PAPER Special Section on VLSI Design and CAD Algorithms

Heuristics to Minimize Multiple-Valued Decision Diagrams

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

SUMMARY In this paper, we propose a method to minimize multiple-valued decision diagrams (MDDs) for multiple-output functions. We consider the following: (1) a heuristic for encoding the 2-valued inputs; and (2) a heuristic for ordering the multiple-valued input variables based on sampling, where each sample is a group of outputs. We first generate a 4-valued input 2-valued multiple-output function from the given 2-valued input 2-valued functions. Then, we construct an MDD for each sample and find a good variable ordering. Finally, we generate a variable ordering from the orderings of MDDs representing the samples, and minimize the entire MDDs. Experimental results show that the proposed method is much faster, and for many benchmark functions, it produces MDDs with fewer nodes than sifting. Especially, the proposed method generates much smaller MDDs in a short time for benchmark functions when several 2-valued input variables are grouped to form multiple-valued variables.

key words: *binary decision diagram (BDD), multiple-valued decision diagram (MDD), multiple-output function, multiple-valued logic, FPGA design*

1. Introduction

Multiple-valued decision diagrams (MDDs) are data structures for multiple-valued functions. MDDs are extensions of binary decision diagrams (BDDs) and usually require fewer nodes than the corresponding BDDs to represent the same logic functions [1]–[4], [7]–[10]. MDDs are useful for logic synthesis, FPGA design, logic simulation, etc. [2], [3], [7]. For example, Fig. 2 shows the multiplexer-based network corresponding to the MDD in Fig. 1. In this paper, we consider multi-rooted MDDs to represent multiple-output functions. From 2-valued input 2-valued output functions, we construct MDDs to represent 4-valued input 2-valued output functions. We use a shared binary decision diagram (SBDD) [4] for a multiple-output function to find good pairs of 2-valued input variables. Since the size of a decision diagram (DD) can vary from linear to exponential due to the orderings of the input variables, finding a good variable ordering of the input variables is very important [6], [11]–[14]. Dynamic variable ordering [11] is one of the good heuristics to order the inputs. However, in the case of a multiple-output function, a set of output functions must be handled at the same time. So, generating a good variable ordering that represents all the output functions compactly is essential. ***Sampling based*** variable ordering methods [13] and ***Interleaving based*** variable ordering methods [12] are effective to find good variable orderings for multiple-output functions quickly. In this paper, we combine both methods to find good orderings of input variables. Experimental results show the effectiveness of our approach. The rest of the paper is organized as follows: Sect. 2 presents basic definitions. Section 3 defines MDDs, and presents their properties. Section 4 shows the minimization method of MDDs. Finally, Sect. 5 presents experimental results.

Manuscript received March 25, 2000.
Manuscript revised June 23, 2000.

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

2. Basic Definitions

This section presents notation and basic definitions.

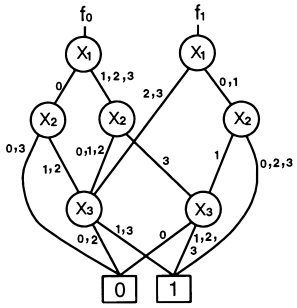

Fig. 1 Example of an MDD.

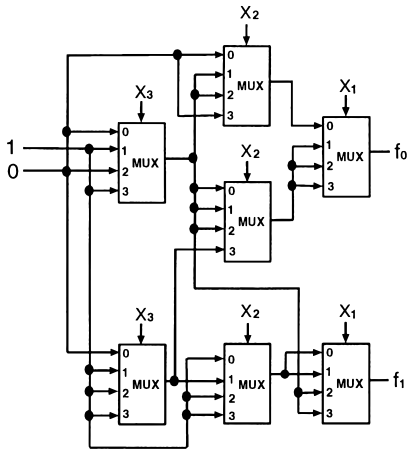

Fig. 2 Multiplexer-based network corresponding to the MDD in Fig. 1.

