PAPER *Special Section on VLSI Design and CAD Algorithms*

Compact Representations of Logic Functions Using Heterogeneous MDDs

Shinobu NAGAYAMA[†a)], ***Student Member and*** **Tsutomu SASAO**[†,††b)], ***Regular Member***

SUMMARY In this paper, we propose a compact representation of logic functions using Multi-valued Decision Diagrams (MDDs) called heterogeneous MDDs. In a heterogeneous MDD, each variable may take a different domain. By partitioning binary input variables and representing each partition as a single multi-valued variable, we can produce a heterogeneous MDD with 16% smaller memory size than a Reduced Ordered Binary Decision Diagram (ROBDD), and with comparable memory size to Free Binary Decision Diagrams (FBDDs). And also, heterogeneous MDDs have shorter Average Path Length (APL) than ROBDDs and FBDDs. We minimized a large number of benchmark functions to show the compactness of heterogeneous MDDs.

key words: *heterogeneous MDD, ROBDD, FBDD, APL, memory size*

1. Introduction

Binary Decision Diagrams (BDDs) and Multi-valued Decision Diagrams (MDDs) are extensively used in logic synthesis [7], logic simulation [1], software synthesis [2], [14], [16], [22], etc. Since the memory sizes for these applications depend on the sizes of BDDs and the MDDs, the minimizations for BDDs and MDDs are important. Most of the minimization algorithms for BDDs and MDDs use the variable reordering approach [6], [7], [10], [13], [25]. In this paper, we propose a method to minimize MDDs using a different approach, that is the partition of binary variables.

To represent a binary logic function using an MDD, binary variables are partitioned into groups. In many cases, the groups have the same number of binary variables. In a heterogeneous MDD that is proposed in this paper, the groups can have different numbers of binary variables. The memory sizes of heterogeneous MDDs can be minimized by considering the partition of binary variables. In this paper, to show the effectiveness of this partition approach, the binary variable order is fixed.

The rest of the paper is organized as follows. Section 2 defines heterogeneous MDDs and a method to represent multiple-output functions. Section 3 considers the memory size and the Average Path Length (APL) for heterogeneous MDDs. Section 4 proposes a memory minimization algorithm and an APL minimization algorithm. And, Sect. 5 compares sizes and APLs of heterogeneous MDDs for many benchmark functions.

Manuscript received March 10, 2003.
Manuscript revised June 10, 2003.
Final manuscript received August 8, 2003.

[†]The authors are with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka-shi, 820-8502 Japan.
[††]The author is with Center for Microelectronics Systems, Kyushu Institute of Technology, Iizuka-shi, 820-8502 Japan.
a) E-mail: nagayama@aries02.cse.kyutech.ac.jp
b) E-mail: sasao@cse.kyutech.ac.jp

2. Definitions

This section defines heterogeneous MDDs, and shows a method to represent multiple-output functions.

2.1 Partitions of Binary Variables

Definition 2.1: Let $f(X)$ be a two-valued logic function, where $X = (x_1, x_2, \ldots, x_n)$ and $x_i (i = 1, 2, \ldots, n)$ are binary variables.

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

We assume that the function is completely specified and includes no redundant variables.

Example 2.1: Consider (X_1, X_2), which is a partition of X, where $X = (x_1, x_2, x_3, x_4, x_5)$ and each x_i is a binary variable. When $X_1 = (x_1, x_2)$ and $X_2 = (x_3, x_4, x_5)$, $k_1 = 2$, $k_2 = 3$, $P_1 = \{0, 1, 2, 3\}$, and $P_2 = \{0, 1, \ldots, 7\}$. Note that X_1 takes 4 values, and X_2 takes 8 values. So, a 5-variable logic function $f(X)$ can be represented by the multi-valued function $f(X_1, X_2)$: $P_1 \times P_2 \rightarrow B$.
(End of Example)

2.2 Heterogeneous MDD

We assume that readers are familiar with BDDs, Reduced Ordered Binary Decision Diagrams (ROBDDs or OBDDs) [5], Free Binary Decision Diagrams (FBDDs) [8], [9], MDDs, and Reduced Ordered Multi-valued Decision Diagrams (ROMDDs) [15].

Definition 2.2: When $X = (x_1, x_2, \ldots, x_n)$ is partitioned into $(X_1, X_2, \ldots, X_u)$, an ROMDD representing a logic function $f(X)$ is called a **heterogeneous MDD**. Specifically, when $k_1 = k_2 = \ldots = k_u$, an ROMDD representing a logic function $f(X)$ is called **homogeneous**

MDD in order to distinguish from a heterogeneous MDD. A homogeneous MDD is denoted by **MDD(k)**, where $k = k_1 = k_2 = \ldots = k_u$. An MDD($k$) represents a mapping $f : P^u \to B$, while a heterogeneous MDD represents a mapping $f : P_1 \times P_2 \times \ldots \times P_u \to B$, where $P = \{0, 1, \ldots, 2^k - 1\}$, $P_i = \{0, 1, \ldots, 2^{k_i} - 1\}$, and $B = \{0, 1\}$.

