Multi-Terminal Multi-Valued Decision Diagrams for Characteristic Function Representing Cluster Decomposition

Hiroki Nakahara*, Tsutomu Sasao†, and Munehiro Matsuura†
*Kagoshima University, Japan, †Kyushu Institute of Technology, Japan

***Abstract*—Binary decision diagrams representing complex logic circuits require a large number of nodes. This paper shows a new method to represent logic circuits using multiple decision diagrams. First, a given logic circuit is converted into a direct acyclic graph (DAG). Then, the DAG is decomposed into clusters. Next, clusters are represented by multi-terminal binary decision diagrams for characteristic function (decomposed MTBDDs for CF). It represents a logic circuit more compactly than the conventional MTBDD for CF. Finally, the decomposed MTBDDs for CF are converted into the multi-terminal multi-valued decision diagrams for CF (decomposed MTMDDs for CF) to be stored in the given memory size. Also, the decomposed MTMDDs for CF is faster to evaluate than the conventional MTMDD for CF using the same memory size on a BDD machine.**

I. Introduction

Binary decision diagrams (BDDs) represent logic functions efficiently, and are used for many applications including formal verification [1]; logic simulation [2], [10], [20]; logic design [6]; and special purpose processors [13].

As for a complex circuit, such as a multiplier, since the number of the nodes in a BDD increases exponentially with the number of inputs, a BDD is unsuitable to represent such circuit, when the number of inputs is large [18]. Bryant et al. have proposed the multiplicative binary moment diagram (*BMD) to represent the multiplier [3]. However, as for the divider, the number of nodes in the *BMD increases exponentially [16]. Thus, for practical arithmetic circuits, sizes for BDDs and *BMDs increase exponentially with the number of inputs. To represent a logic circuit by diagrams with a reasonable size, Bryant et al. decomposed the multiplier into sub-circuits, and represented them by multiple *BMDs [3]. We also decomposed a circuit and represented sub-circuits by decision diagrams [9], [15]. These facts show that the size of the decision diagrams for decomposed circuit is smaller than the monolithic decision diagram for the original circuit.

A heterogeneous multi-valued decision diagram (HMDD) may have nodes with different number of variables [12]. Since the HMDD can use the optimal partition of the input variables to reduce path length, the evaluation time of the HMDD is shorter than the BDD [14]. We will show that the HMDDs representing decomposed circuit can be evaluated faster than the conventional HMDD.

Contributions of the paper are as follows:

To represent large functions, we adopted the HMDDs representing decomposed circuit with a reasonable size. To simulate large functions, BDDs representing decomposed circuit are used [2], [10], [20] on computers. In this application, the reduction of the cache miss leads to the high-speed simulation. Thus, small size BDDs are suitable to fit the cache with the limited size of memory. On the other hand, we enhance the performance by using the multi-terminal multi-valued decision diagrams (MTMDDs) with an enough size of memory. It is an opposite of their works. In the decision diagram machine [13] with no cache, the evaluation time is proportional to the path length of the decision diagram. An enough size of memory reduces the path length of HMDDs [14].

We showed that the HMDDs representing decomposed circuit is faster than the monolithic HMDD with an enough size of memory. Since the HMDDs representing decomposed circuit represents the logic circuit, the evaluation time is $O(g)$, where g is the number of logic gates. On the other hand, since the monolithic HMDD represents the logic function, the evaluation time is $O(n)$, where n is the number of inputs. Generally, since $g \gg n$, the monolithic HMDD seems to be faster than the HMDDs representing decomposed circuit. However, we discover that, with enough size of memory (in our experiment, it is one Mega bytes), the HMDDs representing decomposed circuit is faster than the monolithic HMDD. Decades ago, when the memory was expensive, it was difficult to use a memory with large size. Nowadays, the large-scale integration gives us more than one Giga bytes memory at a low price. Therefore, adopting the HMDDs representing decomposed circuit with a large size of memory is practical.

The rest of the paper is organized as follows: Chapter 2 defines decision diagrams; Chapter 3 defines the circuit decomposition; Chapter 4 introduces the heterogeneous multi-valued decision diagrams representing a decomposed circuit; Chapter 5 shows the experimental results; and Chapter 6 concludes the paper.

II. Definition of Decision Diagrams

A. Decision Diagram (DD)

Definition 2.1: A **binary decision diagram (BDD)** is obtained by applying **Shannon expansions** repeatedly to a logic function f [4]. Each non-terminal node labeled with a variable x_i has two outgoing edges which indicate nodes representing cofactors of f with respect to x_i. When the

0195-623X/12 $26.00 © 2012 IEEE
DOI 10.1109/ISMVL.2012.45

IEEE computer society

Shannon expansions are performed with respect to k variables, all the non-terminal nodes have 2^k edges. In this case, we have a **multi-valued decision diagram (MDD(k))** [8].