Definition 1: Let $F_1 = \{f_0, f_1, \ldots, f_{m-1}\}$, $R = \{0, 1, \ldots, r-1\}$, and $B = \{0, 1\}$. ***An r-valued input 2-valued output function*** F_1 is a mapping

$$F_1 : R^N \rightarrow B^m. \quad \square$$

Definition 2: Let $R = \{0, 1, \ldots, r-1\}$ and $S \subseteq R$. 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 literals. 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. $\square$

Lemma 1: An arbitrary r-valued input 2-valued multiple-output function can be represented as $F_1(X_1, X_2, \ldots, X_N) = X_1^0 F_1(0, X_2, \ldots, X_N) \vee X_1^1 F_1(1, X_2, \ldots, X_N) \vee \cdots \vee X_1^{r-1} F_1(r-1, X_2, \ldots, X_N)$. This is a ***multiple-valued version of Shannon's expansion*** with respect to X_1. $\square$

Definition 3: Let $F = \{f_0, f_1, \ldots, f_{m-1}\}$. Then, the ***size of a decision diagram (DD) for F***, denoted by $size(DD, F)$, is the total number of non-terminal nodes.

Example 1: The size of the MDD in Fig. 1 is 7. $\square$

3. Multiple-Valued Decision Diagrams

Let $F_1 : \{0, 1, \ldots, r-1\}^N \rightarrow \{0, 1\}^m$. A multiple-valued decision diagram (MDD) for $F_1(X_1, X_2, \ldots, X_N)$ is a multi-rooted directed graph that has r outgoing edges labeled $0, 1, \ldots,$ and $r-1$ directed to nodes representing $F_1(0, X_2, \ldots, X_N)$, $F_1(1, X_2, \ldots, X_N), \ldots,$ and $F_1(r-1, X_2, \ldots, X_N)$, respectively. Each of these nodes has r outgoing edges which go to nodes that have r outgoing edges, etc. A terminal node is a node that has no outgoing edges. It is labeled by 0 or 1 which corresponds to a binary value of the function F_1. A reduced ordered MDD (ROMDD) has no node where all r outgoing edges point to the same node and has no equivalent subgraphs. From now on we simply refer to an ROMDD as an MDD. Figure 1 is an example of an MDD.

3.1 Size of MDDs

The size of MDDs is an important characteristic. In this part, we present some upper bounds on the sizes of MDDs for various functions.

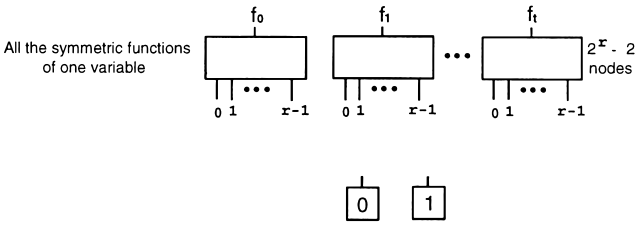

Fig. 3 Realization of all the symmetric functions of a single variable.

Theorem 1: Let $R = \{0, 1, \ldots, r-1\}$ and $B = \{0, 1\}$. Then, the size of the MDD for an N-input m-output function $R^N \rightarrow B^m$ is at most $\min_{k=1}^{N} \left\{ m \cdot \frac{r^{N-k}-1}{r-1} + 2^{r^k} - 2 \right\}$. $\square$

In the case of $r = 2$, we use an SBDD to represent an n-input m-output function, and we have the following:

Corollary 1: The size of the SBDD for an n-input m-output function $B^n \rightarrow B^m$ is at most $\min_{k=1}^{n} \left\{ m \cdot \left(2^{n-k} - 1\right) + 2^{2^k} - 2 \right\}$. $\square$

Lemma 2: Let $R = \{0, 1, \ldots, r-1\}$ and $B = \{0, 1\}$. Then, all the non-constant symmetric functions $R^N \rightarrow B$ can be represented by MDDs with $\sum_{i=1}^{N} \left[2^{\binom{i+r-1}{i}} - 2 \right]$ non-terminal nodes.

