IEICE TRANS. FUNDAMENTALS, VOL.E87–A, NO.5 MAY 2004

PAPER Special Section on Discrete Mathematics and Its Applications

# Area-Time Complexities of Multi-Valued Decision Diagrams

**Shinobu NAGAYAMA**[†a)], ***Student Member***, **Tsutomu SASAO**[†,††b)], **Yukihiro IGUCHI**[†††c)], ***and*** **Munehiro MATSUURA**[†d)], ***Members***

**SUMMARY** This paper considers Quasi-Reduced ordered Multi-valued Decision Diagrams with $k$ bits (QRMDD($k$)s) to represent binary logic functions. Experimental results show relations between the values of $k$ and the numbers of nodes, the memory sizes, the numbers of memory accesses, and area-time complexity for QRMDD($k$). For many benchmark functions, the numbers of nodes and memory accesses for QRMDD($k$)s are nearly equal to $\frac{1}{k}$ of the corresponding Quasi-Reduced ordered Binary Decision Diagrams (QRBDDs), and the memory sizes and the area-time complexities for QRMDD($k$)s are minimum when $k = 2$ and $k = 3$–6, respectively.

***key words:*** *decision diagrams, the number of nodes, area-time complexity, randomly generated function, representation of logic functions*

## 1. Introduction

Binary Decision Diagrams (BDDs) [7] and Multi-valued Decision Diagrams (MDDs) [3], [14], [19], [20] are extensively used in logic synthesis [9], logic simulation [1], [11], [17], software synthesis [2], etc. Since modern computer systems have the memory hierarchical structure, suitable decision diagrams for the memory hierarchy can shorten the runtimes of these applications [17], [34]. Quasi-Reduced ordered BDDs (QRBDDs) and Quasi-Reduced ordered MDDs (QRMDDs) are suitable for the memory hierarchy [23], parallel process [12], [24], and design of LUT cascades [31]. However, in general, QRBDDs and QRMDDs require more nodes than corresponding Reduced ordered BDDs (RBDDs) and Reduced ordered MDDs (RMDDs) to represent logic functions. Hence, the minimizations of QRBDDs and QRMDDs are very important. In many cases, the minimizations of decision diagrams use the variable reordering [8], [9], [13], [21], [26]. In the minimization of MDDs, a partition of binary variables [10], [28] is important as well as the variable ordering.

To represent a binary logic function using an MDD, binary variables are partitioned into groups. The papers [10], [28] present the optimization algorithm of partition of input binary variables into groups of binary variables. However, the size of groups (i.e. the number of binary variables in a group) is fixed in these algorithms. In this paper, we assume that the size of groups, that is the value of $k$ for Quasi-Reduced ordered MDDs with $k$ bits (QRMDD($k$)s), can be changed, and we find the optimum sizes of groups experimentally by showing the relations of the values of $k$ and the numbers of nodes, the memory sizes, and the numbers of memory accesses. To show these relations, we assume that the order of binary variable is fixed. Our statistical results are useful for minimizations of MDDs, software synthesis [2], and logic simulation [1], [11], [17].

The rest of the paper is organized as follows: Sect. 2 defines MDD($k$)s, QRMDD($k$)s, computation model for MDDs, and a method to represent multiple-output functions. Section 3 considers the number of nodes in QRMDD($k$)s for general functions, benchmark functions, and randomly generated functions. Section 4 introduces the measure called *area-time complexity* [4], [33] to find the optimum value of $k$ for QRMDD($k$)s, and derives the optimum values of $k$ by experiments.

This paper is an extended version of [22].

## 2. Definitions

### 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. And, 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 $|X_i| = k$ $(i = 1, 2, \ldots, u)$, then a two-valued logic function $f(X)$ can be represented by the mapping $f(X_1, X_2, \ldots, X_u)$: $\{0, 1, 2, \ldots, 2^k - 1\}^u \rightarrow \{0, 1\}$.

### 2.2 QRMDD($k$)

In this paper, we use the standard terminologies for Reduced ordered Binary Decision Diagrams (RBDDs) [7] and Reduced ordered MDDs (RMDDs) [14].

**Definition 2.2:** When $X = (x_1, x_2, \ldots, x_n)$ is partitioned

Manuscript received August 22, 2003.
Manuscript revised November 10, 2003.
Final manuscript received December 26, 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.
†††The author is with the Department of Computer Science, Meiji University, Kawasaki-shi, 214-8571 Japan.
a) E-mail: nagayama@aries02.cse.kyutech.ac.jp
b) E-mail: sasao@cse.kyutech.ac.jp
c) E-mail: iguchi@cs.meiji.ac.jp
d) E-mail: matsuura@cse.kyutech.ac.jp

into $(X_1, X_2, \ldots, X_u)$, where $|X_i| = k$ $(i = 1, 2, \ldots, u)$, an RMDD representing a logic function $f(X)$ is called a **Reduced ordered MDD with $k$ bits (RMDD($k$))**. In an RMDD($k$), non-terminal nodes have $2^k$ outgoing edges. When $k = 1$, an RMDD(1) is an RBDD.

If $|X_1| = |X_2| = \ldots = |X_{u-1}| = k$ and $|X_u| < k$, then for $|X_u| = k$, we use redundant binary variables which are called **dummy variables**. The set of binary variables with dummy variables is denoted by $\{X'\} = \{x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+t}\}$, where $|X'| = n + t$, and $t$ denotes the number of dummy variables. Note that $f$ is independent of $x_{n+1}, x_{n+2}, \ldots$ and $x_{n+t}$.

**Definition 2.3:** In a decision diagram (DD), a path from the root node to a terminal node is a **path of DD**. The number of non-terminal nodes on the path is the **length of the path**.