Definition 2.2: In a DD, a sequence of edges and non-terminal nodes leading from the root node to a terminal node is a **path**. **An ordered BDD (OBDD)** has the same variable order on any path. **A reduced ordered BDD (ROBDD)** is derived by applying the following two reduction rules to an OBDD:

1. Share equivalent sub-graphs.
2. If all the outgoing edges of a non-terminal node v point the same succeeding node u, then delete v and connect the incoming edges of v to u.

An **ROMDD(k)** can be defined similarly to the ROBDD. Note that, an MDD(1) means a BDD. In this paper, a BDD and an MDD(k) mean an ROBDD and an ROMDD(k), respectively, unless stated otherwise.

Definition 2.3: Let $f(X) : B^n \rightarrow B$ be a two-valued logic function, where $B = \{0, 1\}$. Let $X = (x_1, x_2, \ldots, x_n)$, $x_i \in B$ be an ordered set of binary variables. Let $\{X\}$ denote the unordered set of variables in X. If $\{X\} = \{X_1\} \cup \{X_2\} \cup \cdots \cup \{X_u\}$ and $\{X_i\} \cap \{X_j\} = \emptyset (i \neq j)$, then $(X_1, X_2, \ldots, X_u)$ is a **partition of** X, where X_i denotes **a super variable**. When $k_i = |X_i| (i = 1, 2, \ldots, u)$, we have the relation $k_1 + k_2 + \cdots + k_u = n$.

Definition 2.4: Let $X = (X_1, X_2, \ldots, X_u)$ be a partition of the input variables, and $k_i = |X_i|$ be the number of inputs for node i. When $k = |X_1| = |X_2| = \cdots = |X_u|$, an ROMDD is **a homogeneous MDD (MDD(k))**. On the other hand, if there exists a pair (i, j) such that $|X_i| \neq |X_j|$, then, it is **a heterogeneous MDD (HMDD)**.

An HMDD is a generalization of an MDD(k). If the evaluation time for all the nodes are the same, then the evaluation time for an HMDD is proportional to the **average path length (APL)** [5].

Definition 2.5: Let $(X_1, X_2, \ldots, X_u)$ be a partition of the input variables X. Suppose that X_i can take any value c in $\{0, 1, \ldots, 2^{k_i} - 1\}$. Then, $P(X_i = c)$ denotes the probability that X_i has the value c. **The Path Probability (PP)** of a path p_i, denoted by $PP(p_i)$, is the probability that the path p_i is selected in all assignments of values to the 2^{k_i}-valued variables. Then, we have $PP(p_i) = \sum_{\vec{c} \in C_i} P(X_1 = c_1) \cdot P(X_2 = c_2) \cdot \ldots \cdot P(X_u = c_u)$, where C_i denotes a set of assignments of values to the variables X selecting the path p_i, and $\vec{c} = (c_1, c_2, \ldots, c_u)$. **The average path length (APL)** of a DD is $APL = \sum_{i=1}^{N} PP(p_i) \cdot l_i$, where N denotes the number of paths, and l_i denotes the path length of path p_i.

B. Representation of Multi-output Logic Function Using Decision Diagrams for Characteristic Function

Many practical applications use multiple-output functions. Here, we represent an n-input m-output logic function using a decision diagram (DD).

Definition 2.6: Let $\vec{X} = (x_1, x_2, \ldots, x_n)$ be the input variables, $\vec{Y} = (y_1, y_2, \ldots, y_m)$ be the output variables, and $\vec{f} = (f_1(\vec{X}), f_2(\vec{X}), \ldots, f_m(\vec{X}))$ be a multiple-output function. **The characteristic function (CF) of a multiple-output function** is $\vec{\chi}(\vec{X}, \vec{Y}) = \bigwedge_{i=1}^{m} (y_i \equiv f_i(\vec{X}))$.

The characteristic function of an n-input m-output function is a two-valued logic function with $(n + m)$ inputs. It has input variables x_i $(i = 1, 2, \ldots, n)$, and output variables y_j for outputs f_j. Let $B = \{0, 1\}$, $\vec{a} \in B^n$, $\vec{F} = (f_1(\vec{a}), f_2(\vec{a}), \ldots, f_m(\vec{a})) \in B^m$, and $\vec{b} \in B^m$. Then, the characteristic function satisfies the relation

$$\vec{\chi}(\vec{a}, \vec{b}) = \begin{cases} 1 & (when\ \vec{b} = \vec{F}(\vec{a})) \\ 0 & (otherwise) \end{cases}$$

Definition 2.7: **A support variable** of a function f is a variable on which f actually depends.