Proof: The number of non-constant symmetric functions $f : R^N \rightarrow B$ is $2^{\binom{N+r-1}{N}} - 2$, where $R = \{0, 1, \ldots, r-1\}$ and $B = \{0, 1\}$.
(1) When $N = 1$, there are $2^r - 2$ symmetric functions, and they are realized as shown in Fig. 3.
(2) Suppose that all the non-constant symmetric functions of $(N-1)$ variables are realized with $\sum_{i=1}^{N-1} \left[2^{\binom{i+r-1}{i}} - 2 \right]$ non-terminal nodes. Note that an arbitrary symmetric function of N variables is represented as follows:

$$f(X_1, X_2, \ldots, X_N) = X_N^0 f_0 \vee X_N^1 f_1 \vee \cdots \vee X_N^{r-1} f_{r-1},$$

where $f_j = f(X_1, X_2, \ldots, X_{N-1}, j)$ is a symmetric function of $(N-1)$ variables, and $j = 0, 1, \ldots, r-1$. Thus, all the non-constant symmetric functions of N variables are realized as shown in Fig. 4. The total number of non-terminal nodes in Fig. 4 is

$$\sum_{i=1}^{N-1} \left[2^{\binom{i+r-1}{i}} - 2 \right] + \left[2^{\binom{N+r-1}{N}} - 2 \right] = \sum_{i=1}^{N} \left[2^{\binom{i+r-1}{i}} - 2 \right].$$

Thus, from (1) and (2), we have the lemma. $\square$

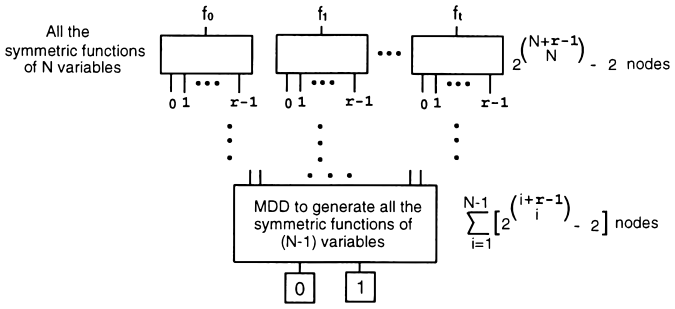

Fig. 4 Realization of symmetric functions of N variables.

Theorem 2: Let $R = \{0, 1, \ldots, r-1\}$ and $B = \{0, 1\}$. Then, the size of the MDD for an N-input m-output symmetric function $R^N \rightarrow B^m$ is at most $\min_{k=1}^{N} \left\{ m \cdot \sum_{i=0}^{k} \binom{i+r-1}{i} + \sum_{i=1}^{N-k} \left[2^{\binom{i+r-1}{i}} - 2 \right] \right\}$.

Proof: Since functions are completely symmetric, the different number of k-variable functions generated by the r-valued complete decision tree is equal to the number of ways to select k objects from r distinct objects with repetition. The number of ways to select k objects from r distinct objects with repetition is $\binom{k+r-1}{k}$. So, the total number of non-terminal nodes in the r-valued decision trees of m functions is $m \cdot \sum_{i=0}^{k} \binom{i+r-1}{i}$. By Lemma 2, in the $N-k$ variables, there are $2^{\binom{N-k+r-1}{N-k}} - 2$ non-constant symmetric functions, and they require $\sum_{i=1}^{N-k} \left[2^{\binom{i+r-1}{i}} - 2 \right]$ non-terminal nodes. Therefore, the size of the MDD for an r-valued N-input 2-valued m-output symmetric function is at most $\min_{k=1}^{N} \left\{ m \cdot \sum_{i=0}^{k} \binom{i+r-1}{i} + \sum_{i=1}^{N-k} \left[2^{\binom{i+r-1}{i}} - 2 \right] \right\}$. □

In the case of $r = 2$, we use an SBDD to represent an n-input m-output symmetric function, and we have the following:

Corollary 2: The size of the SBDD for an n-input m-output symmetric function $B^n \rightarrow B^m$ is at most $\min_{k=1}^{n} \left\{ m \cdot \frac{(k+1)(k+2)}{2} + 2^{n-k+2} - 2(n-k) - 4 \right\}$. □

From now on we assume that an MDD represents a 4-valued input 2-valued multiple-output function, where $r = 4$.

Definition 4: Let ***inc*** n be an n-input $(n+1)$-output function that computes $K+1$, where K is a binary number consisting of n bits. It represents an incrementing circuit. □

Theorem 3: Suppose that the 2-valued input variables of inc n are paired as $X_1 = [x_1, x_2]$, $X_2 = [x_3, x_4]$, …, and $X_N = [x_{n-1}, x_n]$, where $n = 2N$ and the variable ordering of the 4-valued inputs is $(X_1, X_2, \ldots, X_N)$. Then, size (MDD, inc n) $\leq 2n - 1$ $(n \geq 2)$.