In an MDD(k), non-terminal nodes have 2^k edges. When $k = 1$, an MDD(1) is an ROBDD. In a heterogeneous MDD, non-terminal nodes representing a super variable X_i have 2^{k_i} edges, where k_i denotes the number of binary variables in X_i.

Definition 2.3: In a decision diagram (DD), the **number of nodes in the DD**, denoted by $nodes(\text{DD})$, includes only non-terminal nodes.

Definition 2.4: The **width of the DD with respect to** X_i, denoted by $width(\text{DD}, i)$, is the number of nodes in the DD corresponding to the super variable X_i.

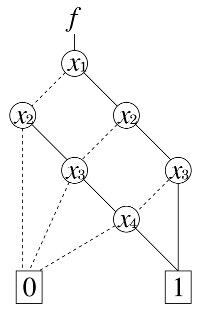
The number of nodes in the MDD with the partition $(X_1, X_2, \ldots, X_u)$ is given by

$$nodes(\text{MDD}) = \sum_{i=1}^{u} width(\text{MDD}, i).$$

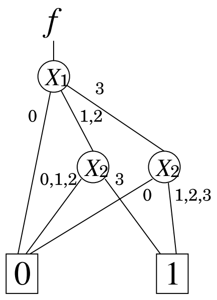
Example 2.2: Consider the function:

$$f = x_1x_2x_3 \vee x_2x_3x_4 \vee x_3x_4x_1 \vee x_4x_1x_2.$$

Figure 1(a), Fig. 1(b) and Fig. 2 represent the ROBDD, the MDD(2), and the 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 input 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. 2(a), $X_1 = (x_1, x_2, x_3)$ and $X_2 = (x_4)$. However, in Fig. 2(b), $X_1 = (x_1), X_2 = (x_2, x_3, x_4)$. (End of Example)

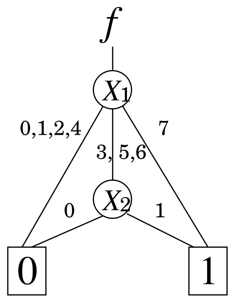


(a) BDD.

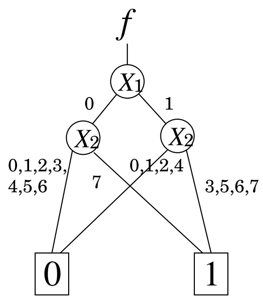


(b) MDD(2).

Fig. 1 BDD and MDD(2).



(a) Heterogeneous MDD with minimum memory requirement.



(b) Heterogeneous MDD with maximum memory requirement.

Fig. 2 Heterogeneous MDDs.

2.3 Representations of Multiple-Output Functions

Logic networks usually have many outputs. In most cases, independent representation of each output is inefficient. Let the multiple-output functions be $F = (f_0, f_1, \ldots, f_{m-1})$: $B^n \to B^m$, where $B = \{1, 0\}$, and n and m denote the number of input and output variables, respectively. Several methods exist to represent multiple-output functions by using BDDs [19], [26]–[28]. In this paper, we use Shared Binary Decision Diagrams (SBDDs) [19] to represent multiple-output functions. In the following, a BDD means an SBDD unless stated otherwise.

3. Measures for Heterogeneous MDDs

Lemma 3.1: When the order of the input variables $X = (x_1, x_2, \ldots, x_n)$ is fixed, the number of different partitions of X is 2^{n-1}.

(Proof) $X = (x_1, x_2, \ldots, x_n)$ has $n - 1$ **partition points**, the positions that can be partitioned. At each partition point, we can choose whether to partition at this point or not. Thus, 2^{n-1} different partitions exist. (Q.E.D.)

Therefore, when the order of the input variables is fixed, the number of different heterogeneous MDDs to consider is 2^{n-1}. From these heterogeneous MDDs, we can find an optimal heterogeneous MDD based on some criteria. This section shows the memory size and the Average Path Length (APL) for heterogeneous MDDs.

3.1 Memory Size

Definition 3.1: The **memory size of a DD** is the number of words needed to represent the DD in memory.

Definition 3.2: Given a logic function f, the **minimum heterogeneous MDD** for the logic function f is the heterogeneous MDD with the minimum memory size over all ordered partitions of the variables.