Definition 2.8: **A multi-terminal binary decision diagram for characteristic function (MTBDD for CF)** of a multiple-output function $\vec{f} = (f_1, f_2, \ldots, f_m)$ represents the characteristic function $\vec{\chi}$. We assume that the root node is in the top of the MTBDD, and the variable y_i is below the support variable of f_i, where y_i is the variable representing f_i.

Definition 2.9: [19] **The width of the MTBDD for CF** at the height k is the number of edges crossing the section of the graph between x_k and x_{k+1}, where the edges incident to the same nodes are counted as one.

III. Cluster Decomposition of Circuit

A. Graph Representation of Combinational Circuit

In this paper, a combinational circuit is represented by **a directed acyclic graph (DAG)**. In the DAG, **a primary input node** denotes a primary input; **a primary output node** denotes a primary output; and **an intermediate node** denotes a 2-input logic gate[1]. When the output of a logic gate i is connected to a logic gate j, the DAG has an edge from a node i to a node j. In the DAG, we assume that a primary input does not fan-out. Thus, we need to duplicate the input variables. Therefore, the number of the primary input nodes can be greater than the number of the primary inputs. We denote the input nodes by *(primary input name)_(unique number)*.

Example 3.1: Fig. 1 shows the circuit for a two-bit multiplier, where $\{x_0, x_1, x_2, x_3\}$ denotes the primary inputs and $\{y_0, y_1, y_2, y_3\}$ denotes the primary outputs. Fig. 2 shows the DAG for the multiplier in Fig. 1, where $X = \{x_{0\_0}, x_{0\_1}, x_{1\_0}, x_{1\_1}, x_{2\_0}, x_{2\_1}, x_{3\_0}, x_{3\_1}\}$ denotes the primary input nodes; $Y = \{y_0, y_1, y_2, y_3\}$ denotes the primary output nodes; and $W = \{w_0, w_1, w_2, w_3, w_4, w_5, w_6, w_7\}$ denotes the intermediate nodes. ■

Next, we define a decomposition of the DAG.

Definition 3.10: **A cut** (S, T) is a partition of nodes in the DAG. For $s \in S$ and $t \in T$, no node of T has an edge directed to any node of S.

Definition 3.11: Let X be a set of primary input nodes of the DAG. A set of nodes $D(X)$ depending X is **a depended**

[1]A NOT gate is converted into a NAND gate having the same inputs. Also, a gate with more than two inputs is decomposed into multiple gates with two inputs.

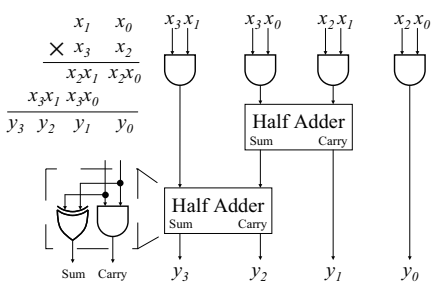

Fig. 1. Circuit for the 2-bit multiplier.

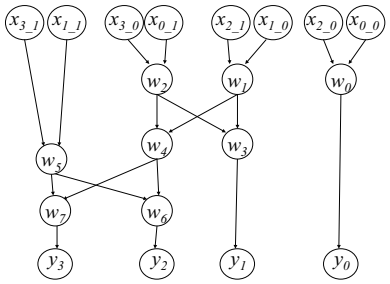

Fig. 2. DAG representing the multiplier shown in Fig. 1.

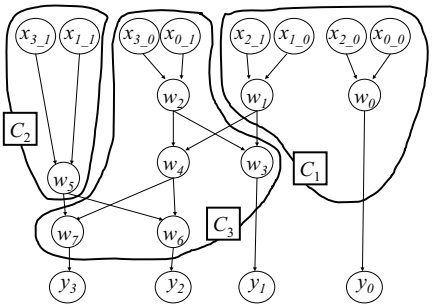

Fig. 3. An example of a cluster decomposition.

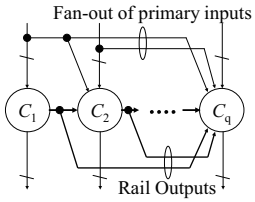

Fig. 4. Circuit realizing the cluster decomposition.