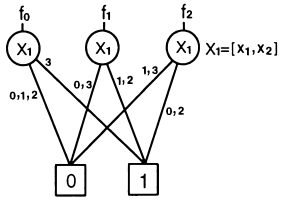

Fig. 5 MDD for inc 2.

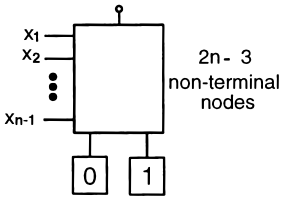

Fig. 6 MDD for inc $(n-1)$.

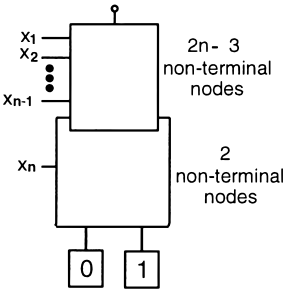

Fig. 7 MDD for inc n.

Proof: We use mathematical induction on the number of 2-valued input variables.
(1) Base: For $n = 2$, the MDD for inc 2 is realized with three non-terminal nodes as shown in Fig. 5.
(2) Induction: Assume that the hypothesis is true for $k = n - 1$ input variables. That is, the MDD for inc $(n-1)$ is realized as Fig. 6 with $2n - 3$ non-terminal nodes. In Fig. 6, first remove the constant 0 and constant 1. Second, insert the input variable x_n and add two non-terminal nodes, as well as nodes for constant 0 and constant 1. Then, we have the diagram in Fig. 7. Note that Fig. 7 shows the MDD for inc n with $2n - 1$ non-terminal nodes which has two more non-terminal nodes than Fig. 6. It is clear that the MDD in Fig. 7 has upper and lower parts: When n is even, x_n is paired with x_{n-1} and two additional non-terminal nodes are added at the level in the bottom of the upper part of the MDD. On the other hand, when n is odd, x_n remains as a 2-valued variable in the lower part of the MDD which requires two non-terminal nodes. Note that $x_1, x_2, \ldots, x_n$ is the order of the 2-valued inputs in the pairs, and $(X_1, X_2, \ldots, X_N)$ is the variable ordering

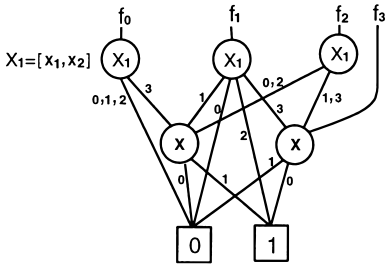

Fig. 8 MDD for inc 3.

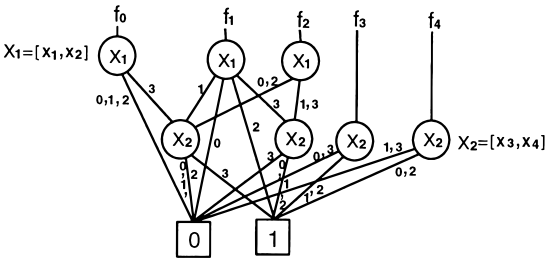

Fig. 9 MDD for inc 4.

of the 4-valued inputs in the MDD. Thus, from (1) and (2), we have the theorem. □

Example 2: Figures. 8 and 9 show the MDDs for inc 3 and inc 4, respectively. The sizes of MDDs in Figs. 8 and 9 are 5 and 7, respectively. □

In the case of $r = 2$, we use an SBDD to represent an n-input $(n+1)$-output inc n, and we have the following:

Corollary 3: $size(SBDD, inc\ n) \leq 3n - 2$. □

4. Minimization of MDDs

The pairing of 2-valued input variables as well as the ordering of multiple-valued variables are important to reduce the number of nodes in MDDs. In this section, we present heuristics to minimize MDDs.

4.1 Pairing of 2-Valued Inputs

When a function has only a single-output, finding good pairs of 2-valued inputs is relatively easy. However, for a multiple-output function, finding good pairs of 2-valued inputs is not so easy. In this part, we present a heuristic to select good pairs of 2-valued inputs from an SBDD.

