On the Optimization of Heterogeneous MDDs

Shinobu Nagayama, *Member, IEEE*, and Tsutomu Sasao, *Fellow, IEEE*

***Abstract*—This paper proposes minimization algorithms for the memory size and the average path length (APL) of heterogeneous multivalued decision diagrams (MDDs). In a heterogeneous MDD, each multivalued variable can take different domains. To represent a binary logic function using a heterogeneous MDD, we partition the binary variables into groups with different numbers of binary variables and treat the groups as multivalued variables. Since memory size and APL of a heterogeneous MDD depend on the partition of binary variables as well as the ordering of binary variables, the memory size and the APL of a heterogeneous MDD can be minimized by considering both orderings and partitions of binary variables. The experimental results show that heterogeneous MDDs can represent logic functions with smaller memory sizes than free binary decision diagrams (FBDDs) and smaller APLs than reduced ordered BDDs (ROBDDs); the APLs of heterogeneous MDDs can be reduced by half of the ROBDDs without increasing memory size; and heterogeneous MDDs have smaller area–time complexities than $\mathrm{MDD}(k)$s.**

***Index Terms*—Area–time complexity, $\mathrm{MDD}(k)$, average path length (APL), FBDD, heterogeneous MDD, logic simulation, memory size, representation of logic functions, ROBDD.**

I. Introduction

BINARY decision diagrams (BDDs) [5] and multivalued decision diagrams (MDDs) [16] are extensively used in logic synthesis [9], logic simulation [1], [13], [19], software synthesis [2], [15], [26], etc. To reduce memory sizes and runtime for these applications, proper optimizations of decision diagrams (DDs) are very important. For example, in logic simulation using DDs [1], [13], [19], minimization of the average path length (APL) of DDs reduces the evaluation time of logic functions, since the evaluation time depends on the path length of DDs. In software synthesis using DDs [2], [15], [26], which automatically generates program codes from a functional specification, minimization of both memory size and APL of DDs makes program codes compact and faster, since the code size and the runtime of the generated program codes depend on the memory size and path length of DDs. Most optimization algorithms for DDs use variable reordering approaches [7]–[11], [14], [20], [21], [23], [28], [32], [38]. However, when MDDs are used to represent binary logic functions, we can use an additional optimization approach, which is a partition of binary variables [12], [29], [33], [36].

Manuscript received July 13, 2004; revised November 15, 2004. This work was supported in part by the Grant in Aid for Scientific Research of The Japan Society for the Promotion of Science (JSPS) and the funds from the Ministry of Education, Culture, Sports, Science and Technology (MEXT) via the Kitakyushu Innovative Cluster Project. This paper was recommended by Associate Editor I. Stok.

The authors are with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka 820-8502, Japan (e-mail: nagayama@ieee.org).

Digital Object Identifier 10.1109/TCAD.2005.852290

To represent a binary logic function using an MDD, we partition the binary variables into groups and treat each group as a multivalued variable. In many cases, the groups have the same number of binary variables [12], [19], [33], [36]. On the other hand, in a heterogeneous MDD [29], the groups can have different numbers of binary variables. Thus, by optimizing both orderings and partitions of binary variables, we can obtain heterogeneous MDDs that have smaller memory sizes and APLs than ordinary MDDs with the same group size (called homogeneous MDDs). In [36], the optimizations of both orderings and partitions of binary variables for only homogeneous MDDs are considered.

In this paper, we propose memory size and APL minimization algorithms for heterogeneous MDDs that consider both orderings and partitions of binary variables. For BDDs and homogeneous MDDs, the APL minimization often increases memory sizes. On the other hand, for heterogeneous MDDs, the APLs can be minimized without increasing memory size. The memory size minimization algorithm for heterogeneous MDDs can reduce both memory sizes and APLs, and the APL minimization algorithm for heterogeneous MDDs can reduce APLs without increasing memory size. By experiments, we show that both memory sizes and APLs of heterogeneous MDDs can be reduced to 86% and 67% of reduced ordered BDDs (ROBDDs), respectively, and APLs of heterogeneous MDDs can be reduced by half of the ROBDDs without increasing memory size.

The rest of the paper is organized as follows. Section II shows the necessary terminology and theorems. Section III proposes memory size and APL minimization algorithms for heterogeneous MDDs. And Section IV compares memory sizes and APLs of heterogeneous MDDs for many benchmark functions.

This paper is an extended version of [30] and [31].

II. Preliminaries

This section defines the necessary terminology and shows theorems. In this paper, we assume that 1) the given logic function is completely specified and has no redundant variables; and 2) the reader is familiar with the standard terminology for BDDs and ROBDDs [5].

A. Free BDD

Definition 2.1: An ROBDD has the same variable order on all paths. A free BDD (FBDD) is a BDD that allows different variable orders along different paths in the BDD.

An FBDD is a generalization of an ROBDD, so FBDDs can be more compact than ROBDDs by considering different variable orders along each path [10], [11], [38].

B. MDD

This section provides a brief definition of MDD. Please refer to [16] for more details.

Definition 2.2: A BDD represents a binary logic function $f_b(x_1, x_2, \ldots, x_n) : B^n \rightarrow B$, where $B = \{0, 1\}$. An MDD represents a multivalued (p-valued) function $f_m(X_1, X_2, \ldots, X_u) : P^u \rightarrow P$, where $P = \{0, 1, \ldots, p-1\}$ and $p \geq 3$.

Nonterminal nodes in an MDD for the p-valued function have p outgoing edges while nonterminal nodes in a BDD have two outgoing edges. As with a BDD, there are two types of MDD: reduced ordered MDD (ROMDD) and free MDD (FMDD). This paper considers only ROMDDs.

C. Partitions of Binary Variables

To represent a binary logic function using an ROMDD, we partition the binary variables into groups and treat each group as a multivalued variable.

Definition 2.3: Let $f(X)$ be a two-valued logic function, where $X = (x_1, x_2, \ldots, x_n)$ is an ordered set of binary variables. Let $\{X\}$ denote the unordered set of variables in X. Let $X_i \subseteq X$. If $\{X\} = \{X_1\} \cup \{X_2\} \cup \cdots \cup \{X_u\}$, $\{X_i\} \neq \phi$ and $\{X_i\} \cap \{X_j\} = \phi (i \neq j)$, then $(X_1, X_2, \ldots, X_u)$ is a partition of X. X_i is called a super variable. If $|X_i| = k_i$ $(i = 1, 2, \ldots, u)$ and $k_1 + k_2 + \cdots + k_u = n$, then a two-valued logic function $f(X)$ can be represented by a multivalued input two-valued output function that is a mapping $f(X_1, X_2, \ldots, X_u) : P_1 \times P_2 \times P_3 \times \cdots \times P_u \rightarrow B$, where $P_i = \{0, 1, 2, \ldots, 2^{k_i} - 1\}$ and $B = \{0, 1\}$.

Definition 2.4: A fixed-order partition of $X = (x_1, x_2, \ldots, x_n)$ is a partition $(X_1, X_2, \ldots, X_u)$, where

$$
\begin{aligned}
X_1 &= (x_1, x_2, \ldots, x_{k_1}) \\
X_2 &= (x_{k_1+1}, x_{k_1+2}, \ldots, x_{k_1+k_2}) \\
&\vdots \\
X_u &= \left(x_{k_1+k_2+\cdots+k_{u-1}+1}, x_{k_1+k_2+\cdots+k_{u-1}+2}, \ldots, x_{n-1}, x_n\right)
\end{aligned}
$$

and $|X_i| = k_i$. That is, in the fixed-order partition of X, the variable order $(x_1, x_2, \ldots, x_n)$ is fixed.

When the variable order is not fixed (i.e., the partition is not fixed-order partition), we call the partition nonfixed-order partition. In this paper, a partition means fixed-order partition unless stated otherwise.

Example 2.1: Consider (X_1, X_2), which is a fixed-order partition of X, where $X = (x_1, x_2, x_3, x_4, x_5)$, each x_i is a binary variable, and the variable order $(x_1, x_2, x_3, x_4, x_5)$ is fixed. When $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4, x_5)$, we have $k_1 = 2$, $k_2 = 3$, $P_1 = \{0, 1, 2, 3\}$, and $P_2 = \{0, 1, \ldots, 7\}$. Note that X_1 takes four values and X_2 takes eight values. So, a five-variable logic function $f(X)$ can be represented by the multivalued input two-valued output function $f(X_1, X_2) : P_1 \times P_2 \rightarrow B$.

Similarly, we show all possible nonfixed-order partitions of X into (X_1, X_2), where $k_1 = 2$ and $k_2 = 3$. Since the variable order $(x_1, x_2, x_3, x_4, x_5)$ is not fixed, all possible nonfixed-order partitions of X can be considered as

$$
\begin{aligned}
X_1 &= (x_1, x_2), & X_2 &= (x_3, x_4, x_5) \\
X_1 &= (x_1, x_3), & X_2 &= (x_2, x_4, x_5) \\
X_1 &= (x_1, x_4), & X_2 &= (x_2, x_3, x_5) \\
X_1 &= (x_1, x_5), & X_2 &= (x_2, x_3, x_4) \\
X_1 &= (x_2, x_3), & X_2 &= (x_1, x_4, x_5) \\
X_1 &= (x_2, x_4), & X_2 &= (x_1, x_3, x_5) \\
X_1 &= (x_2, x_5), & X_2 &= (x_1, x_3, x_4) \\
X_1 &= (x_3, x_4), & X_2 &= (x_1, x_2, x_5) \\
X_1 &= (x_3, x_5), & X_2 &= (x_1, x_2, x_4) \\
X_1 &= (x_4, x_5), & X_2 &= (x_1, x_2, x_3).
\end{aligned}
$$

D. Heterogeneous MDD

Definition 2.5: When $X = (x_1, x_2, \ldots, x_n)$ is partitioned into $(X_1, X_2, \ldots, X_u)$, an ROMDD representing a multivalued input two-valued output function $f(X_1, X_2, \ldots, X_u)$ is called a heterogeneous MDD. Specially, when $k = |X_i| = |X_2| = \cdots = |X_u|$, an ROMDD for $f(X_1, X_2, \ldots, X_u)$ is a homogeneous MDD and denoted by $\mathrm{MDD}(k)$. A heterogeneous MDD represents a mapping $f : P_1 \times P_2 \times \ldots \times P_u \rightarrow B$ while an $\mathrm{MDD}(k)$ represents a mapping $f : P^u \rightarrow B$, where $P = \{0, 1, \ldots, 2^k - 1\}$, $P_i = \{0, 1, \ldots, 2^{k_i} - 1\}$, and $B = \{0, 1\}$. An $\mathrm{MDD}(k)$ is a special case of heterogeneous MDDs. In a heterogeneous MDD, nonterminal nodes representing a super variable X_i have 2^{k_i} outgoing edges, where k_i denotes the number of binary variables in X_i. Similarly, in an $\mathrm{MDD}(k)$, nonterminal nodes have 2^k outgoing edges.