In memory, a node in a Reduced Ordered Decision Diagram (RODD) requires an index and a set of pointers that refer the succeeding nodes. Since a node in a BDD has two pointers, the memory size needed to represent a BDD is given by

$$(2 + 1) \times nodes(\text{BDD}), \tag{1}$$

where we assume that the size of a word is large enough to store a pointer to a node. Since a node in an MDD(k) has 2^k pointers, the memory size needed to represent an MDD(k)

is given by

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

As for a heterogeneous MDD, the memory size needed to represent it is given by

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

Example 3.1: The memory sizes to represent various DDs are as follows: for the BDD in Fig. 1(a), 18; for the MDD(2) in Fig. 1(b), 15; for the heterogeneous MDD in Fig. 2(a), 12; and for the heterogeneous MDD in Fig. 2(b), 21. (End of Example)

Theorem 3.1: In a minimum heterogeneous MDD, the following relation holds for any super variable $X_i = (x_j, x_{j+1}, \ldots, x_{j+k_i-1})$:

$$(2^{k_i} + 1) width(\text{heterogeneous MDD}, i) \le 3 \times \sum_{t=0}^{k_i-1} width(\text{BDD}, j+t),$$

where the heterogeneous MDD and the BDD represent the same logic function.

(Proof) In a minimum heterogeneous MDD, partition X_i into $(X_{i_0}, X_{i_1}, \ldots, X_{i_{k_i-1}})$, where $X_{i_0} = (x_j)$, $X_{i_1} = (x_{j+1})$,..., and $X_{i_{k_i-1}} = (x_{j+k_i-1})$. The memory size for the heterogeneous MDD with respect to X_{i_t} $(t = 0, 1, \ldots, k_i - 1)$ becomes

$$\sum_{t=0}^{k_i-1} (2 + 1) width(\text{heterogeneous MDD}, i_t) = 3 \times \sum_{t=0}^{k_i-1} width(\text{BDD}, j+t).$$

Note that each node in the BDD requires three words (see the formula (1)). If the theorem does not hold, then the original heterogeneous MDD was not minimum, which is contradiction. (Q.E.D.)

Theorem 3.2: Let $M_{max}(n)$ be the memory size needed to represent the minimum heterogeneous MDD for an n-variable logic function. Then, the following relation holds:

$$M_{max}(n) \le 2^{n-r} + 3 \cdot 2^{2^r} - 5,$$

where r is the largest integer satisfying the relation

$$n - r \ge 2^r + \log_2 3.$$

(Proof) See Appendix.

3.2 Average Path Length (APL)

Definition 3.3: A **path in a DD** is a sequence of nodes for some assignment of values to all variables. The **path length** is the number of non-terminal nodes on the path.

Definition 3.4: In a DD, the **node traversing probability**, denoted by $prob(\text{DD}, i)$, is a fraction of all assignments of values to variables whose path includes v_i.

In an MDD, we assume the following computation model:

1. DDs for logic functions are evaluated by traversing nodes from the root node to a terminal node according to values of the input variables.
2. MDDs are implemented directly, not simulated using the BDD package as described in [15].
3. 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 available as an input to the algorithm.
4. Most computation time is spent for accessing nodes.
5. The access time to all MDD nodes are equal.

In this case, the time to evaluate a DD for a logic function is proportional to the number of non-terminal nodes on the path (i.e., path length). And also, we assume that each binary variable occurs as a 0 with the same probability as a 1. Under these assumptions, we use the **Average Path Length** (APL) to estimate the evaluation time of different types of DDs.

In this paper, we use a Shared Decision Diagram (SDD) to represent multiple-output functions $F = (f_0, f_1, \ldots, f_{m-1})$. The APL of an SDD is the sum of the APLs of individual DDs for each function f_i [29].

Example 3.2: The APLs for different DDs are as follows: For the BDD in Fig. 1(a), 3.125; for the MDD(2) in Fig. 1(b), 1.75; for the heterogeneous MDD in Fig. 2(a), 1.375; and for the heterogeneous MDD in Fig. 2(b), 2.0. (End of Example)

Theorem 3.3: [29] The APL of a DD is given by the sum of the node traversing probabilities of all the non-terminal nodes.

Theorem 3.4: For a BDD and a heterogeneous MDD that represent the same logic function, the following relation holds:

$$(\text{APL of a heterogeneous MDD}) \le (\text{APL of a BDD}).$$

(Proof) The number of non-terminal nodes on the path never increases by grouping of variables in X. Therefore, we have the theorem. (Q.E.D.)

4. Optimizations of Heterogeneous MDDs

In this section, we formulate optimization problems of heterogeneous MDDs, and present optimization algorithms. We need a different optimization algorithm for each measure because the optimal solution based on a measure is not always the optimal for another measure.

4.1 Minimization of Memory Size

When the order of the input variables X is fixed, the memory size needed to represent a heterogeneous MDD depends on the partition of the input variables X. Therefore, we will find the partition of X that makes the required memory minimum.