Algorithm 1: (Pairing the input variables)

1. Let $F_2 : \{0,1\}^n \rightarrow \{0,1\}^m$. Construct an SBDD for F_2.
2. Let $s(x_i, x_j)$ be the number of outputs that depend on either one or both of the input variables x_i and x_j. Then, $[x_i, x_j]$ is a candidate pair of inputs if $s(x_i, x_j)$ is the smallest among all the pairs. Apply the same idea to the rest of the inputs recursively to find good pairs of input variables. In the case of tie, use Step 3 to find the best one among them.
3. $[x_i, x_j]$ is a good pair of inputs in F_2 if in the SBDD, most of the incoming edges into nodes labeled x_j are from nodes labeled x_i.

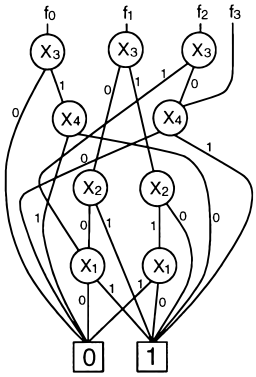

Fig. 10 SBDD for finding the pairs of 2-valued inputs.

Example 3: Consider the SBDD in Fig. 10, where $s(x_1, x_2) = 2$, $s(x_1, x_3) = s(x_2, x_3) = 3$, and $s(x_1, x_4) = s(x_2, x_4) = s(x_3, x_4) = 4$. $[x_1, x_2]$ is a good pair of input variables, since $s(x_1, x_2)$ is the smallest among all $s(x_i, x_j)$. The remaining inputs are x_3 and x_4. Thus, $[x_3, x_4]$ is the another pair. Therefore, $[x_1, x_2]$ and $[x_3, x_4]$ are good pairs of 2-valued inputs. □

4.2 Ordering of Multiple-Valued Variables

The sizes of MDDs are sensitive to orderings of input variables [10]–[14]. Several algorithms exist to find the exact variable ordering. However, such algorithms work only for functions with small number of inputs and are useless for general purposes. To find the optimum variable ordering is an NP-complete problem [9]. So, heuristics are used for the practical problems [11]–[13]. In the real life, many logic circuits have multiple outputs, and most CAD tools handle multiple-output functions at the same time. Thus, finding the same variable ordering for different output functions is important. In this part, we present a heuristic to order the inputs of multiple-output functions. We use a sampling technique to compute variable orderings of MDDs: each sample corresponds to a group of output functions, and an MDD represents a sample. Then, we incorporate an interleaving technique to generate a good variable ordering for entire MDDs from the variable orderings of the MDDs, and minimize the entire MDDs. The techniques for the proposed method are presented in Algorithms 2 and 3. From now on the variable ordering of an MDD for a sample is a ***sample variable ordering***, and the variable ordering for all the outputs obtained from the sample variable orderings is the ***final variable ordering***. To obtain the final variable ordering,

we merge a variable ordering of an MDD for a sample with higher priority into one with lower priority while maintaining the good variable ordering of each MDD as much as possible. The input variables in which a multiple-output function strongly depends on, are ***influential***. The influential variables greatly affect the size of the DD and such variables should be placed in the higher positions in the final variable ordering.

Definition 5: A ***sample*** is a multiple-output function consisting of a set of outputs. These outputs form a part of total outputs, and the number of outputs in a sample is the ***size of the sample***. A sample with the larger size of the MDD has the higher priority. □

Definition 6: ***support***(f) is the set of input variables that the function f depends on. The ***size of the support*** is the number of variables in the support(f). □

Definition 7: Let f_1 and f_2 be two output functions. ***The size of the union of the support for*** f_1 ***and*** f_2 is the number of support variables for $\{f_1, f_2\}$. □

Example 4: Consider the 2-valued 4-input 2-output function:

$$f_0(x_1, x_2, x_3, x_4) = \bar{x}_1 x_2 \vee x_1 \bar{x}_3 \vee \bar{x}_2 x_3, \quad \text{and}$$
$$f_1(x_1, x_2, x_3, x_4) = x_1 x_3 \vee \bar{x}_3 x_4.$$

The size of the union of the support for f_0 and f_1 is 4, since x_1, x_2, x_3, and x_4 are the support variables for $\{f_0, f_1\}$. □

Algorithm 2: (Derivating samples)
Let $F_2 : \{0,1\}^n \to \{0,1\}^m$.