For n-variable functions f, if $n < ku$ (i.e., n is indivisible by k), additional redundant binary variables are used to construct $\mathrm{MDD}(k)$s. The set of binary variables with such variables is $\{X'\} = \{x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{ku}\}$, where $|X'| = ku$. Note that f is independent of $x_{n+1}, x_{n+2}, \ldots, x_{ku}$.

Example 2.2: Consider a logic function $f = x_1x_2x_3 \vee x_2x_3x_4 \vee x_3x_4x_1 \vee x_4x_1x_2$. Fig. 1(a), (b), (c), and (d) represents the ROBDD, MDD(2), and heterogeneous MDDs for f, respectively. In Fig. 1(a), the solid lines and the dotted lines denote 1-edges and 0-edges, respectively. In Fig. 1(b), the binary variables $X = (x_1, x_2, x_3, x_4)$ are partitioned into (X_1, X_2), where $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4)$. In Fig. 1(c), $X_1 = (x_1, x_2, x_3)$ and $X_2 = (x_4)$. This partition produces a heterogeneous MDD with minimum memory size among heterogeneous MDDs for f. In Fig. 1(d), $X_1 = (x_1)$ and $X_2 = (x_2, x_3, x_4)$. This partition produces a heterogeneous MDD with maximum memory size among heterogeneous MDDs for f.

In this paper, we use shared decision diagrams (SDDs) [22] to represent multiple-output functions $F = (f_0, f_1, \ldots, f_{m-1}) : B^n \rightarrow B^m$, where $B = \{1, 0\}$, and n and m denote the number of input and output variables, respectively. In the following, BDDs and MDDs mean shared BDDs (SBDDs) and shared MDDs (SMDDs), respectively.

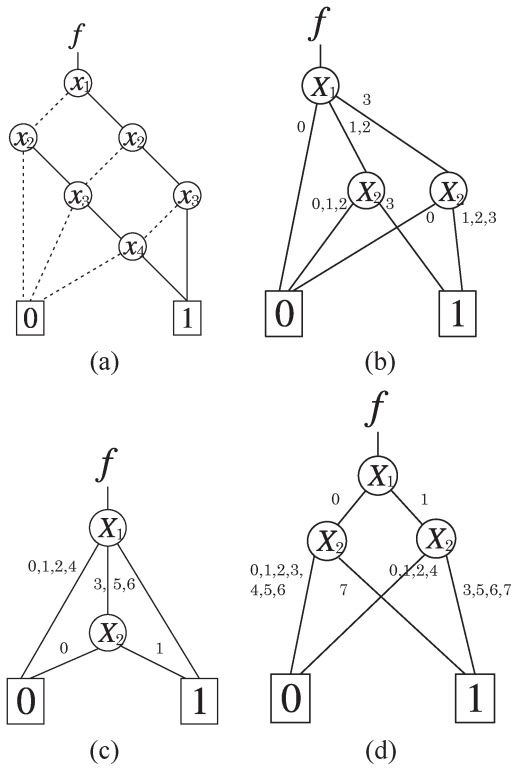

Fig. 1. ROBDD, MDD(2), and heterogeneous MDDs. (a) ROBDD. (b) MDD(2). (c) Minimum heterogeneous MDD. (d) Maximum heterogeneous MDD.

E. Number of Heterogeneous MDDs

This section shows the number of different heterogeneous MDDs to estimate the complexity of optimization for heterogeneous MDDs.

Theorem 2.1: Let $N_{\text{nonfix}}(n)$ be the number of different nonfixed-order partitions of $X = (x_1, x_2, \ldots, x_n)$. Then

$$N_{\text{nonfix}}(n) = \sum_{r=1}^{n} \sum_{i=0}^{r} {}_rC_i (r-i)^n (-1)^i.$$

Proof: See Appendix. ■

Therefore, when both orderings and partitions of the binary variables for an optimization of heterogeneous MDDs are considered, the number of different heterogeneous MDDs for an n-variable logic function is given by $N_{\text{nonfix}}(n)$.

Table I compares the numbers of different ROBDDs, heterogeneous MDDs, and FBDDs for n-variable logic functions, where the number of different ROBDDs is equal to the number of different permutations of variables, that is, $n!$, and the number of different FBDDs S_n is given by [38]

$$S_n = nS_{n-1}^2 = \prod_{k=1}^{n} k^{2^{n-k}}.$$

The number of different heterogeneous MDDs is larger than that of ROBDDs. The number of different FBDDs is much larger than those of ROBDDs and heterogeneous MDDs.

TABLE I
NUMBER OF DIFFERENT DDs FOR n-VARIABLE LOGIC FUNCTIONS

n	ROBDD $n!$	Heterogeneous MDD $N_{\text{nonfix}}(n)$	FBDD S_n
1	1	1	1
2	2	3	2
3	6	13	12
4	24	75	576
5	120	541	1 658 880
6	720	4683	16 511 297 126 400
7	5040	47 293	$> 10^{27}$
8	40 230	545 835	–
9	362 880	7 087 261	–
10	3 628 800	102 247 563	–
11	39 916 800	1 622 632 573	–
12	479 001 600	28 091 567 595	–

For a naive optimization method that finds an optimum solution by enumerating all possible ones, we have the following.

1) An optimization of heterogeneous MDDs is more difficult than that of ROBDDs.
2) Optimizations of heterogeneous MDDs and ROBDDs are much easier than that of FBDDs.

F. Memory Size of DD

In this section, we define the memory sizes of DDs to compare the sizes for different types of DDs.

Definition 2.6: In a DD, the number of nodes in the DD, denoted by nodes (DD), is the sum of all nonterminal nodes.

Definition 2.7: The width of a DD with respect to x_i,[1] denoted by $\text{width}(\text{DD}, i)$, is the number of nodes in the DD corresponding to the variable x_i. The number of nodes in the DD is given by

$$\text{nodes}(\text{DD}) = \sum_{i=1}^{n} \text{width}(\text{DD}, i)$$

where n denotes the number of variables.

Definition 2.8: The memory size of a DD, denoted by Mem(DD), is the number of words needed to store all nonterminal nodes in the DD into a memory, where we assume that a word is large enough to store a variable index or an edge pointer.

To represent a DD, each nonterminal node in the DD requires an index and a set of pointers that refer the succeeding nodes. Since each nonterminal node in a BDD has two pointers, the memory size of a BDD is given by

$$\text{Mem}(\text{BDD}) = (2+1) \times \text{nodes}(\text{BDD}). \tag{2.1}$$

Similarly, since each nonterminal node in an $\text{MDD}(k)$ has 2^k pointers, the memory size of an $\text{MDD}(k)$ is given by

$$\text{Mem}\left(\text{MDD}(k)\right) = (2^k + 1) \times \text{nodes}\left(\text{MDD}(k)\right).$$

In a heterogeneous MDD, each super variable can take different domains. Therefore, the memory size of a heterogeneous MDD

[1] Note that this definition differs from that of "width of BDDs" in [23].

is the sum of sizes for the super variables

$$\mathrm{Mem}(\text{heterogeneous MDD}) = \sum_{i=1}^{u}(2^{k_i}+1) \times \mathrm{width}(\text{heterogeneous MDD}, i).$$

where u and k_i denote the number of super variables and the number of binary variables in a super variable X_i, respectively.

Example 2.3: The memory sizes of ROBDD, MDD(2), and heterogeneous MDDs are as follows: for the ROBDD in Fig. 1(a), it is 18; for the MDD(2) in Fig. 1(b), it is 15; for the heterogeneous MDD in Fig. 1(c), it is 12; and for the heterogeneous MDD in Fig. 1(d), it is 21.

Definition 2.9: Given a logic function f and the order of input variables, the fixed-order minimum heterogeneous MDD for the logic function f is the heterogeneous MDD with minimum memory size among the heterogeneous MDDs with fixed-order partitions of the variables.

Definition 2.10: Given a logic function f, the minimum heterogeneous MDD for the logic function f is the heterogeneous MDD with minimum memory size among the heterogeneous MDDs with nonfixed-order partitions of the variables.

Theorem 2.2 [27]: Consider an ROBDD and a heterogeneous MDD for an n-variable logic function that is not a constant function. When an order of binary variables is fixed, for the number of nodes in the ROBDD and the memory size of heterogeneous MDD obtained by considering only the fixed-order partitions, the following relation holds:

$$\mathrm{Mem}(\text{heterogeneous MDD}) \geq \mathrm{nodes}(\mathrm{ROBDD}) + 2.$$

Theorem 2.3: Consider an ROBDD and a heterogeneous MDD for an n-variable logic function that is not a constant function. When an order of binary variables is fixed, for the memory sizes of ROBDD and heterogeneous MDD obtained by considering only the fixed-order partitions, the following relation holds:

$$\frac{\mathrm{Mem}(\text{heterogeneous MDD})}{\mathrm{Mem}(\mathrm{ROBDD})} > \frac{1}{3}.$$

Proof: See Appendix. ■

Theorem 2.4: Assume that the number of nodes in an ROBDD for an n-variable function f is the upper bound [17]

$$2^{n-r} + 2^{2^r} - 3$$

where r is the largest integer satisfying $n - r \geq 2^r$. Let $\mathrm{Mem}_{\min}(\mathrm{MDD})$ be the memory size of the minimum heterogeneous MDD for f. Let $s = 2^r + r - n$, where $0 \leq s \leq 2^r$. When n is large and $2 \leq s \leq 2^r$, the following relation holds:

$$\frac{\mathrm{Mem}_{\min}(\mathrm{MDD})}{\mathrm{Mem}(\mathrm{ROBDD})} \simeq \frac{2^s+3}{3(2^s+1)}.$$

Proof: See Appendix. ■

Corollary 2.1: In Theorem 2.4, when n is sufficiently large and $s = 0$ or $9 \leq s \leq 2^r$, the following relation holds:

$$\frac{\mathrm{Mem}_{\min}(\mathrm{MDD})}{\mathrm{Mem}(\mathrm{ROBDD})} \simeq 0.33.$$

Proof: See Appendix. ■

Property 2.1: Consider a logic function $f(X)$. Let $\mathrm{Mem}_{\min}(f)$ be the memory size of a fixed-order minimum heterogeneous MDD for f. When f is decomposed into $f = g(h(X_1), X_2)$, let $\mathrm{Mem}_{\min}(g)$ and $\mathrm{Mem}_{\min}(h)$ be the memory sizes of fixed-order minimum heterogeneous MDDs for g and h, respectively. For many benchmark functions, the following two relations hold:

$$\mathrm{Mem}_{\min}(f) > \mathrm{Mem}_{\min}(g)$$
$$\mathrm{Mem}_{\min}(f) > \mathrm{Mem}_{\min}(h).$$

G. APL of DD

Definition 2.11: In a DD, a sequence of edges and nonterminal nodes leading from the root node to a terminal node is a path. The number of nonterminal nodes on the path is the path length.