Example 4.1: Figure 2(a) shows the heterogeneous MDD with the minimum memory for the function f, while Fig. 2(b) shows the heterogeneous MDD with the maximum memory for the function f. (End of Example)

We can formulate the memory minimization problem as follows:

Problem 4.1: Let the variable order for the BDD be fixed. Given a BDD for the logic function f, find a partition of X that produces the minimum heterogeneous MDD.

A naive method to obtain an optimum heterogeneous MDD for an n-variable logic function is to construct 2^{n-1} different heterogeneous MDDs and then to select an optimum one. When n is small, we can obtain an optimum solution within a reasonable time. For the functions with many input variables, we propose Algorithm 4.1, which shows a pseudo-code to solve Problem 4.1. This algorithm uses dynamic programing. All sub-solutions are stored in the table. For simplicity, we assume that the variable order is $x_1, x_2, \ldots, x_n$.

Algorithm 4.1: (Minimization of memory size)

```
1: minimize_memory (void) {
2:   table[$n$] = $(2+1)width(\text{BDD}, n)$ ;
3:   for($i = n-1; i \geq 1; i--$) {
4:     min_mem = (memory for BDD) ;
5:     for($l = 0; l \leq n-i; l++$) {
6:       $k$ = **branch[$i$][$l$]** ;
7:       mdd_mem = $(2^k+1)width(\text{heterogeneous MDD}, j)$ ;
8:       if (mdd_mem > **upper bound**)
9:         break ;
10:      next index $i' = i + k$ ;
11:      mdd_mem += table[$i'$] ;
12:      if (min_mem > mdd_mem) {
13:        min_mem = mdd_mem ;
14:        **register the partition k** ;
15:      }
16:    }
17:    table[$i$] = min_mem ;
18:  }
19:  return table[1] ;
20:}
```

Algorithm 4.1 finds an exact solution for Problem 4.1. **table[i]** in Algorithm 4.1 stores the minimum memory size for sub-graph from x_i to x_n. In the 6th line in Algorithm 4.1, **branch[i][l]** stores an integer k that makes the following ratio the l-th smallest,

$$ratio = \frac{(2^k+1)width(\text{heterogeneous MDD}, j)}{3 \times \sum_{t=0}^{k-1} width(\text{BDD}, i+t)},$$

where j is the index of corresponding super variable X_j. And, the 8th line uses **upper bound**, which is obtained by Theorem 3.1. The j in the 7th line denotes the index of corresponding super variable X_j.

Let n and N be the numbers of binary variables and nodes for the BDD, respectively. Algorithm 4.1 examines at most $\frac{n^2}{2}$ candidates, and calculates the value of $(2^k+1)width(\text{heterogeneous MDD}, j)$ per the examination. The time complexity to calculate the value of $(2^k+1)width(\text{heterogeneous MDD}, j)$ is $O(N)$. Therefore, the time complexity for Algorithm 4.1 is $O(n^2N)$. The space complexity for Algorithm 4.1 is $O(N)$.

4.2 Minimization of APL

When the order of the input variables X is fixed, the APL for a heterogeneous MDD also depends on a partition of the input variables X. The partition of X that minimizes the APL is the trivial partition, $X = X_1$, where $k_1 = n$. However, the memory size needed to represent the heterogeneous MDD for the trivial partition is $(2^n+1) \times width(\text{heterogeneous MDD}, 1)$, and is too large in most cases. Therefore, we find a partition of X that minimizes the APL within a given memory size. We formulate the APL minimization problem as follows:

Problem 4.2: Suppose that the variable order for the BDD is fixed. Given a BDD for the logic function f and a memory size L, find a partition of X that makes the APL of the heterogeneous MDD minimum within the memory size L.

In Algorithm 4.2, we show a pseudo-code to solve Problem 4.2. This algorithm uses a branch-and-bound method and a cache to reduce computation time. The sub-solutions are stored in the cache, but only a subset of sub-solutions is kept in it because the number of sub-solutions is too large in many cases. In other words, this algorithm is similar to the dynamic programing, except for that the cache is overwritten. In the case of cache miss, the sub-solution is searched again. For simplicity, we assume that the variable order is $x_1, x_2, \ldots, x_n$. This algorithm uses the index of the top variable for the BDD and the memory size L as the arguments.