1. Generate $F_3 : \{0,1,2,3\}^N \to \{0,1\}^m$ from F_2 by using Algorithm 1 and construct an MDD for F_3. Two output functions in F_3 are a candidate pair if the size of the union of the support for the pair is the smallest among all the pairs. Apply the same idea to the rest of the outputs recursively to find good pairs of outputs. In the case of tie, goto Step 2 to find the best one among them, else goto Step 3.
2. Let w_i, w_j, and w_{ij} be the numbers of nodes in the MDD for f_i, f_j, and $\{f_i, f_j\}$, respectively. Let $W_{ij} = w_i + w_j - w_{ij}$. Then, choose the pair of outputs with the maximum W_{ij}.
3. Find good pairs of outputs by using Steps 1 and 2, and make a partition of outputs.
4. Order the output functions as they appeared in the pairs of the partition, and make an initial sample with the ordered outputs.
5. Check the size of the sample, and do the process of generating samples by using Step 6 only if the size of the sample is larger than the expected one, otherwise stop the process for this sample.
6. Check the supports of the outputs of the sample. If all the outputs depend on all the inputs, then goto Step 7, otherwise goto Step 8.
7. Randomly divide the sample into some such that the construction of the MDDs for each sample is easy† to handle.
8. Divide the sample into two such that the outputs with common support variables are in the same sample, and the number of common support variables between samples is small†. Return to Step 5 for each sample.

Algorithm 3: (Minimization of MDDs)
Let $F_3 : \{0,1,2,3\}^N \to \{0,1\}^m$.

1. Generate samples for F_3 using Algorithm 2 and construct an MDD for each sample.
2. Optimize each MDD by using sifting starting with an initial variable ordering [11], [12], [14], and obtain the size of the MDD and the sample variable ordering.
3. Arrange the sample variable orderings in descending order of the sizes of the MDDs.
4. Compute the final variable ordering from sample variable orderings by using the following:
   (a) Let v_g be an input variable in the final variable ordering. Let v_h be an input variable of a sample variable ordering which is not in the final ordering and is more influential than v_g. Then, in the final variable ordering, insert v_h in the higher position than v_g.
   (b) Let G be a set of sample variable orderings. Choose an input variable from the top of the sample variable orderings of G and form a final ordering by maintaining the priorities of the samples in descending order and the property of Step **(a)**. Note that an input variable of a sample variable ordering is inserted into the final ordering iff the variable is not already in it.

Example 5: Let $F_3 = \{f_0, f_1, f_2, f_3\} : \{0,1,2,3\}^7 \to \{0,1\}^4$. Let $\{f_0, f_2\}$ and $\{f_1, f_3\}$ be two samples for the function F_3. Let $order[A] = (X_0, X_1, X_2, X_3)$ and $order[B] = (Y_0, Y_1, X_2, Y_3)$ be sample variable orderings obtained from the MDDs representing samples $\{f_0, f_2\}$ and $\{f_1, f_3\}$, respectively. Let 5 and 13 be the sizes of MDDs under the sample variable orderings, $order[A]$ and $order[B]$, respectively. $\{f_1, f_3\}$ has the highest priority, since the size of the MDD for this sample is the largest. So, we check $order[B]$ first and then $order[A]$ in order to generate the final variable ordering ($order[C]$). In this example, $G = \{(X_0, X_1, X_2, X_3), (Y_0, Y_1, X_2, Y_3)\}$. To compute the final variable ordering, we select the influential input variables from $order[A]$ and $order[B]$ of G according to Steps (a) and (b) as follows:

†Note that the size of a sample should not be too small so that the number of samples is large. To solve this problem, we finally combine some smaller samples together to get a considerable number of samples.

Table 1 Sizes and CPU time of DDs for benchmark functions.