Definition 2.12: The APL of a DD, denoted by APL(DD), is the sum of path lengths for all assignments of values to the variables divided by the number of the assignments.

In this paper, the APL of an SDD for multiple-output function $F = (f_0, f_1, \ldots, f_{m-1})$ is the sum of the APLs of individual DDs for each function f_i.

In this paper, we assume the following computational model.[2]

1) The logic functions are evaluated by traversing DDs from the root node to a terminal node according to the values of the input variables.
2) Encoded input values are available, and their access time is negligible. For example, when $X_1 = (x_1, x_2, x_3, x_4) = (1, 0, 0, 1)$, $X_1 = 9$ is immediately available as an input to the algorithm.
3) Most computation time is devoted to accessing nodes.
4) The evaluation times for all DD nodes are the same.

In this case, the average evaluation time of a DD is proportional to the APL of the DD. Thus, in this model, we use the APL to compare the evaluation times of different types of DDs.

Example 2.4: The APLs for different DDs are as follows. For the ROBDD in Fig. 1(a), $\mathrm{APL}(\mathrm{ROBDD}) = 3.125$. For the MDD(2) in Fig. 1(b), $\mathrm{APL}[\mathrm{MDD}(2)] = 1.75$. For the heterogeneous MDD in Fig. 1(c), $\mathrm{APL}(\text{heterogeneous MDD}) = 1.375$. For the heterogeneous MDD in Fig. 1(d), $\mathrm{APL}(\text{heterogeneous MDD}) = 2.0$.

Theorem 2.5: When the number of nodes in an ROBDD for an n-variable function f is the upper bound [17]

$$2^{n-r} + 2^{2^r} - 3$$

[2]This model can be implemented by the dedicated hardware proposed in [13].

Algorithm 3.1:

```
     /* X: set of binary variables. */
 1:  exhaustive_search_memory (BDD) {
 2:    min_memory = minimize_memory (BDD);
 3:    for (all permutations of X) {
 4:      Change the variable order for BDD;
 5:      if (min_memory < nodes(BDD) + 2)
 6:        continue;
 7:      current_memory = minimize_memory (BDD);
 8:      if (current_memory < min_memory) {
 9:        min_memory = current_memory;
10:        Record the variable order for the BDD;
11:        Record the partition of X;
12:      }
13:    }
14:  }
```

Fig. 2. Exact memory size minimization algorithm for heterogeneous MDDs.

where r is the largest integer satisfying $n - r \geq 2^r$, there exists a heterogeneous MDD for f that satisfies

$$\mathrm{APL}(\text{heterogeneous MDD}) \leq 2.0$$

$$\mathrm{Mem}(\text{heterogeneous MDD}) \leq \mathrm{Mem}(\text{ROBDD}).$$

Proof: See Appendix. ■

III. Optimization Algorithms of Heterogeneous MDDs

Since memory size and APL of a heterogeneous MDD depend on the partition of binary variables as well as the ordering of binary variables, the memory size and APL of a heterogeneous MDD can be minimized by considering both orderings and partitions of binary variables.

In this section, we formulate the memory size and APL minimization problems of heterogeneous MDDs considering both orderings and partitions of binary variables. We also present exact minimization algorithms to solve them and heuristic minimization algorithms.

A. Memory Size Minimization

We formulate the memory size minimization problem of heterogeneous MDDs considering both orderings and partitions of binary variables as follows.

Problem 3.1: Given a logic function $f(X)$, find an ordering and a partition of X that produce the minimum heterogeneous MDD for f.

Fig. 2 shows a pseudocode to solve Problem 3.1. It uses an ROBDD for the given logic function as the internal representation. In the second and seventh lines in Fig. 2, minimize_memory [29] finds an optimum fixed-order partition that produces the fixed-order minimum heterogeneous MDD. The subprocedure minimize_memory is based on dynamic programming, and its time and space complexities are $O(n^2N)$ and $O(N)$, respectively, where n is the number of binary variables and N is the number of nodes for the given BDD. This subprocedure consumes 50% to 80% of the total computation time for Algorithm 3.1. In the fifth line, Theorem 2.2 is used to reduce the computation time. This algorithm finds the minimum heterogeneous MDD by exhaustive search.

Algorithm 3.2:

```
     /* X: set of binary variables. */
 1:  sifting_memory (BDD) {
 2:    cost = minimize_memory (BDD);
 3:    do {
 4:      for (∀x_i ∈ X) {
 5:        best_p = current position of x_i;
 6:        for (all position p) {
 7:          Move x_i to position p;
 8:          memory = minimize_memory (BDD);
 9:          Compute L_mem;
10:          if (cost ≤ L_mem)
11:            break;
12:          if (memory < cost) {
13:            cost = memory;
14:            best_p = p;
15:            Record the partition of X;
16:          }
17:        }
18:        Move x_i to best_p;
19:      }
20:    } while (cost is reduced);
21:  }
```

Fig. 3. Heuristic memory size minimization algorithm for heterogeneous MDDs.

However, as shown in Section II-E, when the number of binary variables is large, finding a minimum heterogeneous MDD within a reasonable time is difficult. Thus, we developed a heuristic minimization for heterogeneous MDDs using the sifting algorithm [32] and the fixed-order partition algorithm minimize_memory [29]. The sifting algorithm repeatedly performs the following basic steps:

1) Change the variable order;
2) Compute the cost.

Most sifting algorithms for minimizing the number of nodes in an ROBDD use the number of nodes as the cost. The memory size minimization algorithm for a heterogeneous MDD, however, uses the memory size of the heterogeneous MDD as the cost. The computation of the cost (i.e., minimize_memory) consumes 50% to 80% of total computation time for this heuristic approach. Fig. 3 shows a pseudocode for the heuristic minimization algorithm. In this algorithm, each variable x_i is sifted across all possible positions to determine its best position. First, x_i is sifted in one direction to the closer extreme (top or bottom). Then, x_i is sifted in the opposite direction to the other extreme. In the tenth line in Fig. 3, Property 2.1 is used to find useful siftings of x_i. The L_{mem} in the ninth line denotes the memory size of fixed-order minimum heterogeneous MDD for logic function g or h obtained by functional decomposition $f(X) = g(h(X_1), X_2)$. When x_i moves down to the bottom of the ROBDD, we use h to compute L_{mem}, where X_1 contains the binary variables that are above the level of x_i in the variable order and X_2 contains the remaining ones. If $\mathrm{cost} \leq L_{\mathrm{mem}}$, we stop the sifting of x_i to the bottom because sifting of x_i further down to the bottom seldom reduces the memory size due to Property 2.1. Similarly, when x_i moves up to the top of the ROBDD, we use g to compute L_{mem}, where X_2 contains the binary variables that are below the level of x_i

Algorithm 3.3:

```
     /* X: set of binary variables. */
     /* L: memory size limitation. */
 1:  exhaustive_search_APL (BDD, L) {
 2:    min_APL = minimize_APL (1, L, BDD);
 3:    for (all permutations of X) {
 4:      Change the variable order for BDD;
 5:      if (L < nodes(BDD)+ 2)
 6:        continue;
 7:      memory = minimize_memory (BDD);
 8:      if (L < memory)
 9:        continue;
10:      APL = minimize_APL (1, L, BDD);
11:      if (APL < min_APL) {
12:        min_APL = APL;
13:        Record the variable order for the BDD;
14:        Record the partition of X;
15:      }
16:    }
17:  }
```

Fig. 4. Exact APL minimization algorithm for heterogeneous MDDs.

in the variable order and X_1 contains the remaining ones. This bounding method is similar to the one shown in [7] that reduces the computation time of the classical sifting for node minimization.

B. APL Minimization

For any n-variable logic function $f(X)$, the trivial partition of X, where $X = X_1$ and $|X_1| = n$, produces a heterogeneous MDD with the smallest APL (i.e., $\mathrm{APL} = 1.0$), independently of the variable ordering. However, since the memory size of the heterogeneous MDD for the trivial partition is nearly 2^n, such a heterogeneous MDD is too large in most cases. Therefore, an ordering and a partition of X that minimize the APL within a given memory size limitation are sought. We formulate the APL minimization problem considering both orderings and partitions of binary variables as follows.

Problem 3.2: Given a logic function $f(X)$ and a memory size limitation L, find an ordering and a partition of X that produce the heterogeneous MDD with the minimum APL and with memory size equal to or smaller than L.

Fig. 4 shows the pseudocode to solve Problem 3.2. In the second and tenth lines in Fig. 4, the subprocedure minimize_APL [26], [29] finds an optimum fixed-order partition that minimizes the APL of a heterogeneous MDD. Since it is a recursive procedure, the top level for ROBDD (i.e., $\mathrm{level} = 1$) is required as the initial argument. The subprocedure minimize_APL is based on a branch-and-bound method, and its time and space complexities are $O(2^n + n^2N)$ and $O(N)$, respectively, where n is the number of binary variables and N is the number of nodes for the given BDD. Although the worst case time complexity for this subprocedure is high, the actual computation time is short. For example, when $n = 256$ (for des), the computation time for this subprocedure is 0.44 CPU seconds (refer to [29] for more details). This subprocedure consumes 60% to 70% of the total computation time for Algorithm 3.3. Algorithm 3.3 finds an optimum solution for Problem 3.2 by exhaustive search.

Algorithm 3.4:

```
     /* X: set of binary variables. */
     /* L: memory size limitation. */
     /* R: #sifting rounds. */
 1:  sifting_APL (BDD, L, R) {
 2:    cost = minimize_APL (1, L, BDD);
 3:    for (r = 0; r < R; r++) {
 4:      for (∀x_i ∈ X) {
 5:        best_p = current position of x_i;
 6:        for (all positions p) {
 7:          Move x_i to position p;
 8:          memory = minimize_memory (BDD);
 9:          Compute L_mem;
10:          if (L ≤ L_mem)
11:            break;
12:          if (L < memory)
13:            continue;
14:          APL = minimize_APL (1, L, BDD);
15:          if (APL< cost) {
16:            cost = APL;
17:            best_p = p;
18:            Record the partition of X;
19:          }
20:        }
21:        Move x_i to best_p;
22:      }
23:    }
24:  }
```

Fig. 5. Heuristic APL minimization algorithm for heterogeneous MDDs.

As well as the memory size minimization, Algorithm 3.3 is time consuming for functions with many inputs. Thus, we developed a heuristic APL minimization algorithm for heterogeneous MDDs using a sifting algorithm [32] and the fixed-order partition algorithm minimize_APL [26], [29]. The subprocedure minimize_APL consumes 70%–80% of the total computation time for this heuristic approach. Fig. 5 shows a pseudocode for the heuristic APL minimization algorithm. In this algorithm, the APL of a heterogeneous MDD is used as the cost for the sifting algorithm. The APL of a heterogeneous MDD can be computed using a method similar to the APL of ROBDDs in [28]. In the tenth line in Fig. 5, Property 2.1 is used to find useful siftings of x_i. If $L \leq L_{\mathrm{mem}}$, we stop the sifting of x_i because the further sifting of x_i seldom finds a smaller memory size than L_{mem} due to Property 2.1. That is, in most cases, the further sifting of x_i produces heterogeneous MDDs with larger memory size than the memory size limitation L when $L \leq L_{\mathrm{mem}}$.