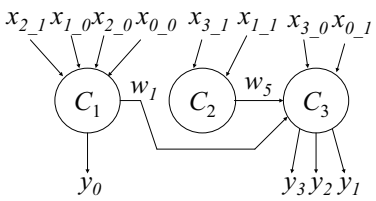

Fig. 5. Circuit realizing the cluster decomposition of 2-bit multiplier.

node set. Note that, $D(X)$ includes the primary input nodes X, while it does not include the primary output nodes Y.

Example 3.2: In Fig. 2, let $X = \{x_{1\_0}, x_{2\_1}, x_{0\_1}, x_{3\_0}\}$ be a subset of primary input nodes of the DAG. In this case, $D(X) = \{x_{1\_0}, x_{2\_1}, x_{0\_1}, x_{3\_0}, w_1, w_2, w_3, w_4\}$. ■

Definition 3.12: Let V be a set of all the nodes in the DAG; $(X_1, X_2, X_3, ..., X_q)$ be a partition of the primary input nodes X, where $X_i \cap X_j = \emptyset$ $(i \neq j)$; and Y be the set of the primary output nodes. **A cut set with topological order** $\{(S_i, T_i) | i = 1, 2, \ldots, q\}$ is the set satisfying:

1. For first cut (S_1, T_1), $S_1 = D(X_1)$ and $T_1 = V - S_1$.
2. For i-th cut (S_i, T_i), $S_i = D(X_1 \cup X_2 \cup \ldots \cup X_i)$ and $T_i = V - S_i$, where $1 < i < q$.
3. For q-th cut (S_q, T_q), $S_q = D(X)$ and $T_q = Y$.

When the partition of the primary input nodes is given, the cut set with topological order is uniquely determined. By evaluating a set S_i $(i = 1, 2, \ldots, q)$ in order, we can evaluate the DAG.

Definition 3.13: Suppose that the DAG is decomposed into a cut set with topological order $\{(S_i, T_i) | i = 1, 2, \ldots, q\}$. $C_i = S_i - S_{i-1}$ is **a cluster**, where $S_0 = \emptyset$. A decomposition of the DAG into a set of clusters $\{C_1, C_2, \ldots C_q\}$ is **a cluster decomposition**.

Example 3.3: Fig. 3 shows an example of a cluster decomposition, where $C_1 = \{x_{0\_0}, x_{2\_0}, x_{1\_0}, x_{2\_1}, w_0, w_1\}$, $C_2 = \{x_{1\_1}, x_{3\_1}, w_5\}$, and $C_3 = \{x_{0\_1}, x_{3\_0}, w_2, w_4, w_6, w_7, w_3\}$. ■

In a set of clusters $\{C_1, C_2, \ldots, C_q\}$, when $i < j$, the cluster C_i may have edges directed to the cluster C_j, while the cluster C_j does not have edges directed to the cluster C_i.

Definition 3.14: Fig. 4 shows a circuit representing a set of clusters $\{C_1, C_2, \ldots C_q\}$. Let $i < j$. The outputs of the cluster C_i directed to the cluster C_j are **rail outputs**, and the inputs of the cluster C_j directed from the cluster C_i are **rail inputs**.

The inputs for a cluster consist of primary inputs and rail inputs, while the outputs for a cluster consist of primary outputs and rail outputs.

Example 3.4: Fig. 5 shows the circuit realizing the cluster decomposition of the two-bit multiplier shown in Fig. 3. ■

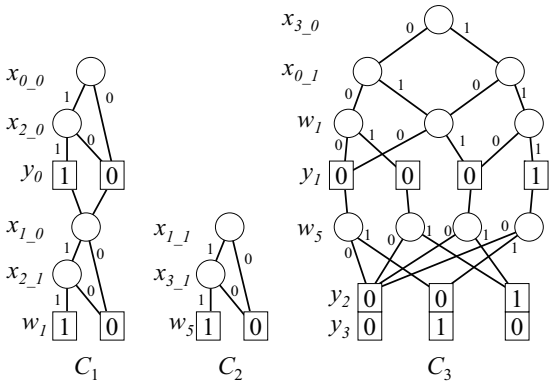

Fig. 6. MTBDDs for CF representing clusters shown in Fig. 5.

IV. Decomposed Multi-Terminal Multi-valued Decision Diagrams for Characteristic Function

A. MTBDDs for CF Representing Cluster Decomposition

In a set of clusters $\{C_1, C_2, \ldots, C_q\}$, C_i is represented by an MTBDD for CF, where the inputs consist of primary inputs and rail inputs, and the outputs consist of primary outputs and rail outputs directing to other MTBDDs for CF representing C_j $(j > i)$.