Function name	In	Out	SBDD*	time† (sec.)	Initial MDD*	time‡ (sec.)	Heuristics minimization			
							sifting [11]		Proposed method	
							MDD*	time** (sec.)	MDD*	time** (sec.)
c499	41	32	27845	13.26	15431	0.32	15319	76.12	15174	18.55
c880	60	26	4139	5.03	3030	0.19	2838	40.59	2905	7.43
c1908	33	25	7430	4.18	4426	0.08	4128	17.24	4166	5.08
c2670	233	140	2706	6.20	2345	0.74	1861	25.05	1830	8.31
c3540	50	22	34680	32.07	24487	0.22	21553	122.46	21607	54.73
c5315	178	123	2440	3.16	1968	0.51	1554	36.24	1473	4.70
c7552	207	108	2836	20.36	2301	2.01	2145	156.80	1836	47.13
count	35	16	81	0.03	66	0.04	64	0.33	64	0.07
des	256	245	3729	5.14	2780	1.10	2380	57.35	2512	7.29
i3	132	6	133	0.10	68	0.01	66	0.45	66	0.15
i9	88	63	2278	0.28	1369	0.03	1122	2.18	1035	0.36
rot	135	107	8393	4.63	6245	1.12	5603	31.47	5201	6.24
s838.1	66	33	196	0.25	159	0.06	128	1.52	129	0.33
s526	24	27	123	0.08	108	0.04	84	0.82	77	0.15
s5378	199	213	2710	2.90	2160	0.39	1739	30.91	1853	4.42
s13207.1	700	790	3076	7.15	2644	1.35	2406	72.45	2265	9.34
s38584.1	1464	1730	15009	44.90	11678	4.07	11457	192.35	9533	66.01

In: number of inputs; Out: number of outputs.
†CPU time in second to minimize the SBDD using a similar method to Algorithms 2 and 3.
‡CPU time in second to encode the 2-valued inputs into 4-valued variables using Algorithm 1.
*Size of the DD without using complemented edges.
**CPU time in second to minimize the MDD.

Table 2 Effect of r-valued ($r = 4, 8, 16, 32$) inputs on MDDs.

Function name	MDD* with $r = 4$	time** (sec.)	MDD* with $r = 8$	time** (sec.)	MDD* with $r = 16$	time** (sec.)	MDD* with $r = 32$	time** (sec.)
c499	15174	18.55	8817	15.04	6138	11.56	5690	8.62
c880	2905	7.43	2513	5.23	1856	4.52	1709	2.19
c1908	4166	5.08	2785	4.16	2311	3.07	1532	1.35
c3540	21607	54.73	18632	46.88	16691	35.90	12035	27.56
c7552	1836	47.13	1701	40.31	1635	27.20	1410	16.32

*Size of the MDD without using complemented edges.
**CPU time in second to minimize the MDD.

$$\begin{array}{l} order[B] = (Y_0,\ Y_1,\ X_2,\ Y_3) \\ order[A] = (X_0,\ X_1,\ X_2,\ X_3) \\ \hline order[C] = (Y_0,\ Y_1,\ X_0,\ X_1,\ X_2,\ X_3,\ Y_3) \end{array}$$

□

5. Experimental Results

We implemented C programs to construct SBDDs and MDDs for benchmark functions. SBDDs were optimized by a similar method to Algorithms 2 and 3, while MDDs were optimized by Algorithms 1–3 and also by sifting [11], [14]. The initial MDDs were generated by using Algorithm 1. DDs in this paper don't use complemented edges. For benchmark functions with ***don't cares***, the don't cares were set to zero.

Table 1 compares the sizes of SBDDs, initial MDDs, and MDDs obtained by heuristics minimization, and presents CPU time to minimize DDs and the CPU time to encode the 2-valued inputs into 4-valued variables. This table shows: (1) MDDs require fewer nodes than SBDDs; (2) MDDs obtained by heuristics minimization require fewer nodes than initial MDDs; (3) the proposed method is often much faster than sifting: for example, sifting required 192.35 CPU seconds to minimize the MDD for s38584.1, while the proposed method required 66.01 CPU seconds to minimize the MDD for s38584.1; and (4) for many benchmark functions, the proposed method produces smaller MDDs than sifting, e.g. c7552. In addition, Table 1 shows that MDDs generated by the proposed method require on the average, 4% fewer nodes than MDDs generated by sifting, and 38% fewer nodes than SBDDs.