IV. Experimental Results

To show the compactness of heterogeneous MDD and the efficiency of optimization algorithms, we compared heterogeneous MDDs with the different types of DDs using benchmark functions. Experiments were conducted in the following environment:

- CPU: Pentium 4 Xeon 2.8 GHz;
- L1 Cache: 32 KB;
- L2 Cache: 512 KB;
- Main Memory: 4 GB;
- Operating System: redhat (Linux 7.3);
- C-Compiler: gcc -O2.

TABLE II
MEMORY SIZES OF ROBDDS, FBDDS, AND HETEROGENEOUS MDDS FOR ALL FOUR-VARIABLE LOGIC FUNCTIONS

Group	ROOBDD			FBDD			Heterogeneous MDD		
No.	Mem	#class	#function	Mem	#class	#function	Mem	#class	#function
0	0	1	2	0	1	2	0	1	2
1	3	1	8	3	1	8	3	1	8
2	6	1	48	6	1	48	5	1	48
3	9	4	364	9	4	364	5	1	12
							8	3	352
4	12	14	3168	12	14	3168	8	3	320
							9	1	96
							10	6	1216
							11	4	1536
5	15	38	12 440	15	38	12 440	9	3	104
							10	7	1056
							11	13	4400
							12	12	6528
							14	3	352
6	18	70	22 488	18	70	22 488	10	3	168
							12	41	12 064
							14	13	4928
							15	13	5328
7	21	68	20 346	18	3	1536	12	11	3520
				21	65	18 810	15	57	16 826
8	24	25	6672	21	10	4032	15	25	6672
				24	15	2640			
Avg.	1.00	–	–	0.99	–	–	0.72	–	–

A. Comparison With FBDDs

In this section, we compare heterogeneous MDDs with FBDDs to show the compactness of heterogeneous MDDs.

We implemented Algorithm 3.1 and compared the minimum heterogeneous MDDs with the minimum ROBDDs and the minimum FBDDs for all four- and five-variable logic functions. To compare them, we classified all the logic functions into NPN equivalence classes [24], [35]. For the four-variable case, 65 536 functions are classified into 222 NPN equivalence classes, and for the five-variable case, 4 294 967 296 functions are classified into 616 126 NPN equivalence classes. Table II compares minimum DD sizes for the four-variable case. In Table II, the NPN representative functions are grouped into nine rows according to the memory size of the minimum ROBDD. The column "Mem" denotes the memory size of each DD. The columns "#class" and "#function" in Table II denote the number of NPN equivalence classes and the number of functions included in the classes, respectively. The bottom row "Avg." denotes the arithmetic average of the relative memory sizes for all functions, where the memory size of ROBDD is set to 1.00. Note that ROBDDs, FBDDs, and heterogeneous MDDs in this table do not use complemented edges [3], [22].

For the four-variable case, FBDDs are smaller than ROBDDs for 5568 functions, 8.5% of all functions, while heterogeneous MDDs are smaller than ROBDDs and FBDDs for all functions except for ten degenerate functions (0, 1, x_i, and $\bar{x}_i$, where $i = 1, 2, 3, 4$). For these ten functions, the memory sizes of ROBDDs, FBDDs, and heterogeneous MDDs are equal. On average over all functions, minimum FBDDs require 99% of the memory size of minimum ROBDDs while minimum heterogeneous MDDs require 72% of the memory size for minimum ROBDDs.

For the five-variable case, FBDDs are smaller than ROBDDs for 1 938 548 576 functions, 45% of all functions, while heterogeneous MDDs are smaller than ROBDDs for 4 294 967 284 functions, 99% of all functions. Also, heterogeneous MDDs are smaller than FBDDs for 4 294 921 204 functions, 99% of all functions, and for the others, heterogeneous MDDs are equal in size to FBDDs. There was no function whose FBDD was smaller than the heterogeneous MDD. On average over all functions, minimum FBDDs require 96% of the memory size for minimum ROBDDs while minimum heterogeneous MDDs require 67% of the memory size for minimum ROBDDs.

TABLE III
MEMORY SIZES OF ROBDDS, FBDDS, AND HETEROGENEOUS MDDS FOR HWB FUNCTIONS

n	ROBDD	FBDD	MDD
4	21	18	12
5	36	33	20
6	54	51	32
10	222	219	170
12	381	381	278
14	624	543	468
16	963	870	746
18	1473	1284	1166
20	2217	2079	1717
22	3291	2985	2575
Avg.	1.00	0.92	0.71

Also for the hidden weighted bit (HWB) functions [6], we compared the memory sizes of DDs. The HWB function is defined as

$$\mathrm{HWB}(X) = \begin{cases} 0, & \text{when } wt(X) = 0 \\ x_{wt(X)}, & \text{when } wt(X) > 0 \end{cases}$$

where $X = (x_1, x_2, \ldots, x_n)$ and the weight function $wt(X)$ is given by

$$wt(X) = x_1 + x_2 + \cdots + x_n$$

where x_i is a binary variable and $+$ denotes the integer addition. The number of nodes in ROBDDs for the HWB function is an exponential function of n [6]. Table III compares the memory sizes of DDs for HWB functions. We took the numbers of nodes for ROBDDs and FBDDs from [10] and [11] and calculated the memory sizes of ROBDDs and FBDDs using (2.1) in Section II-F. The column labeled "n" denotes the number

TABLE IV
MEMORY SIZES OF ROBDDS, FBDDS, AND HETEROGENEOUS MDDS FOR MCNC BENCHMARK FUNCTIONS

Name	In	Out	Memory Size		
			ROBDD	FBDD	MDD
C432	36	7	3189	3171	2824
C499	41	32	77595	77595	59739
C880	60	26	12156	8394	11812
C1908	33	25	16575	15141	13493
C2670	233	64	5319	3186	4649
C3540	50	22	71481	62997	65029
C5315	178	123	5154	4434	4582
C7552	207	107	6633	4782	6119
alu4	14	8	1047	900	855
apex1	45	45	3735	3531	3016
apex6	135	99	1491	1365	1414
cps	24	102	2910	2706	2533
dalu	75	16	2064	1947	1548
des	256	245	8832	8706	7288
frg2	143	139	2886	2760	2671
i3	132	6	396	396	330
i8	133	81	3825	3570	3662
i10	257	224	61977	56439	55766
k2	45	45	3735	3408	3018
too_large	38	3	954	858	857
vda	17	39	1431	1401	1088
Average of Ratios			1.00	0.90	0.86

of binary variables for HWB functions. The column labeled "MDD" denotes the memory sizes of heterogeneous MDDs. For $n = 4$ to 6, the memory sizes of minimum DDs obtained by exact minimization algorithms are given. For $n = 10$ to 22, the memory sizes of DDs obtained by heuristic minimization algorithms are given, that is, they may not be an exact minimum. The bottom row "Avg." denotes the arithmetic average of the relative memory sizes, where the memory size of ROBDD is set to 1.00. Note that ROBDDs, FBDDs, and heterogeneous MDDs in this table use complemented edges to make our results compatible with the results in [10] and [11].

For these functions, on the average, FBDDs require 92% of the memory size for ROBDDs while heterogeneous MDDs require 71% of the memory size for ROBDDs.

Algorithm 3.1 could obtain exact minimum heterogeneous MDDs for functions with up to 12 inputs within a reasonable computation time while the exact FBDD minimization algorithm [10] can find the minimum one for functions with up to eight inputs.

Table IV compares heterogeneous MDDs with ROBDDs and FBDDs for selected MCNC benchmark functions. The ROBDDs are obtained by the best known variable orders [37], and the numbers of nodes for FBDDs are taken from [10] and [11]. Table IV includes the same benchmark functions as the experiments in [10] and [11] except for alu2. Unfortunately, the variable order of ROBDD for alu2 is not shown in [37]. The memory sizes of ROBDDs and FBDDs are calculated by (2.1) in Section II-F. The columns "In" and "Out" in Table IV denote the number of inputs and outputs for each benchmark function, respectively. Column "MDD" denotes the heterogeneous MDDs obtained by Algorithm 3.2, where the ROBDDs [37] are used as initial solutions. The DDs in this table may not be the exact minimum since the algorithms are heuristic methods. The bottom row "Average of Ratios" denotes the arithmetic average of the relative memory sizes, where the memory size of ROBDD is set to 1.00. Note that ROBDDs, FBDDs, and heterogeneous MDDs in this table use complemented edges to make our results compatible with the results in [10], [11], and [37].

Heterogeneous MDDs require smaller memory sizes than FBDDs for 14 out of 21 benchmark functions in Table IV. Especially, for C499, dalu, and vda, heterogeneous MDDs require at most 80% of the memory sizes for the FBDDs.

B. Comparison With ROBDDs

Table V compares the memory sizes and the APLs of ROBDDs and heterogeneous MDDs for n-variable logic functions. The ROBDDs and heterogeneous MDDs were optimized using four different algorithms: 1) exact nodes minimization algorithm for an ROBDD considering only the orderings (column "MinNodes"); 2) exact APL minimization algorithm for an ROBDD considering only the orderings of binary variables (column "MinAPL_B"); 3) exact memory size minimization algorithm for a heterogeneous MDD (Algorithm 3.1) considering both orderings and partitions of binary variables (column "MinMem"); and 4) exact APL minimization algorithm for a heterogeneous MDD (Algorithm 3.3) considering both orderings and partitions of binary variables (column "MinAPL_M"). The memory size limitations L for Algorithm 3.3 were set to the memory sizes of the ROBDDs in "MinNodes." The values in this table are the normalized averages of n-variable logic functions, where the memory sizes and APLs of "MinNodes" are set to 1.00. Columns "MinAPL_B," "MinMem," and "MinAPL_M" show the relative values of the memory sizes and APLs to "MinNodes." Column "#samples" denotes the number of sample functions used for each n-variable function. Note that the ROBDDs and heterogeneous MDDs in this table do not use complemented edges.

For four- and five-variable logic functions, we calculated the exact averages over all functions. This was done by recognizing that the minimum memory size and the APL for a function in one NPN equivalence class [24], [35] are identical to the minimum memory sizes and APLs for other functions in the same class. Thus, it is sufficient to consider only one function from each class and form a sum weighted by the size of each class. For a larger n, there are too many NPN equivalence classes. For $6 \leq n \leq 10$, we generated 1000 pseudorandom n-variable logic functions with different number of minterms and calculated the normalized averages for them.

For ROBDDs, APLs can be reduced by up to 97% of ROBDDs with the minimum nodes but the memory sizes increase to 108%. On the other hand, for heterogeneous MDDs, the APLs can be reduced by up to 17% of ROBDDs with the minimum nodes without increasing memory sizes, and both the memory sizes and APLs can be reduced by up to 54% and 26% of minimum ROBDDs, respectively. Table V shows that the relative values of memory sizes and APLs for heterogeneous MDDs decrease as the number of binary variables n increases. Algorithm 3.3 finds exact minimum APLs of heterogeneous MDDs for functions with up to 11 variables within a reasonable computation time.