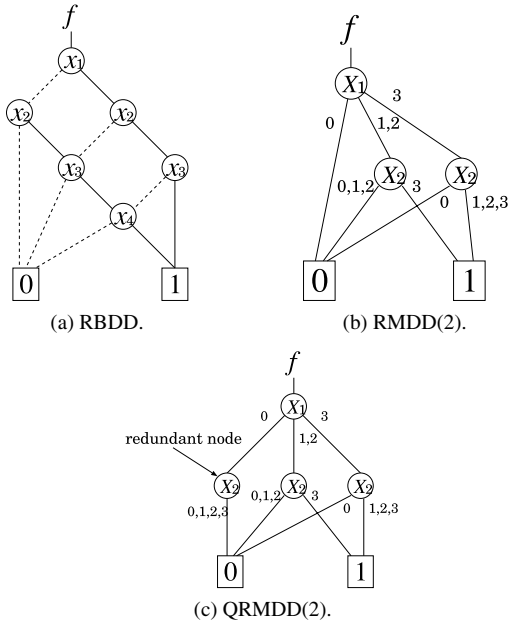
**Definition 2.4:** In a DD, the **number of nodes in the DD**, denoted by *nodes*(DD), includes only non-terminal nodes.

**Definition 2.5:** When all $X_i$ $(i = 1, 2, \ldots, u)$ appear in this order on an arbitrary path of an MDD($k$), the MDD($k$) is a **Quasi-Reduced ordered MDD with $k$ bits (QRMDD($k$))**. QRMDD($k$) has no isomorphic subgraphs.

The length of an arbitrary path in a QRMDD($k$) is the number of super variables. An RMDD($k$) has no redundant nodes, while a QRMDD($k$) usually has redundant nodes. Therefore, we have the following relation in the number of nodes of an RMDD($k$) and its corresponding QRMDD($k$):

$$nodes(\text{RMDD}(k)) \leq nodes(\text{QRMDD}(k)).$$

**Example 2.1:** Consider a logic function $f = x_1x_2x_3 \vee x_2x_3x_4 \vee x_3x_4x_1 \vee x_4x_1x_2$. The RBDD, the RMDD(2), and the QRMDD(2) for $f$ are shown in Figs. 1(a), (b), and (c), respectively. In Fig. 1(a), the solid lines and the broken lines denote 1-edges and 0-edges, respectively. In Figs. 1(b) and (c), 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 $X_2 = (x_3, x_4)$. We have *nodes*(RBDD) = 6, *nodes*(RMDD(2)) = 3, and *nodes*(QRMDD(2)) = 4. (End of Example)




(a) RBDD. (b) RMDD(2).

(c) QRMDD(2).

**Fig. 1** DDs for $f$.

### 2.3 Computation Model for MDDs

In an MDD, we assume the following computation model:

1. MDD($k$)s for logic functions are evaluated by traversing nodes from the root node to a terminal node according to values of the super variables.
2. MDDs are implemented directly, not simulated using the BDD package as described in [14].
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 super variable.
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 QRMDD for a logic function is proportional to the path length (i.e., the number of super variables).

### 2.4 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 $F = (f_0, f_1, \ldots, f_{m-1})$: $B^n \rightarrow B^m$, where $B = \{1, 0\}$, and $n$ and $m$ denote the number of input variables and outputs, respectively. Several methods exist to represent multiple-output functions by using DDs [18], [27]–[29], [32]. In this paper, we use an encoded characteristic function for non-zero output (ECFN) [30], [32] to represent multiple-output functions. An ECFN uses $\lceil \log_2 m \rceil$ auxiliary variables to represent the outputs. In the following, a DD means a DD for an ECFN.

**Definition 2.6:** The **density** for an $n$-variable logic function $f$ is defined as

$$\frac{|f|}{2^n} \times 100,$$

where $|f|$ denotes the number of $\vec{a}$ such that $f(\vec{a}) = 1$.

The density for a multiple-output function $F$ is the density for an ECFN representing $F$.

## 3. Number of Nodes in QRMDD($k$)

In this section, we first obtain an upper bound on the number of nodes in a QRMDD($k$). Then, we obtain the numbers of nodes in QRMDD($k$)s for benchmark functions, and show that an interesting property holds for many benchmark functions. Finally, we obtain the numbers of nodes in

**Table 1** Upper bounds on the number of nodes in QRMDD($k$).

| | $k$ | | | | |
|---|---|---|---|---|---|
| $n$ | 1 | 2 | 3 | 4 | 5 |
| 10 | 275 | 101 | 77 | 33 | 33 |
| 11 | 531 | 345 | 89 | 273 | 37 |
| 12 | 787 | 357 | 329 | 273 | 49 |
| 13 | 1299 | 601 | 589 | 277 | 289 |
| 14 | 2323 | 1381 | 601 | 289 | 1057 |
| 15 | 4371 | 1625 | 841 | 529 | 1057 |
| 16 | 8467 | 5477 | 4685 | 4369 | 1061 |
| 17 | 16659 | 5721 | 4697 | 4373 | 1073 |
| 18 | 33043 | 21861 | 4937 | 4385 | 1313 |
| 19 | 65811 | 22105 | 37453 | 4625 | 33825 |
| 20 | 131347 | 87397 | 37465 | 69905 | 33825 |

QRMDD($k$)s for randomly generated functions, and show that they have quite different property from the benchmark functions.

### 3.1 Number of Nodes for General Functions

**Theorem 3.1:** An arbitrary $n$-variable logic function can be represented by a QRBDD with at most

$$2^{n-r} - 1 + \sum_{i=1}^{r} 2^{2^i}$$

non-terminal nodes, where $r$ is the largest integer that satisfies relation $n - r \geq 2^r$ [16].

**Theorem 3.2:** An arbitrary $n$-variable logic function can be represented by a QRMDD($k$) with at most

$$\frac{2^{sk} - 1}{2^k - 1} + \sum_{i=1}^{u-s} 2^{2^{(ki-t)}}$$

non-terminal nodes, where $u$ is the number of super variables, $t$ is the number of dummy variables, and $s$ is the smallest integer that satisfy relation

$$s \geq \frac{n-r}{k}.$$

Appendix B and Appendix C show the proofs of Theorem 3.1 and Theorem 3.2.

Table 1 shows the upper bounds on the number of nodes in QRMDD($k$)s for $n$-variable logic functions. We can see that the upper bounds are non-monotone functions of $k$.

### 3.2 Number of Nodes for Benchmark Functions

We used 157 benchmark functions [5], [29], [36] in Table A· 1. In this paper, encodings for ECFNs and binary variable orders of BDDs are obtained by the heuristic algorithm in [32]. In the following experiments, we use these variable orders, and we consider only the partition of binary variables. For each benchmark function, we counted the number of nodes in the corresponding QRMDD($k$)s for various $k$. In Table 2, *avg* denotes the arithmetic average of the relative numbers of nodes, where the number of nodes in QRBDD is set to 1.00, and *stdv* denotes the standard deviation.

**Table 2** Relation of nodes in QRMDD($k$) and $k$ for benchmark functions.

| | $k$ | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| *avg* | 1.000 | 0.498 | 0.333 | 0.248 | 0.202 |
| *stdv* | 0.000 | 0.013 | 0.009 | 0.016 | 0.016 |

**Table 3** Benchmark functions with $\eta \geq 0.1$.

| Name | # in | # out | # nodes | dens. | cate. |
|---|---|---|---|---|---|
| C499 | 41 | 32 | 24476 | 50.0 | 1 |
| C1355 | 41 | 32 | 30156 | 50.0 | 1 |
| C1908 | 33 | 25 | 9292 | 45.8 | 1 |
| adr8 | 16 | 9 | 153 | 50.0 | 2 |
| adr9 | 18 | 10 | 180 | 50.0 | 2 |
| comp | 32 | 3 | 114 | 37.5 | 2 |
| inc17 | 17 | 18 | 236 | 48.4 | 3 |
| log16 | 16 | 16 | 11216 | 59.9 | 1 |
| log17 | 17 | 17 | 23054 | 55.1 | 1 |
| log18 | 18 | 18 | 31458 | 55.2 | 1 |
| mlp8 | 16 | 16 | 10112 | 41.5 | 1 |
| mlp9 | 18 | 18 | 28332 | 37.5 | 1 |
| mlp10 | 20 | 20 | 82077 | 38.5 | 1 |
| my_adder | 33 | 17 | 450 | 50.0 | 2 |
| nrm8 | 16 | 9 | 8689 | 49.1 | 1 |
| nrm9 | 18 | 10 | 23152 | 49.0 | 1 |
| pcle | 19 | 9 | 221 | 29.3 | 3 |
| rot16 | 16 | 9 | 1021 | 60.8 | 3 |
| rot17 | 17 | 9 | 1429 | 49.3 | 3 |
| sqr16 | 16 | 32 | 18366 | 42.9 | 1 |
| tcon | 17 | 16 | 183 | 50.0 | 3 |
| vg2 | 25 | 8 | 217 | 22.9 | 3 |
| vtx1 | 27 | 6 | 326 | 12.5 | 3 |
| x1dn | 27 | 6 | 332 | 12.5 | 3 |

**Definition 3.1:** The relation '$\simeq$' is defined as follows:

$$a \simeq b \Leftrightarrow \eta < 0.1,$$

where $a$ and $b$ are positive integers, and the **normalized difference** $\eta$ is given by:

$$\eta = \frac{|a-b|}{\min(a,b)}.$$

If $a \simeq b$, then $a$ and $b$ are **nearly equal**.

For 133 functions in Table A· 1, the following property holds.

**Property 3.1:**

$$nodes(\text{QRMDD}(k)) \simeq \frac{1}{k} nodes(\text{QRBDD})$$

For the remaining 24 functions, $\eta \geq 0.1$ holds. Table 3 lists these 24 functions. In Table 3, "# nodes," "dens.," and "cate." denote the numbers of nodes in QRBDDs, the densities, and the categories of functions described below, respectively. Figure 2 shows the relation between the normalized difference $\eta$ and the density for benchmark functions. The symbols +,×,⊙, and △ correspond to the values for $k = 2, 3, 4$, and 5, respectively. For each function, we assume that Property 3.1 holds when all the symbols are below the border line of $\eta = 0.1$ (i.e., $\eta < 0.1$ holds for $k =$ 2–5). From Fig. 2, we categorized 24 functions in Table 3 into three sets.

1. The densities of functions are between 40% and 60%, and the number of nodes for QRBDDs are large relative to the number of inputs.

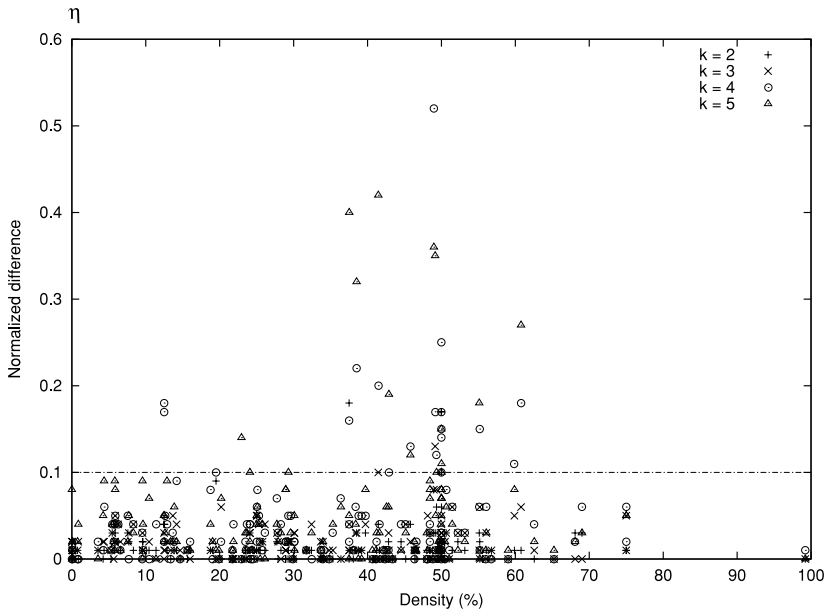



**Fig. 2** Relation between the normalized difference $\eta$ and density for benchmark functions.

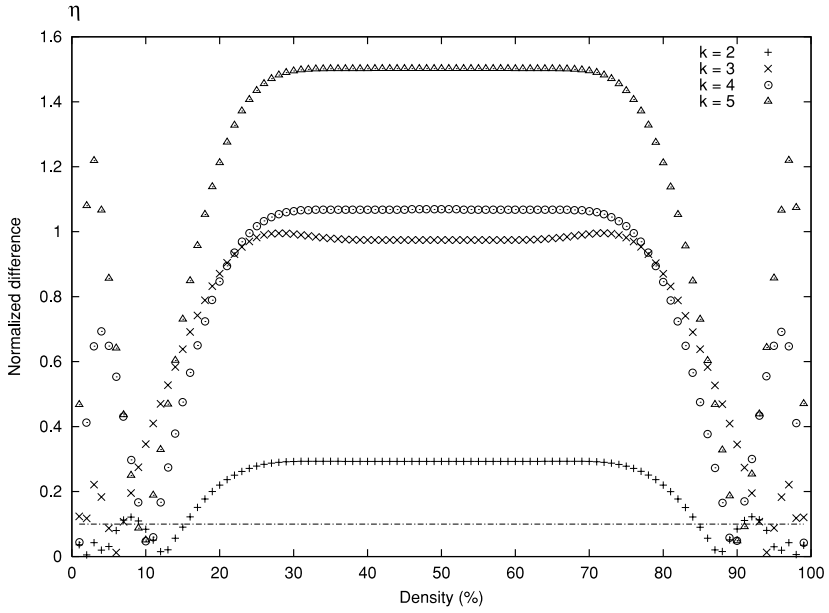



**Fig. 3** Relation between the normalized difference $\eta$ and density for randomly generated functions.

2. The functions have iterative properties (i.e., adder and comparator).
3. The numbers of nodes, inputs, and outputs are small. Property 3.1 does not hold for $k = 4$ or 5.

### 3.3 Number of Nodes for Randomly Generated Functions

For $d = 1, 2, \ldots, 99$, we randomly generated one 25-variable function with density $d$ to obtain 99 functions. Figure 3 shows the relation between the normalized difference $\eta$ and the density for randomly generated functions. In this case, no randomly generated functions of 25 variables satisfied Property 3.1. This fact shows that randomly generated functions have quite different property from the benchmark functions in Table A·1. For many benchmark functions, the numbers of nodes in QRMDD($k$)s decrease as $k$ increase. However, for randomly generated functions, the number of nodes is a non-monotone function of $k$. For example, for many randomly generated functions of 25 variables, the numbers of nodes in QRMDD(5)s were larger than those in QRMDD(3)s.

For $n = 10, 11, \ldots, 20$, we also randomly generated ten $n$-variable functions with density 50%. Table 4 shows the average numbers of nodes in QRMDD($k$)s for randomly

**Table 4** Number of nodes in QRMDD($k$) for randomly generated functions.

| | $k$ | | | | |
|---|---|---|---|---|---|
| $n$ | 1 | 2 | 3 | 4 | 5 |
| 10 | 247.4 | 101.0 | 77.0 | 33.0 | 33.0 |
| 11 | 437.1 | 251.2 | 89.0 | 179.2 | 37.0 |
| 12 | 754.0 | 356.5 | 296.5 | 272.5 | 49.0 |
| 13 | 1292.8 | 596.6 | 587.2 | 277.0 | 284.6 |
| 14 | 2316.0 | 1374.1 | 601.0 | 289.0 | 1050.1 |
| 15 | 4341.1 | 1625.0 | 841.0 | 529.0 | 1057.0 |
| 16 | 8336.5 | 5346.5 | 4554.5 | 4238.5 | 1061.0 |
| 17 | 16165.3 | 5721.0 | 4697.0 | 4373.0 | 1073.0 |
| 18 | 31155.9 | 19973.9 | 4937.0 | 4385.0 | 1313.0 |
| 19 | 58836.4 | 22105.0 | 30478.4 | 4625.0 | 26850.4 |
| 20 | 107220.3 | 63270.3 | 37465.0 | 45778.3 | 33825.0 |

generated functions. The deviations were within ±2% of the averages. From Table 1 and Table 4, we can see that the numbers of nodes in QRMDD($k$)s for randomly generated functions with density 50% are nearly equal to the upper bounds.

## 4. Area-Time Complexity of QRMDD($k$)

### 4.1 Memory Size for QRMDD($k$)

**Definition 4.1:** The **memory size for a QRMDD($k$)** is the number of bits needed to store the QRMDD($k$) in memory.

In memory, a non-terminal node in an RMDD($k$) requires an index and a set of pointers that refer the succeeding nodes. However, in a QRMDD($k$), each non-terminal node has no index because $X_1, X_2, \ldots, X_u$ are evaluated always in this order, and the index of the super variable to evaluate can be obtained by a counter, where the super variable order is $X_1, X_2, \ldots, X_u$.

**Example 4.1:** Figure 4 illustrates data structures of a node in an RMDD(2) and a QRMDD(2). (End of Example)

Because each node in a QRMDD($k$) has $2^k$ outgoing edges, we need

$$2^k nodes(\text{QRMDD}(k))$$

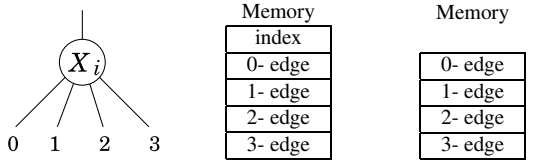
words to store all nodes in a QRMDD($k$). Since each node in a memory requires a unique address, each pointer requires

$$\lceil \log_2(nodes(\text{QRMDD}(k))) \rceil$$

bits to specify the address. Therefore, the memory size for a QRMDD($k$) is

$$2^k nodes(\text{QRMDD}(k)) \lceil \log_2(nodes(\text{QRMDD}(k))) \rceil.$$

As shown in Sect. 3.2, for many benchmark functions, the numbers of nodes in QRMDD($k$)s can be reduced with increasing $k$. On the other hand, the memory sizes for QRMDD($k$)s increase with $2^k$. This fact shows that in QRMDD($k$), there exists optimum value of $k$ that minimizes the memory size.




(a) A node in DDs. (b) RMDD(2). (c) QRMDD(2).

**Fig. 4** Data structure of a node in DDs.

### 4.2 Area-Time Complexity of QRMDD($k$)s

Because a QRMDD($k$) evaluates $k$ binary variables at a time, the number of memory accesses of a QRMDD($k$) is $\frac{1}{k}$ of the corresponding QRBDD. On the other hand, the memory size for a QRMDD($k$) increases with $2^k$. In this section, we consider the area-time complexity [4], [33] for QRMDD($k$) and obtain the $k$ that minimizes the area-time complexity.

**Definition 4.2:** The **area-time complexity** is the measure of computational cost considering both area and time. It is defined by

$$AT = (area) \times (time), \quad AT^2 = (area) \times (time)^2.$$

In this paper, the area $A$ corresponds to the necessary memory size for QRMDD($k$), and the time $T$ corresponds to the number of memory accesses to evaluate logic function.

The measure $AT$ is used when both the memory size and the number of memory accesses are equally important. The measure $AT^2$ is used when the number of memory accesses is more important than the memory size. For example, $AT$ can be used for software synthesis, while $AT^2$ can be used for logic simulators. In the software synthesis for embedded systems [2], compact and fast program codes are required because of the memory limitations and the time limitations for systems. Thus, in the software synthesis using DDs, the optimization of DDs considering both the memory size and the number of memory accesses is important. In logic simulators [1], [11], [17], fast evaluation of logic functions is more important to reduce the design verification time. Thus, in logic simulators, minimizing the number of memory accesses using a reasonable amount of memory is important.

### 4.3 Experimental Results

For each benchmark function in Table A·1, we obtained three measures $A$, $AT$, and $AT^2$. Table 5, Table 6, and Table 7 show the relations of $k$ and $A$, $AT$, and $AT^2$, respectively. In these tables, *avg* denotes the arithmetic average, and *stdv* denotes the standard deviation for benchmark functions.

For each benchmark function in Table A·1, $A$ takes its minimum when $k = 2$; $AT$ takes its minimum when $k = 3$ or $k = 4$; and $AT^2$ takes its minimum when $k = 4$–6.

**Table 5** Relation of $k$ and $A$ for QRMDD($k$) for benchmark functions.

| | $k$ | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| *avg* | 1.00 | 0.90 | 1.14 | 1.61 | 2.54 |
| *stdv* | 0.000 | 0.035 | 0.079 | 0.144 | 0.292 |

**Table 6** Relation of $k$ and $AT$ for QRMDD($k$) for benchmark functions.

| | $k$ | | | | |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| *avg* | 1.00 | 0.46 | 0.39 | 0.42 | 0.54 |
| *stdv* | 0.000 | 0.019 | 0.030 | 0.039 | 0.070 |

**Table 7** Relation of $k$ and $AT^2$ for QRMDD($k$) for benchmark functions.

| | $k$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| *avg* | 1.000 | 0.232 | 0.133 | 0.110 | 0.114 | 0.128 | 0.167 |
| *stdv* | 0.000 | 0.011 | 0.012 | 0.012 | 0.019 | 0.023 | 0.046 |

### 4.4 Analysis for the Functions that Satisfy Property 3.1

In Sect. 4.3, for QRMDD($k$)s, we found the values of $k$ that make $A$, $AT$, and $AT^2$ minimum, experimentally. In this section, we assume that Property 3.1 holds, and will find the values $k$ that make $A$, $AT$, and $AT^2$ minimum, analytically. Let $A$ and $T$ be the memory size for a QRMDD($k$) and the number of memory accesses necessary to evaluate a QRMDD($k$), respectively. Then, we have the following:

$$A = 2^k nodes(\text{QRMDD}(k))\lceil \log_2(nodes(\text{QRMDD}(k)))\rceil,$$
$$T = \left\lceil \frac{n}{k} \right\rceil.$$

Let $nodes(\text{QRMDD}(1)) = N$ and assume that Property 3.1 holds. Then we have:

$$A \simeq \frac{2^k}{k} N \left\lceil \log_2\left(\frac{N}{k}\right)\right\rceil,$$
$$AT \simeq \frac{2^k n}{k^2} N \left\lceil \log_2\left(\frac{N}{k}\right)\right\rceil,$$
$$AT^2 \simeq \frac{2^k n^2}{k^3} N \left\lceil \log_2\left(\frac{N}{k}\right)\right\rceil.$$

Note that $N$ is usually so greater than 200, while $k$ is usually at most 7. Thus, we can use the following approximation:

$$\lceil \log_2(N) - \log_2(k)\rceil \simeq \lceil \log_2(N)\rceil.$$

Therefore, $A$, $AT$, and $AT^2$ can be simplified to

$$A \simeq \frac{2^k}{k} C_0, \quad AT \simeq \frac{2^k}{k^2} C_1, \quad \text{and} \quad AT^2 \simeq \frac{2^k}{k^3} C_2,$$

respectively, where the constants $C_0$, $C_1$ and $C_2$ are independent of $k$. From the above formulas, we can see that $A$, $AT$, and $AT^2$ take their minimum when $k = 2$, $k = 3$, and $k = 4$, respectively.

## 5. Conclusion and Comments

In this paper, we considered representations of logic functions using QRMDD($k$)s. Experimental results showed that: 1) For many benchmark functions, the numbers of nodes in QRMDD($k$)s are nearly equal to $\frac{1}{k}$ of the corresponding QRBDDs. On the other hand, for randomly generated functions, the number of nodes is a non-monotone function of $k$. 2) For many benchmark functions, the memory sizes and the area-time complexities for QRMDD($k$)s take their minimum when $k = 2$ and $k = 3$–6, respectively.

In commercial LUT-based FPGAs, the numbers of inputs $k$ for LUT cells are usually between 4 and 6 [6]. The studies in [15], [25] show that when $k = 4$–6, the architectures of FPGAs are optimum. The cost of $k$-LUT cell increases with $k$, while the level of network reduces with $k$. Thus, in logic synthesis with FPGAs, we can do a similar discussion. However, the optimum value of $k$ for FPGAs depends on interconnection delay, logic synthesis tools, and process technology as well as the cost of $k$-LUT cell and the level of networks [35]. It is interesting that in both cases, the optimum values of $k$ are 4–6 even if they have different cost functions.

## 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 the Takeda Foundation. Prof. Jon. T. Butler improved presentation. The comments of a reviewer motivated to produce Fig. 2 and Fig. 3.

### 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 multi-valued functions with an application to genetic algorithms," Proc. International Symposium on Multiple Valued Logic, pp.40–45, May 1994.
[4] R.P. Brent and H.T. Kung, "The area-time complexity of binary multiplication," J. ACM, vol.28, no.3, pp.521–534, July 1981.
[5] 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.
[6] S. Brown, R. Francis, J. Rose, and Z. Vranesic, Field-Programmable Gate Arrays, Kluwer Academic Publishers, 1992.
[7] R.E. Bryant, "Graph-based algorithms for Boolean function manipulation," IEEE Trans. Comput., vol.C-35, no.8, pp.677–691, Aug. 1986.
[8] 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.

[9] 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.
[10] H.Md. Hasan 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, Japan, Jan. 2000.
[12] Y. Iguchi, T. Sasao, and M. Matsuura, "Implementation of multiple-output functions using PQMDDs," Proc. 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] T. Kam, T. Villa, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, "Multi-valued decision diagrams: Theory and applications," Multiple-Valued Logic, vol.4, no.1-2, pp.9–62, 1998.
[15] Kouloheris and A. El Gamal, "FPGA area versus cell granularity—Lookup tables and PLA cells," Proc. ACM First Int. Workshop on Field Programmable Gate Arrays, pp.9–14, Feb. 1992.
[16] 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.
[17] 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.
[18] 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.
[19] D.M. Miller, "Multiple-valued logic design tools," Proc. International Symposium on Multiple Valued Logic, pp.2–11, May 1993.
[20] D.M. Miller and R. Drechsler, "Implementing a multiple-valued decision diagram package," Proc. International Symposium on Multiple-Valued Logic, pp.52–57, Fukuoka, 1998.
[21] D.M. Miller and R. Drechsler, "Augmented sifting of multiple-valued decision diagrams," Proc. 33rd International Symposium on Multiple-Valued Logic, pp.375–382, Tokyo, Japan, 2003.
[22] S. Nagayama, T. Sasao, Y. Iguchi, and M. Matsuura, "Representations of logic functions using QRMDDs," Proc. International Symposium on Multiple-Valued Logic, pp.261–267, Boston, Massachusetts, U.S.A., May 2002.
[23] H. Ochi, K. Yasuoka, and S. Yajima, "Breadth-first manipulation of very large binary-decision diagrams," 1993 IEEE/ACM International Conference on Computer-Aided Design, pp.48–55, Nov. 1993.
[24] H. Ochi, N. Ishiura, and S. Yajima, "Breadth-first manipulation of SBDD of Boolean function for vector processing," 28th ACM/IEEE Design Automation Conference, pp.413–416, 1991.
[25] J. Rose, R.J. Francis, D. Lewis, and P. Chow, "Architecture of field-programmable gate arrays: The effect of logic block functionality on area efficiency," IEEE J. Solid-State Circuits, vol.25, no.5, pp.1217–1225, Oct. 1990.
[26] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," ICCAD'93, pp.42–47.
[27] T. Sasao and M. Fujita, eds., Representations of Discrete Functions, Kluwer Academic Publishers 1996.
[28] T. Sasao and J.T. Butler, "A method to represent multiple-output switching functions by using multi-valued decision diagrams," Proc. International Symposium on Multiple-Valued Logic, pp.248–254, Santiago de Compostela, Spain, May 1996.
[29] T. Sasao, Switching Theory for Logic Synthesis, Kluwer Academic Publishers, 1999.
[30] T. Sasao, "Compact SOP representations for multiple-output functions: An encoding method using multiple-valued logic," Proc. International Symposium on Multiple-Valued Logic, pp.207–211, Warsaw, Poland, May 2001.
[31] T. Sasao, M. Matsuura, and Y. Iguchi, "A cascade realization of multiple-output function and its application to reconfigurable hardware," International Workshop on Logic and Synthesis, pp.225–230, Lake Tahoe, June 2001,
[32] T. Sasao, M. Matsuura, Y. Iguchi, and S. Nagayama, "Compact BDD representations for multiple-output functions and their application," IFIP VLSI-SOC'01, pp.406–411, Montpellier, France, Dec. 2001.
[33] C.D. Thompson, "Area-time complexity for VLSI," Ann. Symp. on Theory of Computing, pp.81–89, May 1979.
[34] I. Wegener, Branching Programs and Binary Decision Diagrams, Theory and Applications, SIAM, 2000.
[35] A. Yan, R. Cheng, and S.J. E. Wilton, "On the sensitivity of FPGA architectural conclusions to the experimental assumptions, tools, and techniques," ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp.147–156, Monterey, CA, Feb. 2002.
[36] S. Yang, Logic synthesis and optimization benchmark user guide version 3.0, MCNC, Jan. 1991.

## Appendix A: Benchmark Functions

In this paper, we use 157 benchmark functions [5], [29], [36] shown in Table A· 1, where $n$ and $m$ denote the number of input and output variables, respectively. In this table, the benchmark functions under *sequential* originally represented sequential circuits. We removed flip-flops (FFs) from these sequential circuits to make them combinational. Such functions are renamed by appending a subscript '$c$' to the original names.

## Appendix B: Proof of Theorem 3.1

**Definition B.1:** Suppose that a QRBDD for an $n$-variable logic function is partitioned into two parts as shown in Fig. A· 1. It is partitioned into **the upper part** which has the variables $X_1=(x_1, x_2, \ldots, x_{n-r})$, and **the lower part** which has the variables $X_2=(x_{n-r+1}, \ldots, x_n)$. In this case, the BDD represents the logic function as follows:

$$f(X_1, X_2) = \bigvee_{\vec{a}_i \in B^{n-r}} X_1^{\vec{a}_i} f(\vec{a}_i, X_2),$$

where

$$X_1^{\vec{a}_i} = \begin{cases} 1 & (X_1 = \vec{a}_i) \\ 0 & (otherwise). \end{cases}$$

The upper part realizes $X_1^{\vec{a}_i}$, and the lower part realizes $f(\vec{a}_i, X_2)$.

**(Proof of Theorem 3.1)** When the upper part of the QRBDD (see Fig. A· 1) has $2^{n-r} - 1$ nodes (i.e., a complete binary tree), it is the maximum. Because $f(\vec{a}_i, X_2)$ is an $r$-variable logic function, the number of different $f(\vec{a}_i, X_2)$ is $2^{2^r}$. When $2^{2^i}$ logic functions are realized for each level $i$ ($i = 1, 2, \ldots, r$) from the terminal node to the $r$, the lower part is the maximum. Therefore, the number of nodes in a QRBDD is at most

$$2^{n-r} - 1 + \sum_{i=1}^{r} 2^{2^i}.$$

**Table A· 1** Benchmark functions.

| Name | n | m | Name | n | m | Name | n | m |
|---|---|---|---|---|---|---|---|---|
| 3adr6 | 18 | 12 | i5 | 133 | 66 | signet | 39 | 8 |
| C432 | 36 | 7 | i6 | 138 | 67 | soar | 83 | 94 |
| C499 | 41 | 32 | i7 | 199 | 67 | spla | 16 | 46 |
| C880 | 60 | 26 | i8 | 133 | 81 | sqr16 | 16 | 32 |
| C1355 | 41 | 32 | i9 | 88 | 63 | t1 | 21 | 23 |
| C1908 | 33 | 25 | i10 | 257 | 224 | t2 | 17 | 16 |
| C2670 | 233 | 140 | ibm | 48 | 17 | table5 | 17 | 15 |
| C3540 | 50 | 22 | in1 | 16 | 17 | tcon | 17 | 16 |
| C5315 | 178 | 123 | in2 | 19 | 10 | term1 | 34 | 10 |
| C7552 | 207 | 108 | in3 | 35 | 29 | ti | 47 | 72 |
| accpla | 50 | 69 | in4 | 32 | 20 | too_large | 38 | 3 |
| adr8 | 16 | 9 | in5 | 24 | 14 | ts10 | 22 | 16 |
| adr9 | 18 | 10 | in6 | 33 | 23 | ttt2 | 24 | 21 |
| al2 | 16 | 47 | in7 | 26 | 10 | unreg | 36 | 16 |
| alcom | 15 | 38 | inc16 | 16 | 17 | vda | 17 | 39 |
| apex1 | 45 | 45 | inc17 | 17 | 18 | vg2 | 25 | 8 |
| apex2 | 39 | 3 | inc18 | 18 | 19 | vtx1 | 27 | 6 |
| apex3 | 54 | 50 | jbp | 36 | 57 | wgt17 | 17 | 5 |
| apex5 | 117 | 88 | k2 | 45 | 45 | wgt18 | 18 | 5 |
| apex6 | 135 | 99 | lal | 26 | 19 | x1 | 51 | 35 |
| apex7 | 49 | 37 | log16 | 16 | 16 | x3 | 135 | 99 |
| b2 | 16 | 17 | log17 | 17 | 17 | x4 | 94 | 71 |
| b3 | 32 | 20 | log18 | 18 | 18 | x1dn | 27 | 6 |
| b4 | 33 | 23 | mainpla | 27 | 54 | x2dn | 82 | 56 |
| b9 | 41 | 21 | mark1 | 20 | 31 | x6dn | 39 | 5 |
| bc0 | 26 | 11 | misex2 | 25 | 18 | x7dn | 66 | 15 |
| bca | 26 | 46 | misg | 56 | 23 | x9dn | 27 | 7 |
| bcb | 26 | 39 | mish | 94 | 43 | xparc | 41 | 73 |
| bcc | 26 | 45 | misj | 35 | 14 | sequential | | |
| bcd | 26 | 38 | mlp8 | 16 | 16 | $s208_c$ | 18 | 9 |
| c8 | 28 | 18 | mlp9 | 18 | 18 | $s298_c$ | 17 | 20 |
| cc | 21 | 20 | mlp10 | 20 | 20 | $s344_c$ | 24 | 26 |
| chkn | 29 | 7 | mux | 21 | 1 | $s349_c$ | 24 | 26 |
| cht | 47 | 36 | my_adder | 33 | 17 | $s382_c$ | 24 | 27 |
| cm150a | 21 | 1 | nrm8 | 16 | 9 | $s400_c$ | 24 | 27 |
| comp | 32 | 3 | nrm9 | 18 | 10 | $s420_c$ | 34 | 17 |
| cordic | 23 | 2 | opa | 17 | 69 | $s444_c$ | 24 | 27 |
| count | 35 | 16 | pair | 173 | 137 | $s510_c$ | 25 | 13 |
| cps | 24 | 109 | pcle | 19 | 9 | $s526_c$ | 24 | 27 |
| dalu | 75 | 16 | pcler8 | 27 | 17 | $s641_c$ | 54 | 43 |
| des | 256 | 245 | pdc | 16 | 40 | $s713_c$ | 54 | 42 |
| dk48 | 15 | 17 | pm1 | 16 | 13 | $s820_c$ | 23 | 24 |
| duke2 | 22 | 29 | rckl | 32 | 7 | $s832_c$ | 23 | 24 |
| e64 | 65 | 65 | rdm16 | 16 | 16 | $s838_c$ | 66 | 33 |
| ex4 | 128 | 28 | rdm17 | 17 | 17 | $s1196_c$ | 32 | 32 |
| example2 | 85 | 66 | rdm18 | 18 | 18 | $s1423_c$ | 91 | 79 |
| exep | 30 | 63 | rot | 135 | 107 | $s5378_c$ | 214 | 228 |
| frg1 | 28 | 3 | rot16 | 16 | 9 | $s9234_c$ | 247 | 250 |
| frg2 | 143 | 139 | rot17 | 17 | 9 | $s13207_c$ | 700 | 790 |
| i1 | 25 | 16 | rot18 | 18 | 10 | $s15850_c$ | 611 | 684 |
| i2 | 201 | 1 | sct | 19 | 15 | $s38417_c$ | 1664 | 1742 |
| i3 | 132 | 6 | seq | 41 | 35 | $s38584_c$ | 1464 | 1730 |
| i4 | 192 | 6 | shift | 19 | 16 | | | |

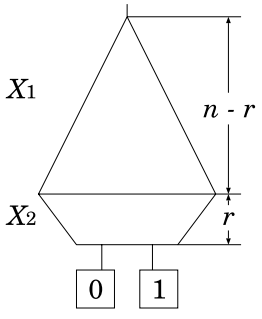



**Fig. A· 1** Partition of QRBDD.

This upper bound becomes the tightest when $r$ is the maximum integer satisfying $n - r \geq 2^r$ [16]. (Q.E.D.)

## Appendix C: Proof of Theorem 3.2

The proof of Theorem 3.2 also uses similar approach.

**(Proof of Theorem 3.2)** Since each node in a QRMDD($k$) has $2^k$ outgoing edges, the upper part of QRMDD($k$) is maximum when it is equivalent to a complete $2^k$-valued tree. Therefore, the upper part has at most

$$\frac{2^{sk}-1}{2^k-1}$$

nodes, where $s$ denotes the number of super variables in upper part. The lower part is maximum when all $i$-variable functions are realized for each level $i$ ($i = 1, 2, \ldots, u - s$), which have $2^k$-valued inputs and binary outputs. Note that $X_u$ may include dummy variables. Therefore, the number of nodes in a QRMDD($k$) is at most

$$\frac{2^{sk}-1}{2^k-1} + \sum_{i=1}^{u-s} 2^{2^{(ki-t)}}.$$

(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.

**Yukihiro Iguchi** was born in Tokyo, and received the B.E., M.E., and Ph.D. degree in electronic engineering from Meiji University, Kanagawa Japan, in 1982, 1984, and 1987, respectively. He is now an associate professor of Meiji University. His research interest includes logic design, switching theory, and reconfigurable systems. In 1996, he spent a year at Kyushu Institute of Technology.

**Munehiro Matsuura** was born on January 1, 1965 in Kitakyushu City, Japan. He studied at the Kyushu Institute of Technology from 1983 to 1989. He has been working as a Technical Assistant at the Kyushu Institute of Technology since 1991. He has implemented several logic design algorithms under the direction of Professor Tsutomu Sasao. His interests include decision diagrams and exclusive-OR based circuit design.