Example 4.5: Fig. 6 shows MTBDDs for CF representing clusters shown in Fig. 5. ■

Definition 4.15: Let $\{C_1, C_2, \ldots, C_q\}$ be a set of clusters, where C_i is represented by an MTBDD for CF. The **MTBDDs for CF representing a cluster decomposition (decomposed MTBDDs for CF)** connect MTBDDs for CF in the topological order of $\{C_1, C_2, \ldots, C_q\}$.

Example 4.6: Fig. 7 shows the decomposed MTBDDs for CF representing the two-bit multiplier in Fig. 3. ■

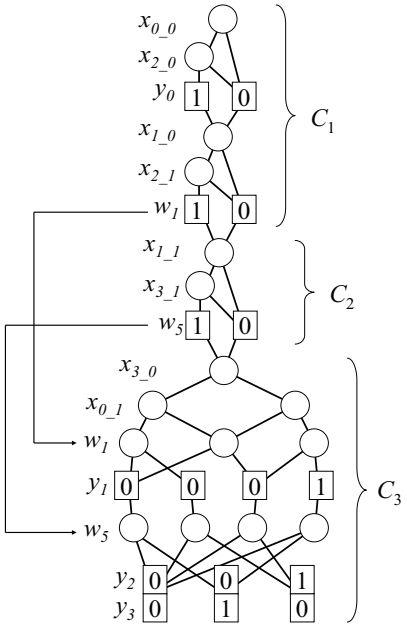

Fig. 7. An example of the decomposed MTBDDs for CF.

TABLE I
TAXONOMY OF DECISION DIAGRAMS (DDS).

Decision Diagram Type	Fan-out of Primary Inputs	Fan-out of Rail Outputs
Monolithic MTBDD for CF [19]	Not allowed	Not allowed
Indexed BDD [7]	Allowed	Not allowed
Decomposed MTBDDs for CF	Allowed	Allowed

Definition 4.16: A **monolithic MTBDD for CF** represents a set of clusters. It is a conventional MTBDD for CF.

Table I shows the taxonomy of decision diagrams representing a cluster decomposition in terms of the fan-outs of the primary inputs and rail outputs. Table I shows that the decomposed MTBDDs for CF are unique decision diagrams that allow the fan-out of rail outputs.

B. Decomposed MTMDDs for CF

Decomposed MTBDDs for CF are extended to **decomposed multi-valued decision diagrams for CF (decomposed MTMDDs for CF)**. The decomposed MTMDDs for CF can use more memory than the decomposed MTBDDs for CF, to reduce their evaluation time.

Example 4.7: Fig. 8 shows the decomposed MTMDDs for CF converted from the MTBDDs for CF shown in Fig. 7. Note that, the decomposed MTMDDs for CF is obtained by merging the cluster C_1 and a part of cluster C_2, and by changing the variable order of a node. ■

As shown in Example 4.7, in MTBDDs for CF, multiple clusters are merged and the variable order of a node is changed to obtain MTMDDs for CF.

Since the decomposed MTBDDs for CF allow fan-outs of both the primary inputs and the rail outputs, they can represent a complex circuit compactly. Thus, we have the following:

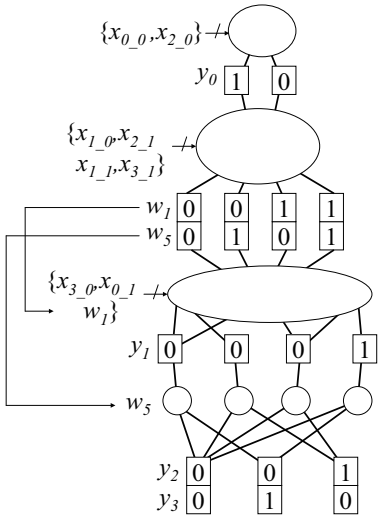

Fig. 8. An example of the decomposed MTMDDs for CF.

Conjecture 4.1: Suppose that the given circuit has fan-outs of primary inputs and rail outputs. When the circuit is converted into the BDD, the number of nodes may increase exponentially.

Experimental results in Section V-A confirm Conjecture 4.1. Note that, the decomposed MTBDDs for CF lost canonicity of the BDD. Thus, the application of decomposed MTBDDs for CF to formal verification is difficult. However, applications of the decomposed MTBDDs for CF to logic simulation is straightforward, since the canonicity is not used.