To demonstrate Theorem 2.4, Corollary 2.1, and Theorem 2.5, we randomly generated n-variable functions with 2^{n-1} minterms for $n = 22$ to 29. For randomly generated n-variable functions with 2^{n-1} minterms, the number of nodes in ROBDD is nearly equal to the upper bound [25], [34].

TABLE V
MEMORY SIZES AND APLS OF ROBDDS AND HETEROGENEOUS MDDS FOR n-VARIABLE LOGIC FUNCTIONS

	Memory Size				APL				
	ROBDD		Heterogeneous MDD		ROBDD		Heterogeneous MDD		
n	MinNodes	MinAPL$_B$	MinMem	MinAPL$_M$	MinNodes	MinAPL$_B$	MinMem	MinAPL$_M$	#samples
4	1.00	1.07	0.72	0.86	1.00	0.99	0.52	0.37	2^{16}
5	1.00	1.07	0.67	0.91	1.00	0.98	0.39	0.27	2^{32}
6	1.00	1.08	0.68	0.80	1.00	0.97	0.45	0.32	1000
7	1.00	1.08	0.64	0.79	1.00	0.97	0.40	0.27	1000
8	1.00	1.08	0.58	0.81	1.00	0.97	0.33	0.22	1000
9	1.00	1.07	0.55	0.83	1.00	0.98	0.29	0.19	1000
10	1.00	1.06	0.54	0.84	1.00	0.98	0.26	0.17	1000

TABLE VI
MEMORY SIZES AND APLS OF ROBDDS AND HETEROGENEOUS MDDS FOR RANDOMLY GENERATED FUNCTIONS

n	r	Memory Size					APL		
		ROBDD	Heterogeneous MDD		BDD / MDD	γ	ROBDD	Heterogeneous MDD	
			MinMem	MinAPL				MinMem	MinAPL
22	4	979 053	455 261	526 575	0.47	0.47	21.18	4.18	1.99
23	4	1 769 307	720 825	1 050 863	0.41	0.41	22.18	4.18	1.99
24	4	3 342 030	1 245 179	2 099 439	0.37	0.37	23.18	4.18	1.99
25	4	6 487 176	2 293 755	4 196 591	0.35	0.35	24.18	4.18	1.99
26	4	12 777 186	4 390 907	8 390 895	0.34	0.34	25.18	4.18	1.99
27	4	25 353 708	8 585 211	16 779 503	0.34	0.34	26.18	4.18	1.99
28	4	50 499 738	16 973 819	33 556 719	0.34	0.34	27.18	4.18	1.99
29	4	100 757 271	33 751 035	67 111 151	0.33	0.33	28.18	4.18	1.99

TABLE VII
MEMORY SIZES AND APLS OF ROBDDS AND HETEROGENEOUS MDDS FOR MCNC BENCHMARK FUNCTIONS

			Memory Size				APL			
			ROBDD		Heterogeneous MDD		ROBDD		Heterogeneous MDD	
Name	In	Out	MinNodes	MinAPL$_B$	MinMem	MinAPL$_M$	MinNodes	MinAPL$_B$	MinMem	MinAPL$_M$
C432	36	7	3189	3243	2824	3179	86.58	86.24	55.74	45.45
C499	41	32	77 595	96 315	59 739	77 589	813.64	641.16	381.14	192.52
C880	60	26	12 156	54 810	11 812	12 154	135.79	121.03	125.73	99.13
C1908	33	25	16 575	56 328	13 493	16 564	254.35	183.61	145.81	92.09
C2670	233	64	5319	8286	4649	5319	214.05	202.08	167.90	133.78
C3540	50	22	71 481	74 292	65 029	71 480	209.15	208.06	141.10	91.78
C5315	178	123	5154	5460	4582	5153	462.05	446.26	373.23	304.38
C7552	207	107	6633	6585	6119	6633	484.03	469.54	424.85	314.03
alu4	14	8	1047	1080	855	1019	40.81	40.70	24.41	19.59
apex1	45	45	3735	4254	3016	3728	180.59	177.69	87.35	67.63
apex6	135	99	1491	1887	1414	1490	291.54	230.91	260.66	231.06
cps	24	102	2910	4656	2533	2906	290.25	235.39	187.90	151.81
dalu	75	16	2064	2970	1548	2064	102.67	78.81	39.40	28.09
des	256	245	8832	9177	7288	8831	1210.00	1080.38	910.63	687.50
frg2	143	139	2886	5070	2671	2884	624.69	322.17	499.27	348.60
i3	132	6	396	396	330	396	26.76	26.76	17.84	12.61
i8	133	81	3825	6954	3662	3825	302.54	270.82	229.12	207.54
i10	257	224	61 977	685 215	55 766	61 974	1084.96	776.10	887.62	614.53
k2	45	45	3735	4254	3018	3728	180.52	177.69	87.32	67.61
too_large	38	3	954	2361	857	954	13.16	11.52	8.47	6.24
vda	17	39	1431	1515	1088	1424	176.34	171.54	81.72	69.54
Average of Ratios			1.00	2.03	0.86	1.00	1.00	0.88	0.67	0.51

Table VI shows the memory sizes and APLs of ROBDDs and heterogeneous MDDs. The column labeled "r" shows the largest integer satisfying $n - r \geq 2^r$. The column "MinMem" denotes the heterogeneous MDDs obtained by Algorithm 3.2. The column "MinAPL" denotes the heterogeneous MDDs obtained by Algorithm 3.4. The memory size limitations L for Algorithm 3.4 are set to the memory sizes of the ROBDDs. Column "BDD/MDD" shows the values given by

$$\frac{\mathrm{Mem}(\text{heterogeneous MDD in "MinMem"})}{\mathrm{Mem}(\mathrm{ROBDD})}.$$

The column "γ" denotes the ratio used in Theorem 2.4, i.e.,

$$\gamma = \frac{2^s + 3}{3(2^s + 1)}$$

where $s = 2^r + r - n$. Note that the ROBDDs and heterogeneous MDDs in this table do not use complemented edges. The memory sizes and APLs in this table may not be an exact minimum since the algorithms are heuristic methods.

Table VI shows that for these randomly generated functions with 2^{n-1} minterms, Theorem 2.4, Corollary 2.1, and Theorem 2.5 hold; that is, BDD/MDD approaches 0.33 when n is large, "MinAPL" is smaller than 2.0, and Mem(heterogeneous MDD) $\leq$ Mem(ROBDD).

Table VII compares memory sizes and APLs of ROBDDs and heterogeneous MDDs for the same MCNC benchmark functions as Table IV. Columns labeled "MinNodes" denote the ROBDDs obtained by the best known variable orders [37]. These were used as the initial ROBDDs for the algorithms in this experiment. Columns "MinAPL$_B$" denote the ROBDDs obtained by the sifting algorithm for the APLs [28]. Columns "MinMem" denote the heterogeneous MDDs obtained by

TABLE VIII
CPU TIMES (SECOND) FOR MEMORY SIZE AND APL MINIMIZATION ALGORITHMS

	ROBDD	Heterogeneous MDD	
Name	MinAPL_B	MinMem	MinAPL_M
C432	0.23	0.23	1.04
C499	10.76	5.12	698.31
C880	4.54	1.23	22.09
C1908	1.44	0.67	27.38
C2670	2.21	1.99	1957.51
C3540	12.74	36.96	523.45
C5315	0.43	1.31	3663.57
C7552	1.35	4.76	2258.88
alu4	0.02	0.02	0.05
apex1	0.11	0.29	36.07
apex6	0.05	0.33	79.47
cps	0.09	0.12	0.80
dalu	0.15	0.25	132.41
des	0.91	3.59	60144
frg2	0.29	0.89	218.46
i3	0.01	0.23	95.69
i8	0.31	0.59	30.15
i10	160.91	69.27	71464
k2	0.11	0.29	33.99
too_large	0.07	0.07	0.31
vda	0.02	0.01	0.15

Algorithm 3.2. And columns "MinAPL_M" denote the heterogeneous MDDs obtained by Algorithm 3.4. The memory size limitations L for Algorithm 3.4 were set to the memory sizes of the ROBDD in "MinNodes." In the sifting algorithm [28] and Algorithm 3.4, the number of rounds of sifting was set to two. Note that the ROBDDs and heterogeneous MDDs in this table use complemented edges. The memory sizes and APLs in this table may not be an exact minimum since the algorithms are heuristic methods. The row labeled "Average of Ratios" represents the normalized averages of memory size and APL, where the memory size and the APL of "MinNodes" are set to 1.00.

The sifting algorithm [28] reduced APLs to 88% of "MinNodes," on average, but doubled the memory sizes. Especially, for C880, C1908, i10, and too_large, the sifting algorithm increased the memory sizes significantly. On the other hand, by considering both orderings and partitions of binary variables, Algorithm 3.2 reduced both memory sizes and APLs to 86% and 67% of "MinNodes," respectively. Algorithm 3.4 reduced APLs to 51% of "MinNodes" without increasing the memory size.

C. Comparison of Computation Time for Algorithms

Table VIII compares the computation times for the sifting algorithm for the APLs [28], Algorithm 3.2, and Algorithm 3.4. The values in Table VIII show the CPU times needed to obtain the ROBDDs and heterogeneous MDDs in Table VII, in seconds.

Although Algorithm 3.2 considers both orderings and partitions of binary variables for memory size minimization, its computation time is as short as that of the sifting algorithm that considers only variable orderings for APL minimization. Algorithm 3.4 requires longer computation times than the other two algorithms since Algorithm 3.4 keeps the memory size within the limitation as well as minimizes the APL.

D. Comparison With $\mathrm{MDD}(k)$s

Similarly, we compared heterogeneous MDDs with $\mathrm{MDD}(k)$s.

Tables IX and X compare the memory sizes and APLs of ROBDDs, heterogeneous MDDs, and $\mathrm{MDD}(k)$s for n-variable logic functions, respectively. In these tables, $\mathrm{MDD}(k)$s have the exact fewest nodes. The values in these tables are the normalized averages of n-variable logic functions, where the memory sizes and APLs of ROBDD with the fewest nodes (column "ROBDD") are set to 1.00. Columns "MinMem," "MinAPL_M," "MDD(2)s," "MDD(3)," "MDD(4)," and "MDD(5)" show the relative values of the memory sizes and APLs to "ROBDD."

From Tables IX and X, it can be seen that: for n-variable logic functions, heterogeneous MDDs obtained by Algorithm 3.3 have the APLs as small as MDD(5)s. The memory sizes of MDD(5)s are twice the memory sizes of ROBDDs. On the other hand, heterogeneous MDDs have smaller memory sizes than ROBDDs.