In this table, s38584.1 is one of the most complex benchmark functions, and our program required 44.90 CPU seconds to minimize the SBDD and 4.07 CPU seconds to encode the 2-valued inputs into 4-valued variables. Table 2 compares the sizes of MDDs for r-valued ($r = 4, 8, 16, 32$) inputs 2-valued outputs, and presents the CPU time to minimize MDDs: MDDs were minimized by the similar techniques to Algorithms 1–3. This table shows that the sizes and the CPU time of MDDs for the benchmark functions are reduced drastically when r increased from 4 to 32. All the CPU time were measured on a JU1/170 with 160 MB of main memory (Sun Ultra1-170 compatible).

6. Conclusions and Comments

In this paper, we proposed heuristics to minimize multiple-valued decision diagrams (MDDs) for multiple-output functions. We presented upper bounds on the sizes of MDDs for various functions. We also compared the sizes of MDDs with those of shared binary decision diagrams (SBDDs). Experimental results show that MDDs usually require fewer nodes than corresponding SBDDs, and sometimes MDDs require less than a half nodes of SBDDs. The proposed method is much faster, and for many benchmark functions, it produced MDDs that are smaller than ones generated by sifting. In addition, we found that the proposed method produced much smaller MDDs in a short time for benchmark functions when several 2-valued input variables are grouped to form multiple-valued variables.

Acknowledgments

This work was supported in part by a Grant in Aid for Scientific Research of the Ministry of Education, Science, Culture and Sports of Japan. This paper is based on [10].

References

[1] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Trans. Comput., vol.C-35, no.8, pp.677–691, Aug. 1986.
[2] 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.
[3] A. Srinivasan, T. Kam, S. Malik, and R.K. Brayton, "Algorithms for discrete function manipulation," Proc. International Conference on Computer-Aided Design, pp.92–95, Nov. 1990.
[4] 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.
[5] S. Tani, K. Hamaguchi, and S. Yajima, "The complexity of the optimal variable ordering problems of a shared binary decision diagram," IEICE Trans. Inf. & Syst., vol.E79-D, no.4, pp.271–281, April 1996.
[6] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of binary decision diagrams based on exchanges of variables," Proc. International Conference on Computer-Aided Design, pp.472–475, Nov. 1991.
[7] 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.
[8] G. Epstein, Multiple-Valued Logic Design: An Introduction, IOP Publishing Ltd., London, 1993.
[9] A. Thayse, M. Davio, and J-P. Deschamps, "Optimization of multivalued decision algorithms," Proc. IEEE International Symposium on Multiple-Valued Logic, pp.171–178, May 1978.
[10] H.Md. Hasan Babu and T. Sasao, "Minimization of multiple-valued decision diagrams using sampling method," Proc. Ninth Workshop on Synthesis And System Integration of MIxed Technologies (SASIMI'2000), pp.291–298, April 2000.
[11] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," Proc. International Conference on Computer-Aided Design, pp.42–47, Nov. 1993.
[12] H. Fujii, G. Ootomo, and C. Hori, "Interleaving based variable ordering methods for ordered binary decision diagrams," Proc. International Conference on Computer-Aided Design, pp.38–41, Nov. 1993.
[13] J. Jain, W. Adams, and M. Fujita, "Sampling schemes for computing OBDD variable orderings," Proc. International Conference on Computer-Aided Design, pp.631–638, Nov. 1998.
[14] F. Somenzi, Colorado university decision diagram package (CUDD), release 2.1.2, 1997.

Hafiz Md. Hasan Babu was born in Bangladesh. He received the M.Sc. degree in Computer Science and Engineering from the Brno University of Technology, Czech Republic, in 1992 and the Ph.D. degree in Computer Science and Electronics from the Kyushu Institute of Technology, Japan in March, 2000. In 1995, he was at the Asian Institute of Technology (AIT), Thailand under the DAAD Fellowship from the Federal Republic of Germany. Now, he is working as an Assistant Professor in the Department of Computer Science and Engineering, Khulna University, Bangladesh. His research interests include logic synthesis, representations of logic functions, and multiple-valued logic. He is student members of the IEEE and the IEEE Computer Society.

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.