V. EXPERIMENTAL RESULTS

A. Comparison of BDDs Representing Cluster Decompositions

To confirm Conjecture 4.1, we used multipliers [2] and **hidden weight bit (HWB) functions** whose BDDs increase exponentially with n. The HWB function selects the input corresponding to the weight of the inputs. Fig. 12 shows the circuit for the seven-bit HWB function. We compare the numbers of nodes for three decision diagrams:

1. The monolithic MTBDD for CF: Does not allow fan-outs of primary inputs nor rail outputs.
2. The indexed BDD: Allows the fan-out of primary inputs, while does not allow fan-out of rail outputs.
3. The decomposed MTBDDs for CF: Allow the fan-outs of both primary inputs and rail outputs.

To obtain the decomposed MTBDDs for CF, first, we generated the netlist from the Verilog-HDL description by using Quartus II logic synthesis tool ver.9.1 with area minimization option. Then, we decomposed the netlist into clusters by using the greedy algorithm [15]. Next, we converted clusters into the decomposed MTBDDs for CF. To reduce the number of nodes in MTBDDs, we used the sifting algorithm [17]. As

[2]The *BMDs can represent multipliers compactly.

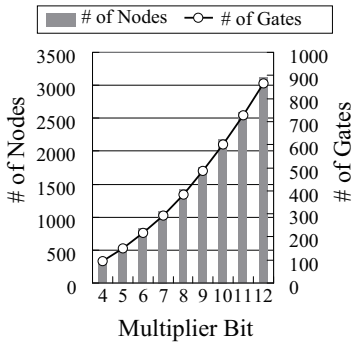

Fig. 9. Number of nodes and gates for n-bit multiplier.

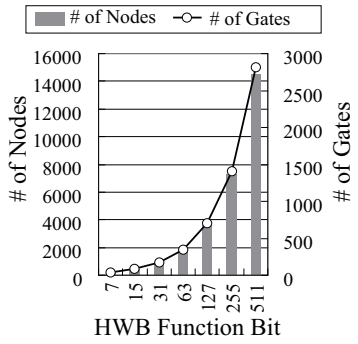

Fig. 10. Number of nodes and gates for n-bit HWB function.

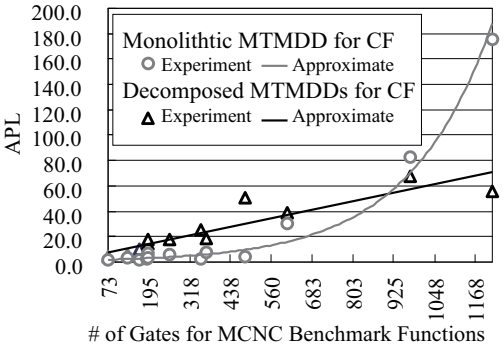

Fig. 11. APL for two types of MDDs.

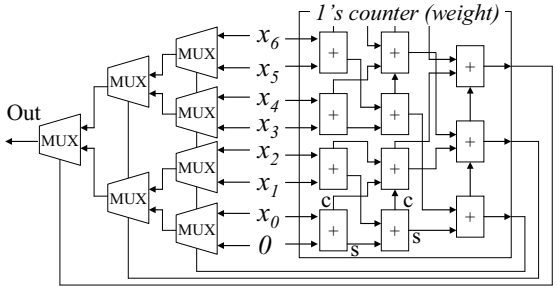

Fig. 12. Circuit for the seven-bit HWB function.

TABLE II
NUMBER OF NODES FOR n-BIT MULTIPLIER.

Multiplier Width Bits	Monolithic MTBDD for CF	Indexed BDD	Decomposed MTBDDs for CF
4×4	507	372	**323**
5×5	2014	784	**527**
6×6	7889	1452	**839**
7×7	31006	2468	**1097**
8×8	122438	3892	**1433**

for the indexed BDD and the monolithic MTBDD for CF for the HWB function, we borrowed the numbers of nodes in [7]. Table II compares the numbers of nodes for n-bit multipliers, and Table III compares that of n-bit HWB functions. Tables II and III show that the decomposed MTBDDs for CF can represent large and complex function compactly.

TABLE III
NUMBER OF NODES FOR n-BIT HWB FUNCTION.