Tables XI and XII compare the memory sizes and APLs of ROBDDs, heterogeneous MDDs, and $\mathrm{MDD}(k)$s for MCNC benchmark functions, respectively. $\mathrm{MDD}(k)$s in these tables are obtained by the minimization algorithm in [33]. ROBDDs and heterogeneous MDDs are the same as those in Table VII.

Tables XI and XII show that in heterogeneous MDDs, APLs can be reduced to half of the ROBDDs without increasing memory sizes. On the other hand, in $\mathrm{MDD}(k)$s, to reduce the APLs to half of the ROBDDs, we need to increase the memory sizes to 488% of the ROBDDs. The APLs of heterogeneous MDDs obtained by the memory size minimization algorithm (Algorithm 3.2) are as small as the APLs of MDD(3)s.

Finally, Table XIII compares the area–time complexities [4], [6], [39] of ROBDDs, heterogeneous MDDs, and $\mathrm{MDD}(k)$s for MCNC benchmark functions. The area–time complexity is the measure of computational cost considering both area and time. It is defined as

$$AT = \mathrm{area} \times \mathrm{time}.$$

In this section, the area A corresponds to the memory size and the time T corresponds to APL.

Table XIII shows that for these benchmark functions, area–time complexities of heterogeneous MDDs are half of the ROBDDs and are much smaller than $\mathrm{MDD}(k)$s.

V. CONCLUSION AND COMMENTS

This paper proposed minimization algorithms for the memory size and average path length (APL) of heterogeneous multivalued decision diagrams (MDDs) that consider both orderings and partitions of binary variables. The experimental results show that: 1) heterogeneous MDDs represent logic functions more compactly than reduced ordered binary decision diagrams (ROBDD s) and free BDDs. Especially, for all five-variable logic functions, the minimum heterogeneous MDDs require 67% of the memory sizes for the minimum ROBDDs, on average. Algorithm 3.1 can find exact minimum heterogeneous MDDs for functions with up to 12 inputs in a reasonable computation time, and Algorithm 3.2 can reduce the memory

TABLE IX
MEMORY SIZES OF ROBDDS, HETEROGENEOUS MDDS, AND MDD(k)S FOR n-VARIABLE LOGIC FUNCTIONS

n	ROBDD	Heterogeneous MDD		MDD(2)	MDD(3)	MDD(4)	MDD(5)	#samples
		MinMem	MinAPL$_M$					
4	1.00	0.72	0.86	0.96	1.22	0.94	1.83	2^{16}
5	1.00	0.67	0.91	0.93	1.24	1.47	0.99	2^{32}
6	1.00	0.68	0.80	0.92	1.10	1.50	2.02	1000
7	1.00	0.64	0.79	0.90	0.98	1.45	1.87	1000
8	1.00	0.58	0.81	0.85	0.98	1.49	1.79	1000
9	1.00	0.55	0.83	0.83	1.24	1.07	1.87	1000
10	1.00	0.54	0.84	0.81	0.82	0.96	2.01	1000

TABLE X
APLS OF ROBDDS, HETEROGENEOUS MDDS, AND MDD(k)S FOR n-VARIABLE LOGIC FUNCTIONS

n	ROBDD	Heterogeneous MDD		MDD(2)	MDD(3)	MDD(4)	MDD(5)	#samples
		MinMem	MinAPL$_M$					
4	1.00	0.52	0.37	0.59	0.47	0.34	0.34	2^{16}
5	1.00	0.39	0.27	0.60	0.47	0.36	0.25	2^{32}
6	1.00	0.45	0.32	0.59	0.43	0.43	0.33	1000
7	1.00	0.40	0.27	0.58	0.42	0.36	0.36	1000
8	1.00	0.33	0.22	0.55	0.41	0.30	0.31	1000
9	1.00	0.29	0.19	0.59	0.38	0.32	0.26	1000
10	1.00	0.26	0.17	0.53	0.43	0.31	0.23	1000

TABLE XI
MEMORY SIZES OF ROBDDS, HETEROGENEOUS MDDS, AND MDD(k)S FOR MCNC BENCHMARK FUNCTIONS

Name	In	Out	ROBDD	Heterogeneous MDD		MDD(2)	MDD(3)	MDD(4)	MDD(5)
				MinMem	MinAPL$_M$				
C432	36	7	3189	2824	3179	3075	4833	5508	12 441
C499	41	32	77 595	59 739	77 589	62 660	76 248	100 810	174 669
C880	60	26	12 156	11 812	12 154	15 610	21 933	32 742	53 526
C1908	33	25	16 575	13 493	16 564	16 415	18 720	30 039	36 135
C2670	233	64	5319	4649	5319	7600	12 483	19 584	36 102
C3540	50	22	71 481	65 029	71 480	84 315	127 809	194 650	307 197
C5315	178	123	5154	4582	5153	6725	9981	17 000	30 789
C7552	207	107	6633	6119	6633	8615	13 338	21 301	36 828
alu4	14	8	1047	855	1019	1290	1431	2295	3927
apex1	45	45	3735	3016	3728	4575	6138	8840	15 411
apex6	135	99	1491	1414	1490	2190	3924	7123	12 210
cps	24	102	2910	2533	2906	3000	4482	7786	11 748
dalu	75	16	2064	1548	2064	2610	3690	6749	10 263
des	256	245	8832	7288	8831	9630	16 299	21 488	41 712
frg2	143	139	2886	2671	2884	4395	7191	12 818	21 879
i3	132	6	396	330	396	340	603	646	1452
i8	133	81	3825	3662	3825	6035	9855	17 884	33 924
i10	257	224	61 977	55 766	61 974	85 535	124 065	234 260	380 655
k2	45	45	3735	3018	3728	4570	6165	8823	15 345
too_large	38	3	954	857	954	1090	1521	2465	3696
vda	17	39	1431	1088	1424	1375	2286	2788	4290
Average of Ratios			1.00	0.86	1.00	1.20	1.80	2.84	4.88

TABLE XII
APLS OF ROBDDS, HETEROGENEOUS MDDS, AND MDD(k)S FOR MCNC BENCHMARK FUNCTIONS

Name	In	Out	ROBDD	Heterogeneous MDD		MDD(2)	MDD(3)	MDD(4)	MDD(5)
				MinMem	MinAPL$_M$				
C432	36	7	86.58	55.74	45.45	59.84	48.58	40.92	35.52
C499	41	32	813.64	381.14	192.52	422.28	282.65	225.88	189.23
C880	60	26	135.79	125.73	99.13	115.52	104.51	95.67	86.27
C1908	33	25	254.35	145.81	92.09	168.38	118.34	103.37	86.56
C2670	233	64	214.05	167.90	133.78	189.35	179.05	154.69	152.89
C3540	50	22	209.15	141.10	91.78	160.52	132.50	109.62	93.03
C5315	178	123	462.05	373.23	304.38	400.05	378.00	342.61	339.44
C7552	207	107	484.03	424.85	314.03	418.23	380.40	336.50	309.67
alu4	14	8	40.81	24.41	19.59	31.57	21.66	19.85	15.58
apex1	45	45	180.59	87.35	67.63	154.81	124.86	94.26	95.72
apex6	135	99	291.54	260.66	231.06	268.39	271.51	263.92	247.19
cps	24	102	290.25	187.90	151.81	203.95	211.43	192.44	149.73
dalu	75	16	102.67	39.40	28.09	70.55	51.92	51.58	41.09
des	256	245	1210.00	910.63	687.50	931.14	838.53	749.61	729.31
frg2	143	139	624.69	499.27	348.60	584.58	531.48	512.31	501.66
i3	132	6	26.76	17.84	12.61	18.84	15.03	13.10	11.97
i8	133	81	302.54	229.12	207.54	292.75	248.84	243.43	238.52
i10	257	224	1084.96	887.62	614.53	950.62	821.89	903.42	788.22
k2	45	45	180.52	87.32	67.61	155.30	125.76	95.23	96.78
too_large	38	3	13.16	8.47	6.24	9.00	7.47	6.39	5.62
vda	17	39	176.34	81.72	69.54	111.91	110.19	76.03	74.33
Average of Ratios			1.00	0.67	0.51	0.78	0.68	0.60	0.55

TABLE XIII
AREA–TIME COMPLEXITIES OF ROBDDS, HETEROGENEOUS MDDS, AND MDD(k)S FOR MCNC BENCHMARK FUNCTIONS

Name	In	Out	ROBDD	Heterogeneous MDD		MDD(2)	MDD(3)	MDD(4)	MDD(5)
				MinMem	MinAPL$_M$				
C432	36	7	276 118	157 399	144 476	184 002	234 806	225 380	441 951
C499	41	32	63 134 444	22 768 960	14 937 095	26 460 143	21 551 378	22 771 246	33 053 379
C880	60	26	1 650 677	1 485 138	1 204 830	1 803 197	2 292 213	3 132 374	4 617 949
C1908	33	25	4 215 784	1 967 478	1 525 388	2 764 020	2 215 349	3 105 084	3 127 715
C2670	233	64	1 138 518	780 551	711 573	1 439 035	2 235 054	3 029 509	5 519 726
C3540	50	22	14 950 022	9 175 809	6 560 680	13 534 607	16 934 893	21 338 299	28 577 026
C5315	178	123	2 381 424	1 710 121	1 568 471	2 690 305	3 772 817	5 824 415	10 451 014
C7552	207	107	3 210 593	2 599 668	2 082 937	3 603 038	5 073 795	7 167 682	11 404 484
alu4	14	8	42 730	20 867	19 967	40 725	30 992	45 567	61 168
apex1	45	45	674 496	263 461	252 108	708 266	766 385	833 270	1 475 135
apex6	135	99	434 681	368 575	344 280	587 764	1 065 407	1 879 894	3 018 209
cps	24	102	844 642	475 959	441 173	611 840	947 620	1 498 369	1 758 999
dalu	75	16	211 905	60 990	57 976	184 138	191 568	348 115	421 705
des	256	245	10 686 720	6 636 635	6 071 313	8 966 884	13 667 221	16 107 606	30 421 083
frg2	143	139	1 802 850	1 333 541	1 005 363	2 569 241	3 821 905	6 566 790	10 975 884
i3	132	6	10 597	5887	4994	6405	9062	8465	17 384
i8	133	81	1 157 231	839 045	793 848	1 766 746	2 452 355	4 353 531	8 091 520
i10	257	224	67 242 627	49 499 023	38 084 982	81 311 523	101 968 393	211 634 736	300 040 993
k2	45	45	674 228	263 525	252 046	709 723	775 331	840 241	1 485 118
too_large	38	3	12 556	7258	5957	9805	11 362	15 745	20 762
vda	17	39	252 348	88 910	99 025	153 883	251 900	211 979	318 856
Average of Ratios			1.00	0.59	0.51	0.96	1.27	1.85	2.96

size of heterogeneous MDDs as fast as the sifting algorithm [28]; 2) in heterogeneous MDDs, APLs can be reduced by half of the corresponding ROBDDs, on average, without increasing the memory size. And both memory sizes and APLs can be reduced to 86% and 67% of ROBDDs, respectively. Algorithm 3.3, considering both partitions and orderings of binary variables, finds heterogeneous MDDs with the minimum APLs for functions with up to 11 variables within a reasonable time, and 3) in MDD(k)s, to reduce the APLs to half of the ROBDDs, we need to increase the memory sizes to 488% of the ROBDDs. Area–time complexities of heterogeneous MDDs are half of the ROBDDs and are much smaller than MDD(k)s.

