition, we used algorithms in Sect. 4 to optimize SMTMDDs. Tables 3–6 show that,

in most cases,

$$size_n(SMTMDD) < size_n(SBDD), \text{ and}$$
$$size_n(SMDD) < size_n(SBDD).$$

The exception is x4 in Table 6.

Tables 3 and 4 show that, for nrm n $(3 \leq n \leq 7)$, and for $\log n$ $(6 \leq n \leq 14)$,

$$size_n(SMTMDD) < size_n(SMDD).$$

These tables also show that, for nrm n $(3 \leq n \leq 7)$, and for $\log n$ $(10 \leq n \leq 14)$,

$$size_n(SMTMDD) < \frac{1}{2} size_n(SBDD).$$

Table 5 shows that, for adr3 and adr4,

$$size_n(SMTMDD) < \frac{1}{2} size_n(SBDD).$$

It also shows that, for adr n $(3 \leq n \leq 7)$,

$$size_n(SMDD) < \frac{1}{2} size_n(SBDD), \text{ and}$$
$$size_n(SMDD) < size_n(SMTMDD).$$

Table 6 shows that, for cordic, ex1010, max1024, prom1, prom2, rd84, rot10, sao2, sqr6, and ts10,

$$size_n(SMTMDD) < \frac{1}{2} size_n(SBDD).$$

It also shows that, for al2, apex5, c432, cordic, exp, ex1010, max1024, misex3, pdc, prom1, prom2, rot10, sao2, sqr6, and ts10,

$$size_n(SMTMDD) < size_n(SMDD).$$

Table 5 shows that, the size$_n$s of SBDDs, SMDDs, and SMTMDDs for n-bit adders $(3 \leq n \leq 7)$ are $(9n-7)$, $(3n-1)$, and $(5n-6)$, respectively. The bottom rows of the tables show the ***average relative size***$_n$s for SBDDs, SMDDs, and SMTMDDs. Note that apex5 is one of the most complex benchmark functions in Table 6, and our program required 1.5 seconds to construct the SMTMDD from the SBDD on a JU1/170 with 160 MB of main memory (Sun Ultra1-170 compatible workstation).

7. Conclusions and Comments

In this paper, we considered time-division multiplexing (TDM) realizations of multiple-output functions based on shared binary decision diagrams (SBDDs), shared multiple-valued decision diagrams (SMDDs), and shared multi-terminal multiple-valued decision diagrams (SMTMDDs). In the network, each non-terminal node of a decision diagram (DD) is realized by a multiplexer (MUX). For an n-variable function, the BDD-based realization requires n levels, while the MDD-based realization requires $\frac{n}{2}$ levels. The TDM method reduces the interconnections among the modules as shown in Figs. 4 and 5. In addition, the SMTMDD-based realization reduces the number of gates by considering the pairing of input variables and pairing of output functions. Note that SBDD-based realizations and SMDD-based realizations require extra gates in the outputs (which are not included in the tables). The TDM method requires clock pulse that makes delay in the network. However, the number of pins in the TDM realization is a half of the non-TDM realization. MDD-based realizations are more economical than SBDD-based ones when the ratios are less than 0.5. Experimental results showed that, for nrm n and $\log n$, SMTMDD-based realizations require the smallest amount of hardware. However, for n-bit adders, SMDD-based realizations require the smallest amount of hardware. We also showed that there are cases where SBDD-based realizations are the most economical. For arithmetic functions, MDD-based realizations tend to be more economical than SBDD-based ones. The presented method can be extended to the case where p output functions are grouped by using p-phase clock pulses.

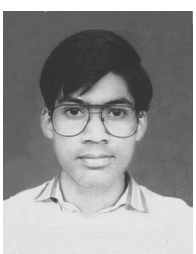
Acknowledgements

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. The authors thank to the constructive comments of the reviewers. A preliminary version of this paper was presented at the 28th IEEE International Symposium on Multiple-Valued Logic, Fukuoka, Japan [10].

References

[1] S.B. Akers, "Binary decision diagrams," IEEE Trans. Comput., vol.C-27, no.6, pp.509–516, June 1978.
[2] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Trans. Comput., vol.C-35, no.8, pp.677–691, Aug. 1986.
[3] 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.
[4] T. Sasao and J.T. Butler, "A method to represent multiple-output switching functions by using multi-valued decision diagrams," Proc. IEEE International Symposium on Multiple-Valued Logic, pp.248–254, May 1996.
[5] T. Sasao and J.T. Butler, "A design method for look-up table type FPGA by pseudo-Kronecker expansion," Proc. IEEE International Symposium on Multiple-Valued Logic, pp.97–106, May 1994.
[6] T. Sasao, ed., "Logic Synthesis and Optimization," Kluwer Academic Publishers, Boston, 1993.
[7] T. Sasao, "Switching Theory for Logic Synthesis," Kluwer Academic Publishers, Boston, 1999.
[8] G. Epstein, "Multiple-Valued Logic Design: An Introduction," IOP Publishing Ltd., London, 1993.
[9] H.Md. Hasan Babu and T. Sasao, "Shared multi-terminal binary decision diagrams for multiple-output functions,"

IEICE Trans. Fundamentals, vol.E81-A, no.12, pp.2545–2553, Dec. 1998.
[10] H.Md. Hasan Babu and T. Sasao, "Design of multiple-output networks using time domain multiplexing and shared multi-terminal multiple-valued decision diagrams," Proc. IEEE International Symposium on Multiple-Valued Logic, pp.45–51, May 1998.
[11] L.S. Hurst, "Multiple-valued logic—Its status and its future," IEEE Trans. Comput., vol.C-33, no.12, pp.1160–1179, Dec. 1984.
[12] M. Kameyama and T. Higuchi, "Synthesis of multiple-valued logic networks based on tree-type universal logic module," IEEE Trans. Comput., vol.C-26, no.12, pp.1297–1302, Dec. 1977.
[13] D.M. Miller, "Multiple-valued logic design tools," Proc. IEEE International Symposium on Multiple-Valued Logic, pp.2–11, May 1993.
[14] D.M. Miller and N. Muranaka, "Multiple-valued decision diagrams with symmetric variable nodes," Proc. IEEE International Symposium on Multiple-Valued Logic, pp.242–247, May 1996.
[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] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation," Proc. 27th ACM/IEEE DAC, pp.52–57, June 1990.
[17] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," Proc. International Conference on Computer-Aided Design, pp.42–47, Nov. 1993.



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 has been 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 many books on logic design including, "Logic Synthesis and Optimization," "Representation of Discrete Functions," and "Switching Theory for Logic Synthesis," Kluwer Academic Publishers 1993, 1996, and 1999, 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.