HWB Function Bits	Monolithic MTBDD for CF	Indexed BDD	Decomposed MTBDDs for CF
15	282	**137**	444
31	6326	**529**	921
63	2.5×10^9 (Estimation)	2081	**1813**
127	239×10^9 (Estimation)	8257	**3645**

B. Complexities of Decomposed MTBDDs for CF

Let g_{mul} be the estimated number of gates to implement the n-bit multiplier consisting of an $n \times n$ array of adders. Then, we have $g_{mul} = n^2 \times g_{adr}$, where g_{adr} denotes the number of gates in a full adder. Let g_{HWB} be the estimated number of gates for the circuit of the n-bit HWB function, where $n = 2^q - 1$ $(q = 2, 3, 4, \ldots)$. As shown in Fig. 12, the circuit for the HWB function consists of an n-bit selector and an n-bit 1's counter. Thus, we have $g_{HWB} = n \times g_{mux} + \sum_{p=1}^{\lceil log_2(n+1) \rceil} 2^{\lceil log_2(n+1) \rceil - p} g_{adr}$, where g_{mux} denotes the number of gates to implement a two-input multiplexor.

Fig. 9 shows the relationship between the number of nodes of the decomposed MTBDDs for CF and the number of gates for the n-bit multiplier, while Fig. 9 shows the relationship for the n-bit HWB function. As shown in Figs. 9 and 10, the number of nodes in the decomposed MTBDDs for CF is proportional to the number of gates g. Therefore, the complexity of the decomposed MTBDDs for CF is $O(g)$.

C. Comparison of the Evaluation Times for MDDs

The monolithic MTBDD for CF does not allow fan-outs of the primary inputs nor the rail outputs, while the decomposed MTBDDs for CF allow these fan-outs. Thus, the APL for the monolithic MTBDD for CF is shorter than that for the decomposed MTBDDs for CF. As for the HMDD, the APL can be reduced by increasing the memory size. So, in general, evaluation using the HMDD can be faster than the BDD. We compared the numbers of nodes. To obtain the decomposed MTMDDs for CF, first, we set the memory size limitation to 1 Mega bytes (MB). Then, we converted the decomposed MTBDDs for CF into an MTMDDs for CF by using the dynamic programming [11]. Table IV compares the APL for the decomposed MTMDDs for CF with that for the monolithic MTMDD for CF representing MCNC benchmark functions [21]. Note that, the monolithic MTBDD for CF could not represent s38417 nor s38584. Fig. 11 shows the APL for each MDD.

As for small functions (s420, s510, s641, s1196, s5378), the APL for the monolithic MTMDD for CF is shorter than

TABLE IV
COMPARISON OF APLS FOR MCNC BENCHMARK FUNCTIONS (MEMORY SIZE LIMITATION 1 [MB]).

Name	In	Out	FF	# of gates	Monolithic MTMDD for CF	Decomposed MTMDDs for CF
Small functions						
s420	18	1	16	187	**2.3**	8.5
s510	19	7	22	256	**6.2**	18.0
s641	36	23	19	193	**3.8**	17.5
s1196	13	13	19	483	**4.1**	51.3
s5378	35	49	164	1294	**112.8**	129.1
Large functions						
s9234	36	39	211	974	82.3	**67.7**
s13207	36	39	211	1219	175.2	**55.6**
s38417	28	106	1636	8278	—	**112.7**
s38584	38	204	1424	6724	—	**1016.3**

that for the decomposed MTMDDs for CF. Since both BDDs can represent the small function compactly, they can reduce the APL by using extra memory space with the MDD. As for large functions (s9234, s13207), the APL for the decomposed MTMDDs for CF is shorter than that for the monolithic MTMDD for CF. Therefore, we can use an appropriate MDD as follows:

- For a small function, use a monolithic MTMDD for CF
- For a large function, use decomposed MTMDDs for CF

VI. CONCLUSION AND COMMENTS

This paper showed the decomposed MTMDDs for CF. To obtain the decomposed MTMDDs for CF, first, the given circuit is decomposed into clusters. Then, each cluster is converted into the MTBDD for CF. Next, the decomposed MTBDDs for CF is obtained by connecting them by the topological order. Finally, the decomposed MTBDDs for CF is converted into MTMDDs for CF to be stored in the given memory space. Its complexity is $O(g)$, where g is the number of gates for the given circuit. The decomposed MTBDDs for CF is smaller than the monolithic MTBDD for CF. Also, under the same memory size, the decomposed MTMDDs for CF is faster to evaluate than the monolithic MTMDD for CF for large functions on a BDD machine.

The decomposed MTMDDs for CF can represent the large functions that cannot be represented by a monolithic decision diagram. Thus, the decomposed MTMDDs for CF are promising for new applications. The future project is to utilize the decomposed MTMDDs for CF in practical applications.