APPENDIX

The following lemma is used in the proof of Theorem 2.1.

Lemma A.1 [18]: The number of different distributions of n objects into r distinct cells is calculated by the formula

$$a(n,r)=\sum_{i=0}^{r} {}_rC_i(r-i)^n(-1)^i$$

where each cell has at least one object and the order of objects within a cell is not important.

Theorem 2.1: Let $N_{\mathrm{nonfix}}(n)$ be the number of different nonfixed-order partitions of $X=(x_1,x_2,\ldots,x_n)$. Then

$$N_{\mathrm{nonfix}}(n)=\sum_{r=1}^{n} a(n,r)=\sum_{r=1}^{n}\sum_{i=0}^{r} {}_rC_i(r-i)^n(-1)^i.$$

Proof: From Lemma A.1, the number of different nonfixed-order partitions of n binary variables into r super variables is calculated by

$$a(n,r)=\sum_{i=0}^{r} {}_rC_i(r-i)^n(-1)^i.$$

Since $N_{\mathrm{nonfix}}(n)$ is the summation of $a(n,r)$ for $r=1,2,\ldots,n$, we have the theorem. ■

Theorem 2.3: Consider an ROBDD and a heterogeneous MDD for an n-variable logic function that is not a constant function. When an order of binary variables is fixed, for the memory sizes of ROBDD and heterogeneous MDD obtained by considering only the fixed-order partitions, the following relation holds:

$$\frac{\mathrm{Mem}(\text{heterogeneous MDD})}{\mathrm{Mem}(\mathrm{ROBDD})}>\frac{1}{3}.$$

Proof: From Theorem 2.2, we have

$$\begin{aligned}3\,\mathrm{Mem}(\text{heterogeneous MDD}) &\geq 3\,\mathrm{nodes}(\mathrm{ROBDD})+6\\ &=\mathrm{Mem}(\mathrm{ROBDD})+6.\end{aligned}$$

Therefore, we have

$$\frac{\mathrm{Mem}(\text{heterogeneous MDD})}{\mathrm{Mem}(\mathrm{ROBDD})}\geq\frac{1}{3}+\frac{2}{\mathrm{Mem}(\mathrm{ROBDD})}>\frac{1}{3}.$$

■

Theorem 2.4: Assume that the number of nodes in an ROBDD for an n-variable function f is the upper bound [17]

$$2^{n-r}+2^{2^r}-3$$

where r is the largest integer satisfying $n-r\geq 2^r$. Let $\mathrm{Mem}_{\min}(\mathrm{MDD})$ be the memory size of the minimum heterogeneous MDD for f. Let $s=2^r+r-n$, where $0\leq s\leq 2^r$. When n is large and $2\leq s\leq 2^r$, the following relation holds:

$$\frac{\mathrm{Mem}_{\min}(\mathrm{MDD})}{\mathrm{Mem}(\mathrm{ROBDD})}\simeq\frac{2^s+3}{3\,(2^s+1)}.$$

Proof: From (2.1) in Section II-F, we have

$$\mathrm{Mem}(\mathrm{ROBDD})=3\,\left(2^{n-r}+2^{2^r}-3\right).$$

The memory size of the minimum heterogeneous MDD for f is given by

$$\mathrm{Mem}_{\min}(\mathrm{MDD})=2^{n-r'}+3\,2^{2^{r'}}-5$$

where r' is the largest integer satisfying $n - r' \geq 2^{r'} + \log_2 3$ [27].

When $2 \leq s \leq 2^r$, the following relation holds:

$$\begin{aligned} n - r &= 2^r + s \\ 2^{n-r} &= 2^s \times 2^{2^r} \\ r' &= r. \end{aligned}$$

Then, when n is large, we have

$$\begin{aligned} \mathrm{Mem}(\mathrm{ROBDD}) &= 3\left(2^s \times 2^{2^r} + 2^{2^r} - 3\right) \\ &= 3\left(2^s + 1\right)2^{2^r} - 9 \\ &\simeq 3\left(2^s + 1\right)2^{2^r} \\ \mathrm{Mem}_{\min}(\mathrm{MDD}) &= 2^s \times 2^{2^r} + 3 \times 2^{2^{r'}} - 5 \\ &= (2^s + 3)2^{2^r} - 5 \\ &\simeq (2^s + 3)2^{2^r}. \end{aligned}$$

Therefore, we have the theorem. ■

Corollary 2.1: In Theorem 2.4, when n is sufficiently large and $s = 0$ or $9 \leq s \leq 2^r$, the following relation holds:

$$\frac{\mathrm{Mem}_{\min}(\mathrm{MDD})}{\mathrm{Mem}(\mathrm{ROBDD})} \simeq 0.33.$$

Proof:

1) When $s = 0$ (i.e., $n - r = 2^r$), $2^{n-r} = 2^{2^r}$ and $r' = r - 1$ hold. Then, we have

$$\begin{aligned} \mathrm{Mem}(\mathrm{ROBDD}) &= 3 \times \left(2^{2^r} + 2^{2^r} - 3\right) \\ &= 6 \times 2^{2^r} - 9 \\ \mathrm{Mem}_{\min}(\mathrm{MDD}) &= 2^{n-r'} + 3 \times 2^{2^{r'}} - 5 \\ &= 2 \times 2^{n-r} + 3\left(2^{2^r}\right)^{\frac{1}{2}} - 5 \\ &= 2 \times 2^{2^r} + 3\left(2^{2^r}\right)^{\frac{1}{2}} - 5. \end{aligned}$$

Let $b = (2^{2^r})^{1/2}$, then

$$\begin{aligned} \mathrm{Mem}(\mathrm{ROBDD}) &= 6b^2 - 9 \simeq 6b^2 \\ \mathrm{Mem}_{\min}(\mathrm{MDD}) &= 2b^2 + 3b - 5 \simeq b(2b + 3). \end{aligned}$$

Therefore, when n is large, the following relation holds:

$$\frac{\mathrm{Mem}_{\min}(\mathrm{MDD})}{\mathrm{Mem}(\mathrm{ROBDD})} \simeq \frac{b(2b+3)}{6b^2} = \frac{1}{3} + \frac{1}{2b} \simeq 0.33.$$

2) From Theorem 2.4, when $9 \leq s \leq 2^r$, the corollary holds. ■

The following lemma is used for proof of Theorem 2.5.

Lemma A.2: Let r and r' be the largest integers satisfying

$$\begin{cases} n - r \geq 2^r \\ n - r' \geq 2^{r'} + r' \end{cases}$$

where n is a nonzero positive integer. Then, for r and r', the following relation holds:

$$r - 1 \leq r' \leq r.$$

Proof: From $n - r \geq 2^r$, we have

$$n = 2^r + r + a$$

where a is an integer satisfying $0 \leq a \leq 2^r$.

Let $r' = r + b$, where b is an integer. Then, from $n - r' \geq 2^{r'} + r'$, we have

$$\begin{aligned} n \geq 2^{r'} + 2r' &= 2^{r+b} + 2(r+b) \\ 2^r + r + a &\geq 2^{r+b} + 2(r+b) \\ (1 - 2^b)2^r - r - 2b + a &\geq 0. \end{aligned} \tag{A.1}$$

1) When $b > 0$, (A.1) does not hold.
2) When $b = 0$, (A.1) holds for $r \leq a \leq 2^r$.
3) When $b < 0$, (A.1) holds for $0 \leq a \leq 2^r$.

Since r' is the largest integer satisfying $n - r' \geq 2^{r'} + r'$, b must be the largest integer satisfying (A.1). Therefore, we have the following relation:

$$\begin{cases} r' = r \text{ (i.e., } b = 0), & \text{when } r \leq a \leq 2^r \\ r' = r - 1 \text{ (i.e., } b = -1), & \text{when } 0 \leq a \leq r \end{cases}.$$

■

Theorem 2.5: When the number of nodes in an ROBDD for an n-variable function f is the upper bound [17]

$$2^{n-r} + 2^{2^r} - 3$$

where r is the largest integer satisfying $n - r \geq 2^r$, there exists a heterogeneous MDD for f that satisfies

$$\begin{aligned} &\mathrm{APL}(\text{heterogeneous MDD}) \leq 2.0 \\ &\mathrm{Mem}(\text{heterogeneous MDD}) \leq \mathrm{Mem}(\mathrm{ROBDD}). \end{aligned}$$

Proof: When a heterogeneous MDD is represented by two super variables: $X_1 = (x_1, x_2, \ldots, x_{n-r'})$ and $X_2 = (x_{n-r'+1}, x_{n-r'+2}, \ldots, x_n)$

$$\mathrm{APL}(\text{heterogeneous MDD}) \leq 2.0$$

where for simplicity, we assume that the variable order is $(x_1, x_2, \ldots, x_n)$. In this case, the memory size of the heterogeneous MDD is given by

$$2^{n-r'} + 1 + \left(2^{r'} + 1\right)\left(2^{2^{r'}} - 2\right)$$

where r' is the largest integer satisfying $n - r' \geq 2^{r'} + r'$. From here, we will prove that this memory size is smaller than or equal to that of ROBDD

$$2^{n-r'} + 1 + \left(2^{r'} + 1\right)\left(2^{2^{r'}} - 2\right) \leq 3\left(2^{n-r} + 2^{2^r} - 3\right).$$

By Lemma A.2, we have to only consider two cases: $r' = r$ and $r' = r - 1$.