Algorithm 4.2: (Minimization of APL)

```
1: minimize_APL (index $i$, mem_size $l$) {
2:   if ($i > n$)
3:     return 0 ;
4:   **check the cache** ;
5:   if (cache.index == $i$ && cache.mem == $l$) {
6:     **register the partition cache.k** ;
7:     return cache.APL ;
8:   }
9:   min_APL = (APL for BDD) ;
10:  for ($k = n-i+1; k \geq 1; k--$) {
11:    memory = $(2^k+1)\ width(\text{heterogeneous MDD}, j)$ ;
12:    next_index $i' = i + k$ ;
13:    if (($l -$ memory) < **lower_bound[$i'$]**)
14:      continue ;
15:    current_APL = 0 ;
16:     for (all nodes $v$ representing $X_j$)
17:         current_APL += $prob$(heterogeneous MDD, $v$) ;
18:     current_APL += minimize_APL ($i'$, $l$ − memory) ;
19:     if (current_APL < min_APL) {
20:         register the partition k ;
21:         min_APL = current_APL ;
22:     }
23: }
24: store (overwrite) to the cache ;
25: return min_APL ;
26:}
```

This algorithm produces an exact solution for Problem 4.2 by calculating the APLs for different partitions of X. The calculation of the APL uses Theorem 3.3. To compute the node traversing probability $prob(\text{heterogeneous MDD}, v)$ of the 17th line, we used the method in [29]. The 13th line uses **lower bounds** on the memory size obtained by Algorithm 4.1 to reduce computation time.

Let n, N, and C be the number of binary variables, the number of nodes for the BDD, and the cache size, respectively. Algorithm 4.2 examines at most 2^{n-1} candidates by exhaustive search. The time complexities for the calculations of **lower bounds** and the value of $prob(\text{heterogeneous MDD}, v)$ are $O(n^2N)$ and $O(N)$, respectively. Note that these values are calculated before the exhaustive search and stored in tables. Therefore, the time complexity for Algorithm 4.2 is $O(2^n + n^2N)$. The space complexity for Algorithm 4.2 is $O(N+C) = O(N)$, where C is considered as a constant value.

5. Experimental Results

5.1 Comparison with OBDDs and FBDDs

Table 1 compares the memory size and APL for heterogeneous MDDs with those for OBDDs and FBDDs. The OBDDs are obtained by the best known variable orders [30]. The data for FBDDs were provided by Dr. W. Günther [8], [9]. In Table 1, "MDD1" denotes the minimum heterogeneous MDDs obtained by Algorithm 4.1, where Algorithm 4.1 used the OBDDs [30] as the initial ones. Similarly, "MDD2" denotes heterogeneous MDDs with the minimum APL obtained by Algorithm 4.2, where the memory limitations L used the memory sizes needed for the OBDDs, and the size of the cache to store the sub-solutions is 5,000,000. Note that these DDs use complemented edges. The memory sizes needed for OBDDs and FBDDs are calculated using the formula (1) in Section 3.1. The APLs for DDs are calculated using the method in [29]. The columns "Time1" and "Time2" in Table 1 denote the CPU time for Algorithm 4.1 and Algorithm 4.2, respectively. We used the following environment:

- CPU: Pentium4 Xeon 2.8 GHz
- L1 Cache: 32 KB
- L2 Cache: 512 KB
- Memory: 4 GB
- Operating System: Redhat (Linux 7.3)
- C-compiler: gcc -O2

In Table 1, the bottom row "Average of ratios" denotes the arithmetic average of the relative memory size and APL, where those for OBDD are set to 1.000. These results show that heterogeneous MDDs require comparable memory size to the FBDDs, smaller memory size than OBDDs, and heterogeneous MDDs have shorter APL than OBDDs and FBDDs. And, Algorithm 4.1 and Algorithm 4.2 obtain the optimum solutions in a short computation time.

5.2 Comparison with BDDs and MDD(k)s

Table 2 compares heterogeneous MDDs with BDDs and

Table 2 Comparison of heterogeneous MDDs, BDDs, and MDD(k)s.

DDs	Size	APL
BDD	1.00	1.00
MDD$_{memory}$	0.84	0.69
MDD$_{APL}$	0.97	0.58
MDD(2)	1.08	0.69
MDD(3)	1.54	0.53
MDD(4)	2.39	0.50
MDD(5)	4.10	0.46

Table 1 Memory sizes and APLs for OBDDs, FBDDs, and heterogeneous MDDs.

			$nodes(DDs)$		Memory Size				APL				Time1	Time2
Function	#in	#out	OBDD	FBDD	OBDD	FBDD	MDD1	MDD2	OBDD	FBDD	MDD1	MDD2	[sec]	[sec]
C432	36	7	1063	1061	3189	3183	2910	3180	86.6	86.5	60.1	48.5	0.01	0.01
C880	60	26	4052	2852	12156	8556	11934	12155	135.8	128.8	128.6	112.5	0.01	0.01
C1908	33	25	5525	5067	16575	15201	13493	16570	254.3	231.0	145.8	112.2	0.01	0.05
C2670	233	64	1773	1062	5319	3186	4805	5317	214.0	205.6	178.1	157.1	0.01	0.07
C5315	178	123	1718	1445	5154	4335	4855	5154	462.1	454.3	430.1	395.9	0.01	0.05
C7552	207	107	2211	1589	6633	4767	6383	6633	484.0	466.7	453.8	412.6	0.01	0.05
alu4	14	8	349	298	1047	894	855	1019	40.8	40.3	24.4	19.6	0.01	0.01
apex1	45	45	1245	1356	3735	4068	3117	3734	180.6	189.0	101.2	76.5	0.01	0.01
apex6	135	99	497	456	1491	1368	1437	1491	291.5	284.0	226.4	260.8	0.01	0.01
cps	24	102	970	906	2910	2718	2568	2906	290.3	288.7	172.1	164.7	0.01	0.01
dalu	75	16	688	650	2064	1950	1574	2063	102.7	102.2	49.5	45.8	0.01	0.03
des	256	245	2944	2912	8832	8736	7589	8830	1210.0	1200.0	950.3	810.4	0.01	0.44
frg2	143	139	962	920	2886	2760	2773	2885	624.7	604.3	554.7	536.6	0.01	0.01
i8	133	81	1275	1196	3825	3588	3825	3825	302.5	292.5	302.5	302.5	0.01	0.01
k2	45	45	1245	1141	3735	3423	3119	3728	180.5	177.2	101.2	77.3	0.01	0.01
too_large	38	3	318	284	954	852	859	953	13.2	12.5	8.8	6.6	0.01	0.01
vda	17	39	477	464	1431	1392	1088	1413	176.3	175.4	81.7	79.1	0.01	0.01
Average of ratios					1.000	0.893	0.888	0.997	1.00	0.98	0.73	0.65	–	–

MDD(k)s (k = 2, 3, 4, 5). The numbers of nodes in BDDs were reduced by the dynamic variable reordering [6], [7], [13], [25]. In Table 2, "MDD$_{memory}$" denotes the minimum heterogeneous MDDs obtained by Algorithm 4.1, and "MDD$_{APL}$" denotes the heterogeneous MDDs with the minimum APL obtained by Algorithm 4.2, where Algorithm 4.1 and Algorithm 4.2 used the BDDs as the initial ones, and the memory limitations L for Algorithm 4.2 used the memory sizes needed for the BDDs. The column "Size" shows the arithmetic averages of the relative memory size needed for each DD, where the memory size needed for a BDD is set to 1.00. And, the column "APL" shows the arithmetic averages of the relative APL. Note that no complemented edges are used in these DDs. To obtain an average, 238 benchmark functions are used.

For minimum heterogeneous MDDs, the memory size is 84% of that for BDDs. Specifically, the memory size needed for the minimum heterogeneous MDD for *ex1010* is 46% of the memory size needed for the BDD. Also, the APL for the minimum heterogeneous MDDs is 69% of that for the BDDs.

For MDD(k), we have to increase the memory size to reduce the APL, but in heterogeneous MDDs, we can reduce both the memory size and APL at the same time.

6. Conclusion and Comments

In this paper, we have proposed a new representation of logic functions using Multi-valued Decision Diagrams (MDDs) that is called heterogeneous MDDs. We presented the minimization algorithms for the memory size and the Average Path Length (APL) using new approach, that is the partition of binary variables. Our experimental results with many benchmark functions show that: 1) Heterogeneous MDDs require 84% of the memory size needed for the BDDs, and have 69% of the APL in the BDDs; 2) When heterogeneous MDDs require 97% of the memory size needed for the BDDs, the APLs for heterogeneous MDDs are 58% of the BDDs. 3) Heterogeneous MDDs require comparable memory size to Free Binary Decision Diagrams (FBDDs), and heterogeneous MDDs have shorter APL than FBDDs; 4) The computation time to optimize heterogeneous MDDs is short.

It is important to note that heterogeneous MDDs represent logic functions with small memory size without changing the binary variable order. Also, an optimum heterogeneous MDD can be found relatively easily.

In this paper, the variable order for BDD is fixed. We can obtain the minimum heterogeneous MDDs considering both partitioning and ordering of the input variables by using a similar approach [24].

Acknowledgments

This research is partly supported by the Grant in Aid for Scientific Research of The Japan Society for the Promotion of Science (JSPS), and grant from the MEXT via Kitakyushu innovative cluster project. Comments of Dr. Alan Mishchenko, Prof. Jon. T. Butler, and Yunjian William Jiang improved English presentation. Dr. Wolfgang Günther provided us the data for Free Binary Decision Diagrams (FBDDs) [8], [9].

References

[1] P. Ashar and S. Malik, "Fast functional simulation using branching programs," ICCAD'95, pp.408–412, Nov. 1995.

[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, June 1999.

[3] B. Becker and R. Drechsler, "Efficient graph based representation of multivalued functions with an application to genetic algorithms," Proc. International Symposium on Multiple Valued Logic, pp.40–45, May 1994.

[4] F. Brglez and H. Fujiwara, "Neutral netlist of ten combinational benchmark circuits and a target translator in FORTRAN," Special Session on ATPG and fault simulation, Proc. IEEE Int. Symp. Circuits and Systems, pp.663–698, June 1985.

[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] 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.

[7] M. Fujita, Y. Matsunaga, and T. Kakuda, "On variable ordering of binary decision diagrams for the application of multi-level logic synthesis," EDAC, pp.50–54, March 1991.

[8] W. Günther and R. Drechsler, "Minimization of free BDDs," Asia and South Pacific Design Automation Conference (ASP-DAC'99), pp.323–326, Wanchai, Hong Kong, Jan. 1999.

[9] W. Günther, "Minimization of free BDDs using evolutionary techniques," International Workshop on Logic Synthesis 2000 (IWLS-2000), pp.167–172, Loguna Cliffs Marriott, Dana Point, CA, May–June 2000.

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

[11] Y. Iguchi, T. Sasao, M. Matsuura, and A. Iseno, "A hardware simulation engine based on decision diagrams," Asia and South Pacific Design Automation Conference (ASP-DAC'2000), pp.73–76, Yokohama, Jan. Japan.

[12] Y. Iguchi, T. Sasao, and M. Matsuura, "Implementation of multiple-output functions using PQMDDs," International Symposium on Multiple-Valued Logic, pp.199–205, May 2000.

[13] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of binary decision diagrams based on exchanges of variables," ICCAD, pp.472–475, Nov. 1991.

[14] Y. Jiang and B.K. Brayton, "Software synthesis from synchronous specifications using logic simulation techniques," Design Automation Conference, pp.319–324, New Orleans, LA, U.S.A., June 2002.

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

[16] C. Kim, L. Lavagno, and A. S-Vincentelli, "Free BDD-based software optimization techniques for embedded systems," Design, Automation and Test in Europe (DATE2000), pp.14–19, Paris, March 2000.

[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, June 1992.

[18] P.C. McGeer, K.L. McMillan, A. Saldanha, A.L. Sangiovanni-

Vincentelli, and P. Scaglia, "Fast discrete function evaluation using decision diagrams," ICCAD'95, pp.402–407, Nov. 1995.

[19] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation," Proc. 27th ACM/IEEE Design Automation Conf., pp.52–57, June 1990.

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

[21] S. Nagayama, T. Sasao, Y. Iguchi, and M. Matsuura, "Representations of logic functions using QRMDDs," IEEE International Symposium on Multiple-Valued Logic, pp.261–267, Boston, Massachusetts, U.S.A., May 2002.

[22] S. Nagayama and T. Sasao, "Code generation for embedded systems using heterogeneous MDDs," The 12th workshop on Synthesis and System Integration of Mixed Information technologies (SASIMI 2003), pp.258–264, Hiroshima, Japan, April 2003.

[23] S. Nagayama and T. Sasao, "Compact representations of logic functions using heterogeneous MDDs," IEEE International Symposium on Multiple-Valued Logic, pp.247–252, Tokyo, Japan, May 2003.

[24] S. Nagayama and T. Sasao, "Minimization of memory size for heterogeneous MDDs," Asia and South Pacific Design Automation Conference (ASP-DAC2004), Yokohama, Japan, Jan. 2004 (accepted for publication).

[25] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," ICCAD'93, pp.42–47, 1993.

[26] T. Sasao and M. Fujita, eds., Representations of Discrete Functions, Kluwer Academic Publishers, 1996.

[27] T. Sasao and J.T. Butler, "A method to represent multiple-output switching functions by using multi-valued decision diagrams," IEEE International Symposium on Multiple-Valued Logic, pp.248–254, Santiago de Compostela, Spain, May 1996.

[28] T. Sasao, Switching Theory for Logic Synthesis, Kluwer Academic Publishers, 1999.

[29] T. Sasao, Y. Iguchi, and M. Matsuura, "Comparison of decision diagrams for multiple-output logic functions," International Workshop on Logic and Synthesis, pp.379–384, New Orleans, Louisiana, June 2002.

[30] F. Somenzi, "CUDD: CU decision diagram package release 2.3.1," University of Colorado at Boulder, 2001.

[31] C.D. Thompson, "Area-Time complexity for VLSI," Ann. Symp. on Theory of Computing, pp.81–88, May 1979.

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

Appendix: Proof of Theorem 3.2

Definition A.1: Suppose that a BDD for an n-variable logic function is partitioned into two parts as shown in Fig. A·1. It is partitioned into the **upper part** and the **lower part**. The upper part has the variables $X_{top} = (x_1, x_2, \ldots, x_{n-r})$, while the lower part has the variables $X_{bottom}=(x_{n-r+1}, \ldots, x_n)$.

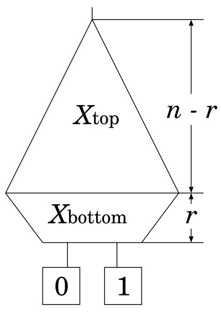


Fig. A·1 Partition of BDD.

Lemma A.1: Consider the variables: $X' = (x_1, x_2, \ldots, x_{n_r})$, where $n_r < n$. When the widths of the BDD are given by

$$width(\text{BDD}, j) = 2^{j-1} \quad (j = 1, 2, \ldots, n_r),$$

the partition of X' that produces the minimum heterogeneous MDD is a trivial partition (i.e., $X' = X_1$, $|X_1| = n_r$). And the memory size needed for the minimum heterogeneous MDD is given by $2^{n_r} + 1$.

(Proof of Lemma A.1) Consider a partition of X': $X' = (X_1, X_2, \ldots, X_s)$, where $s \geq 1$. The memory size needed for a heterogeneous MDD obtained by this partition is

$$A = \sum_{i=1}^{s} (2^{k_i} + 1) width(\text{heterogeneous MDD}, i).$$

When $width(\text{heterogeneous MDD}, j) = 2^{j-1}$ $(j = 1, 2, \ldots, n_r)$, the BDD forms a **complete binary decision tree**. Therefore, we have the following:

$$\begin{aligned} A &= (2^{k_1} + 1) \times 1 + (2^{k_2} + 1) \times 2^{k_1} \\ &+ (2^{k_3} + 1) \times 2^{k_1+k_2} + \ldots \\ &+ (2^{k_s} + 1) \times 2^{k_1+k_2+\ldots+k_{s-1}}. \end{aligned}$$

And, we have:

$$A = 2^{k_1+k_2+\ldots+k_s} + 2^{k_1+k_2+\ldots+k_{s-1}} + B,$$

where

$$B = \sum_{i=1}^{s-1} (2^{k_i} + 1) width(\text{heterogeneous MDD}, i).$$

Since $\sum_{i=1}^{s} k_i = n_r$, we have

$$A = 2^{n_r} + 2^{k_1+k_2+\ldots+k_{s-1}} + B.$$

From the relation $2^{k_1+k_2+\ldots+k_{s-1}} + B \geq 1$, A takes its minimum when $s = 1$. (Q.E.D.)

(Proof of Theorem 3.2) An arbitrary n-variable logic function can be represented by a ROBDD with at most

$$2^{n-r} - 1 + 2^{2^r} - 2$$

nodes [17], where the numbers of nodes in the upper part and the lower part are $2^{n-r} - 1$ and $2^{2^r} - 2$, respectively. From Lemma A.1, the memory size needed for the minimum heterogeneous MDD in the upper part is $2^{n-r} + 1$. Also, from Theorem 3.1, the memory size needed for the minimum heterogeneous MDD in the lower part is at most $3 \times (2^{2^r} - 2)$. Therefore, the memory size needed for this heterogeneous MDD is

$$(2^{n-r} + 1) + \{3 \times (2^{2^r} - 2)\} = 2^{n-r} + 3 \cdot 2^{2^r} - 5.$$

This formula has its minimum value when r is the largest integer that satisfies the relation

$$n - r \geq 2^r + \log_2 3.$$

That is, the memory size needed for the heterogeneous MDD is minimum when r satisfies this condition. (Q.E.D.)

Shinobu Nagayama was born on July 31, 1977 in Kanagawa Japan, and received the B.S. and M.E. from Meiji University, Kanagawa Japan, in 2000 and 2002, respectively. He is now a doctoral student of Kyushu Institute of Technology. His research interest includes decision diagrams, software synthesis, and embedded systems.

Tsutomu Sasao received the B.E., M.E., and Ph.D. degrees in Electronics Engineering from Osaka University, Osaka Japan, in 1972, 1974, and 1977, respectively. He has held faculty/research positions at Osaka University, Japan, IBM T.J. Watson Research Center, Yorktown Height, NY and the Naval Postgraduate School, Monterey, CA. Now, he is a Professor of Department of Computer Science and Electronics, as well as the Director of the Center for Microelectronic Systems at the 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 9 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 Publishers 1993, 1996, 1999, 2001 respectively. He has served 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 NIWA Memorial Award in 1979, and Distinctive Contribution Awards from IEEE Computer Society MVL-TC in 1987 and 1996 for papers presented at ISMVLs, and Takeda Techno-Entrepreneurship Award in 2001. He has served an associate editor of the IEEE Transactions on Computers. Currently, he is the Chairman of the Technical Committee on Multiple-Valued Logic, IEEE Computer Society. He is a Fellow of the IEEE.