VII. ACKNOWLEDGMENTS

This research is supported in part by the Grants in Aid for Scientific Research of JSPS, and the grant of Innovative Cluster Project of MEXT (the second stage).

REFERENCES

[1] A. Aziz, F. Balarin, S. T. Cheng, R. Hojati, T. Kam, S. C. Krishnan, R. K. Ranjan, T. R. Shiple, V. Singhal, S. Tasiran, H. Y. Wang, R. K. Brayton, and A.L. Sangiovanni-Vincentelli, "HSIS: A BDD-based environment for formal verification," *31st Conference on Design Automation (DAC1994)*, June, 1994, pp. 454-459.

[2] P. Ashar and S. Malik, "Fast functional simulation using branching programs," *IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD1995)*, Nov., 1995, pp. 408-412.

[3] R. E. Bryant and Y. A. Chen, "Verification of arithmetic circuits with binary moment diagrams," *32nd Conference on Design Automation (DAC1995)*, 1995, pp. 535-541.

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

[5] J. T. Butler, T. Sasao, and M. Matsuura, "Average path length of binary decision diagrams," *IEEE Trans. on Compt.*, Vol. 54, No. 9, Sep. 2005, pp. 1041-1053.

[6] S. C. Chang, M. Marek-Sadowdka, and T. Hwang, "Technology mapping for LUT FPGAs based on decomposition of binary decision diagrams," *IEEE Trans. on CAD*, Vol. 15, No. 10, Oct., 1996, pp. 1226-1236.

[7] J. Jain, J. Bitner, M. S. Abadir, J. A. Abraham, and D. S. Fussell, "Indexed BDDs: Algorithmic advances in techniques to represent and verify boolean functions," *IEEE Trans. on Compt.*, Vol. 46, No. 11, Nov. 1997, pp. 1230-1245.

[8] T. Kam, T. Villa, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Multi-valued decision diagrams: Theory and applications," *Multiple-Valued Logic*, Vol.4, no.1-2, 1998, pp.9-62.

[9] M. Matsuura, T. Sasao, J. T. Butler, and Y. Iguchi, "Bi-partition of shared binary decision diagrams," *IEICE Trans. on Fundamentals of Electronics*, Vol.E85-A, No.12, Dec. 2002, pp.2693-2700.

[10] P. C. McGeer, K. L. McMillan, A. Saldanha, A. L. Sangiovanni-Vincentelli, and P. Scaglia, "Fast Discrete Function Evaluation using Decision Diagrams," *ICCAD1995*, Nov.,1995, pp. 402-407.

[11] S. Nagayama and T. Sasao, "On the optimization of heterogeneous MDDs," *IEEE Trans. on CAD*, Vol.24, No.11, Nov., 2005, pp.1645-1659.

[12] S. Nagayama and T. Sasao, "Compact representations of logic functions using heterogeneous MDDs," *33rd IEEE Int'l Symp. on Multiple-Valued Logic (ISMVL2003)*, May, 2003, pp.247-255.

[13] H. Nakahara, T. Sasao, and M. Matsuura, "A comparison of multi-valued and heterogeneous decision diagram machines," *Journal of Multiple-Valued Logic*, (To be published).

[14] H. Nakahara, T. Sasao and M. Matsuura, "A comparison of architectures for various decision diagram machines," *40th Int'l Symp. on Multiple-Valued Logic (ISMVL2010)*, Barcelona, Spain, May, 26-28, 2010, pp.229-234.

[15] H. Nakahara, T. Sasao, and M. Matsuura, "A PC-based logic simulator using a look-up table cascade emulator," *IEICE Trans. on Fundamentals of Electronics*, Vol. E89-A, No. 12, Dec., 2006, pp. 3471-3481.

[16] M. Nakanishi, K. Hamaguchi, and T. Kashiwabara "An exponential lower bound on the size of a binary moment diagram representing integer division," *IEICE Trans. on Fundamentals of Electronics*, Vol. E82-A, No. 5, May, 1999, pp. 756-766.

[17] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," *IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD1993)*, Nov., 1993, pp. 42-47.

[18] T. Sasao, *Memory-Based Logic Synthesis, Springer.*, 2011.

[19] T. Sasao and M. Matsuura, "A method to decompose multiple-output logic functions," *41st Design Automation Conference (DAC2004)*, June, 2004, pp. 428-433.

[20] C. Scholl, R. Drechsler, and B. Becker, "Functional simulation using binary decision diagrams," *IEEE/ACM Int'l Conf. on Computer-Aided Design (ICCAD1997)*, Nov., 1997, pp. 8-12.

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