1) When $r' = r$, we have

$$\mathrm{Mem}(\mathrm{ROBDD}) = 3\left(2^{n-r} + 2^{2^r} - 3\right)$$

$$\mathrm{Mem}(\text{heterogeneous MDD}) = 2^{n-r} + 1 + (2^r + 1)\left(2^{2^r} - 2\right)$$

$$\mathrm{Mem}(\mathrm{ROBDD}) - \mathrm{Mem}(\text{heterogeneous MDD}) = 2\left(2^{n-r} + 2^{2^r} + 2^r\right) - 2^{2^r+r} - 8.$$

From $n - r \geq 2^r + r$, we have

$$\begin{aligned} &2\left(2^{n-r} + 2^{2^r} + 2^r\right) - 2^{2^r+r} - 8 \\ &\quad \geq 2\left(2^{n-r} + 2^{2^r} + 2^r\right) - 2^{n-r} - 8 \\ &\quad = 2^{n-r} + 2\left(2^{2^r} + 2^r\right) - 8. \end{aligned}$$

Since $n \geq 1$ and $r \geq 0$, we have

$$2^{n-r} + 2\left(2^{2^r} + 2^r\right) - 8 \geq 0.$$

2) When $r' = r - 1$, we have

$$\mathrm{Mem}(\mathrm{ROBDD}) = 3\left(2^{n-r'-1} + 2^{2^{r'+1}} - 3\right)$$

$$\mathrm{Mem}(\text{heterogeneous MDD}) = 2^{n-r'} + 1 + \left(2^{r'} + 1\right)\left(2^{2^{r'}} - 2\right)$$

$$\mathrm{Mem}(\mathrm{ROBDD}) - \mathrm{Mem}(\text{heterogeneous MDD}) = 2^{n-r'-1} + 3 \times 2^{2^{r'+1}} - \left(2^{r'} + 1\right)2^{2^{r'}} + 2^{r'+1} - 8.$$

From $n - r \geq 2^r$ and $r = r' + 1$, we have $n - r' - 1 \geq 2^{r'+1}$. Then

$$\begin{aligned} &2^{n-r'-1} + 3 \times 2^{2^{r'+1}} - \left(2^{r'} + 1\right)2^{2^{r'}} + 2^{r'+1} - 8 \\ &\quad \geq 2^{2^{r'+1}} + 3 \times 2^{2^{r'+1}} - \left(2^{r'} + 1\right)2^{2^{r'}} + 2^{r'+1} - 8 \\ &\quad = 2^{2^{r'}}\left(4 \times 2^{2^{r'}} - 2^{r'} - 1\right) + 2^{r'+1} - 8. \end{aligned}$$

Since $r' \geq 0$ and $(4 \times 2^{2^{r'}} - 2^{r'} - 1) \geq 6$, we have

$$2^{2^{r'}}\left(4 \times 2^{2^{r'}} - 2^{r'} - 1\right) + 2^{r'+1} - 8 \geq 6.$$

Therefore, the theorem holds. ■

REFERENCES

[1] P. Ashar and S. Malik, "Fast functional simulation using branching programs," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, CA, Nov. 1995, pp. 408–412.

[2] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, A. Sangiovanni-Vincentelli, E. M. Sentovich, and K. Suzuki, "Synthesis of software programs for embedded control applications," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 18, no. 6, pp. 834–849, Jun. 1999.

[3] K. Brace, R. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," in *Proc. Design Automation Conf.*, Orlando, FL, Jun. 1990, pp. 40–45.

[4] R. P. Brent and H. T. Kung, "The area–time complexity of binary multiplication," *J. ACM*, vol. 28, no. 3, pp. 521–534, Jul. 1981.

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

[6] ——, "On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication," *IEEE Trans. Comput.*, vol. 40, no. 2, pp. 205–213, Feb. 1991.

[7] R. Drechsler, W. Günther, and F. Somenzi, "Using lower bounds during dynamic BDD minimization," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 1, pp. 51–57, Jan. 2001.

[8] R. Ebendt, W. Günther, and R. Drechsler, "Minimization of the expected path length in BDDs based on local changes," in *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)*, Yokohama, Japan, Jan. 2004, pp. 866–871.

[9] M. Fujita, Y. Matsunaga, and T. Kakuda, "On variable ordering of binary decision diagrams for the application of multi-level logic synthesis," in *Proc. European Design Automation Conf. (EDAC)*, Amsterdam, The Netherlands, Mar. 1991, pp. 50–54.

[10] W. Günther and R. Drechsler, "Minimization of free BDDs," in *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)*, Wanchai, Hong Kong, Jan. 1999, pp. 323–326.

[11] W. Günther, "Minimization of free BDDs using evolutionary techniques," in *Proc. Int. Workshop Logic Synthesis (IWLS)*, Loguna Cliffs Marriott, Dana Point, CA, May 2000, pp. 167–172.

[12] H. M. Hasan Babu and T. Sasao, "Heuristics to minimize multiple-valued decision diagrams," *IEICE Trans. Fundam.*, vol. E83-A, no. 12, pp. 2498–2504, Dec. 2000.

[13] Y. Iguchi, T. Sasao, M. Matsuura, and A. Iseno, "A hardware simulation engine based on decision diagrams," in *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)*, Yokohama, Japan, Jan. 2000, pp. 73–76.

[14] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of binary decision diagrams based on exchanges of variables," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, Santa Clara, CA, Nov. 1991, pp. 472–475.

[15] Y. Jiang and B. K. Brayton, "Software synthesis from synchronous specifications using logic simulation techniques," in *Proc. Design Automation Conf.*, New Orleans, LA, Jun. 2002, pp. 319–324.

[16] T. Kam, T. Villa, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Multi-valued decision diagrams: Theory and applications," *Int. J. Mult.-Valued Logic*, vol. 4, no. 1–2, pp. 9–62, 1998.

[17] H.-T. Liaw and C.-S. Lin, "On the OBDD-representation of general Boolean function," *IEEE Trans. Comput.*, vol. 4, no. 6, pp. 661–664, Jun. 1992.

[18] C. L. Liu, *Introduction to Combinatorial Mathematics*. New York: McGraw-Hill, 1968.

[19] P. C. McGeer, K. L. McMillan, A. Saldanha, A. L. Sangiovanni-Vincentelli, and P. Scaglia, "Fast discrete function evaluation using decision diagrams," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, San Jose, CA, Nov. 1995, pp. 402–407.

[20] D. M. Miller and R. Drechsler, "Implementing a multiple-valued decision diagram package," in *Proc. 28th Int. Symp. Multiple-Valued Logic*, Fukuoka, Japan, May 1998, pp. 52–57.

[21] ——, "Augmented sifting of multiple-valued decision diagrams," in *Proc. 33rd Int. Symp. Multiple-Valued Logic*, Tokyo, Japan, May 2003, pp. 375–382.

[22] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation," in *Proc. Design Automation Conf.*, Orlando, FL, Jun. 1990, pp. 52–57.

[23] S. Minato, "Minimum-width method of variable ordering for binary decision diagrams," *IEICE Trans. Fundam.*, vol. E75-A, no. 3, pp. 392–399, Mar. 1992.

[24] S. Muroga, *Logic Design and Switching Theory*. New York: Wiley-Interscience, 1979.

[25] S. Nagayama, T. Sasao, Y. Iguchi, and M. Matsuura, "Representations of logic functions using QRMDDs," in *Proc. 32nd Int. Symp. Multiple-Valued Logic*, Boston, MA, May 2002, pp. 261–267.

[26] S. Nagayama and T. Sasao, "Code generation for embedded systems using heterogeneous MDDs," in *Proc. 12th Workshop Synthesis and System Integration Mixed Information Technologies (SASIMI)*, Hiroshima, Japan, Apr. 2003, pp. 258–264.

[27] ——, "Compact representations of logic functions using heterogeneous MDDs," in *Proc. 33rd Int. Symp. Multiple-Valued Logic*, Tokyo, Japan, May 2003, pp. 247–252.

[28] S. Nagayama, A. Mishchenko, T. Sasao, and J. T. Butler, "Minimization of average path length in BDDs by variable reordering," in *Proc. Int. Workshop Logic and Synthesis*, Laguna Beach, CA, May 2003, pp. 207–213.
[29] S. Nagayama and T. Sasao, "Compact representations of logic functions using heterogeneous MDDs," *IEICE Trans. Fundam.*, vol. E86-A, no. 12, pp. 3168–3175, Dec. 2003.
[30] ——, "Minimization of memory size for heterogeneous MDDs," in *Proc. Asia and South Pacific Design Automation Conf. (ASP-DAC)*, Yokohama, Japan, Jan. 2004, pp. 872–875.
[31] ——, "On the minimization of average path lengths for heterogeneous MDDs," in *Proc. 34th Int. Symp. Multiple-Valued Logic*, Toronto, Canada, May 2004, pp. 216–222.
[32] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," in *Proc. Int. Conf. Computer-Aided Design (ICCAD)*, Santa Clara, CA, Nov. 1993, pp. 42–47.
[33] T. Sasao and J. T. Butler, "A method to represent multiple-output switching functions by using multi-valued decision diagrams," in *Proc. 26th Int. Symp. Multiple-Valued Logic*, Santiago de Compostela, Spain, May 1996, pp. 248–254.
[34] T. Sasao, "Ternary decision diagrams survey," in *Proc. 27th Int. Symp. Multiple-Valued Logic*, Antigonish, NS, Canada, May 1997, pp. 241–250.
[35] ——, *Switching Theory for Logic Synthesis*. Norwell, MA: Kluwer, 1999.
[36] F. Schmiedle, W. Günther, and R. Drechsler, "Dynamic re-encoding during MDD minimization," in *Proc. 30th Int. Symp. Multiple-Valued Logic*, Portland, OR, May 2000, pp. 239–244.
[37] F. Somenzi, *CUDD: CU Decision Diagram Package Release 2.3.1*. Boulder, CO: Univ. Colorado, 2001.
[38] K. Takagi, H. Hatakeda, S. Kimura, and K. Watanabe, "Exact minimization of free BDDs and its application to pass-transistor logic optimization," *IEICE Trans. Fundam.*, vol. E82-A, no. 11, pp. 2407–2413, Nov. 1999.
[39] C. D. Thompson, "Area–time complexity for VLSI," in *Proc. Ann. Symp. Theory Computing*, Atlanta, GA, May 1979, pp. 81–88.

Shinobu Nagayama (S'02–M'04) received the B.S. and M.E. degrees from the Meiji University, Kanagawa, Japan, in 2000 and 2002, respectively, and the Ph.D. degree in computer science from the Kyushu Institute of Technology, Iizuka, Japan, in 2004.

He is now a Postdoctoral Researcher in Kyushu Institute of Technology. His research interest includes decision diagrams, software synthesis, and embedded systems.

Tsutomu Sasao (S'72–M'77–SM'90–F'94) received the B.E., M.E., and Ph.D. degrees in electronics engineering from the Osaka University, Osaka, Japan, in 1972, 1974, and 1977, respectively.

He has held faculty/research positions at Osaka University, the IBM T.J. Watson Research Center, Yorktown Heights, NY, and the Naval Postgraduate School, Monterey, CA. He is currently a Professor at the Department of Computer Science and Electronics, 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 more than nine books on logic design, including *Logic Synthesis and Optimization*, *Representation of Discrete Functions*, *Switching Theory for Logic Synthesis*, and *Logic Synthesis and Verification* (Kluwer Academic, 1993, 1996, 1999, and 2001) respectively.

Dr. Sasao has served as Program Chairman for the IEEE International Symposium on Multiple-Valued Logic (ISMVL) many times. Also, he was the Symposium Chairman of the 28th ISMVL held in Fukuoka, Japan, in 1998. He received the National Institute of Water and Atmospheric Research (NIWA) Memorial Award in 1979, Distinctive Contribution Awards from the IEEE Computer Society Technical Committee on Multiple-Valued Logic (MVL-TC) in 1987, 1996, and 2004 for papers presented at ISMVLs, and the Takeda Techno-Entrepreneurship Award in 2001. He has served as an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS.