

Compact Representations of Logic Functions using Heterogeneous MDDs

Shinobu NAGAYAMA¹ and Tsutomu SASAO^{1,2}

¹Department of Computer Science and Electronics, Kyushu Institute of Technology
Iizuka 820-8502, Japan

²Center for Microelectronics Systems Kyushu Institute of Technology
Iizuka 820-8502, Japan

Abstract

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 as small memory size as the Free Binary Decision Diagrams (FBDDs). We minimized a large number of benchmark functions to show the compactness of heterogeneous MDDs.

1. Introduction

Embedded systems are widely used in vehicle control, consumer electronics, personal digital assistance (PDA), cellular phone and so on. These systems have a restriction on the memory size because of the need to reduce their cost, power consumption, and weight. Therefore, compact representations that allow high-speed function evaluation are important.

Reduced Ordered Binary Decision Diagrams (ROBDDs) [6] have been widely used to represent logic functions. In this paper, we propose representations of logic functions using Multi-valued Decision Diagrams (MDDs) that are called heterogeneous MDDs. Heterogeneous MDDs can represent logic functions with smaller memory size and shorter path length [18] than ROBDDs. Free Binary Decision Diagrams (FBDDs) [7, 8] can compactly represent logic functions. Our experimental results show that heterogeneous MDDs require memory size comparable to FBDDs, and can be minimized in shorter time.

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 for heterogeneous MDDs. Section 4 proposes an algorithm that minimizes memory size. And, Section 5 compares sizes of heterogeneous MDDs for many benchmark functions.

Proofs of theorems are available in <http://www.lsi-cad.com/Hetero-MDD/>.

2 Definitions

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

2.1 Representation of Logic Functions

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

Let $\{X\}$ denote the set of variables in X . If $\{X\} = \{X_1\} \cup \{X_2\} \cup \dots \cup \{X_u\}$ and $\{X_i\} \cap \{X_j\} = \emptyset (i \neq j)$, then (X_1, X_2, \dots, X_u) is a **partition** of X . An ordered set of variables X_i is called a **super variable**. If $|X_i| = k_i (i = 1, 2, \dots, u)$ and $k_1 + k_2 + \dots + k_u = n$, then a two-valued logic function $f(X)$ can be represented by the mapping $f(X_1, X_2, \dots, X_u): P_1 \times P_2 \times P_3 \times \dots \times P_u \rightarrow B$, where $P_i = \{0, 1, 2, \dots, 2^{k_i} - 1\}$ 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 $P_2 = \{0, 1, \dots, 7\}$. 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) [6], MDDs, and Reduced Ordered Multi-valued Decision Diagrams (ROMDDs) [11].

Definition 2.1 When $X = (x_1, x_2, \dots, x_n)$ is partitioned into (X_1, X_2, \dots, X_u) , an ROMDD representing a logic function $f(X)$ is called a **heterogeneous MDD**. Specifically, when $k_1 = k_2 = \dots = k_u$, 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 $k = k_1 = k_2 = \dots = k_u$. An MDD(k) represents a mapping $f : P^u \rightarrow B$, while a heterogeneous MDD represents a mapping $f : P_1 \times P_2 \times \dots \times P_u \rightarrow B$, where $P = \{0, 1, \dots, 2^k - 1\}$, $P_i = \{0, 1, \dots, 2^{k_i} - 1\}$, 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 2.2 In a decision diagram (DD), the **number of nodes in the DD**, denoted by $nodes(DD)$, includes only non-terminal nodes.

Definition 2.3 The **width of the DD with respect to X_i** , denoted by $width(DD, i)$, 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, \dots, X_u) is given by

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

Example 2.2 Consider the function:

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

Fig. 2.1(a), Fig. 2.1(b) and Fig. 2.2 represent the ROBDD, the MDD(2), and the heterogeneous MDDs for f , respectively. In Fig. 2.1(a), the solid lines and the dotted lines denote 1-edges and 0-edges, respectively. In Fig. 2.1(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. 2.2(a), $X_1 = (x_1, x_2, x_3)$ and $X_2 = (x_4)$. However, in Fig. 2.2(b), $X_1 = (x_1)$, $X_2 = (x_2, x_3, x_4)$.

(End of Example)

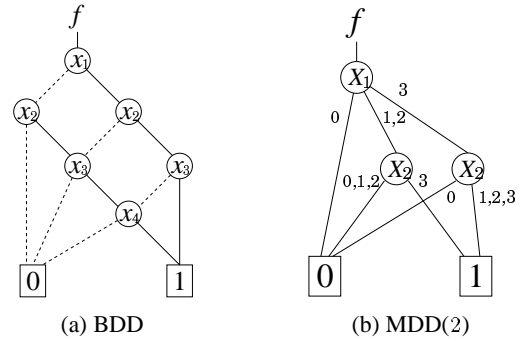


Figure 2.1. BDD and MDD(2)

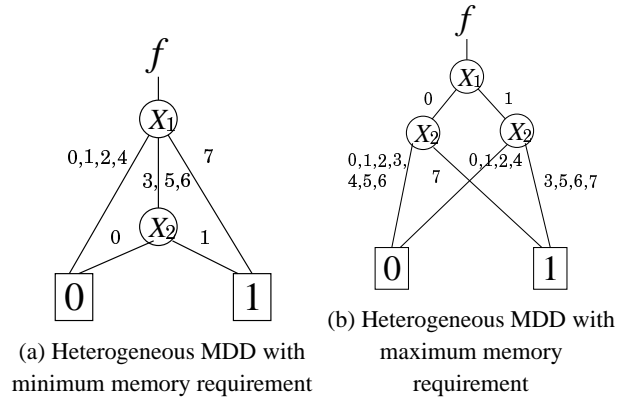


Figure 2.2. Heterogeneous MDDs

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 $F = (f_0, f_1, \dots, f_{m-1}) : B^n \rightarrow B^m$, where $B = \{1, 0\}$, and n and m denote the number of input and output variables, respectively. Several methods exist to represent multiple-output functions by using BDDs [15, 19, 20, 21]. In this paper, we use a Shared Binary Decision Diagram (SBDD) [19] to represent multiple-output functions. In the following, a BDD means a SBDD unless stated otherwise.

3 Memory Size for Heterogeneous MDDs

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

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 MDDs based on some criteria. In [18], we considered the average path length (APL) for heterogeneous MDDs. This section shows the memory size for heterogeneous MDDs.

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 refers the succeeding nodes. Since a node in a BDD has two pointers, the memory size needed to represent a BDD is given by

$$(2 + 1) \times nodes(BDD), \quad (3.1)$$

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

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

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. 2.1(a), 18; for the MDD(2) in Fig. 2.1(b), 15; for the heterogeneous MDD in Fig. 2.2(a), 12; and for the heterogeneous MDD in Fig. 2.2(b), 21. (End of Example)*

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

$$\begin{aligned} & (2^{k_i} + 1)width(\text{heterogeneous MDD}, i) \\ & \leq 3 \times \sum_{t=0}^{k_i-1} width(BDD, j + t). \end{aligned}$$

Theorem 3.2 *Consider a logic function that depends on n variables. Let $M_{min}(n)$ be the memory size needed to represent a heterogeneous MDD for the function. Then, the following relation holds:*

$$M_{min}(n) \geq 2.5n.$$

Corollary 3.1 *Consider a function that depends on n variables. Let M_{sub} be the memory size needed to represent a sub-graph of a heterogeneous MDD for the function:*

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

Then, the following relation holds:

$$M_{sub} \geq 2.5n_j,$$

where n_j denotes the number of binary variables present in the sub-graph.

Theorem 3.3 *Consider a logic function f that depends on n variables. Let $Mem(\text{heterogeneous MDD} : f)$ be the memory size needed to represent a heterogeneous MDD for f . If $n > 1$, then the following relation holds:*

$$Mem(\text{heterogeneous MDD} : f) \geq nodes(BDD) + 2.$$

Corollary 3.2 *Consider a logic function f that depends on n variables. Let $Mem_{sub}(\text{heterogeneous MDD} : f)$ be the memory size needed to represent a sub-graph of a heterogeneous MDD for f :*

$$\begin{aligned} Mem_{sub}(\text{heterogeneous MDD} : f) = \\ \sum_{i=j}^u width(\text{heterogeneous MDD}, i). \end{aligned}$$

If $n_j > 1$, then the following relation holds:

$$\begin{aligned} Mem_{sub}(\text{heterogeneous MDD} : f) \\ \geq \sum_{t=1}^{n_j} width(BDD, t) + 2, \end{aligned}$$

where n_j denotes the number of binary variables present in the sub-graph.

Lemma 3.2 *Consider the variables: $X^l = (x_1, x_2, \dots, x_{n_r})$, where $n_r < n$. When the widths of the BDD are given by*

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

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

Theorem 3.4 *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:*

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

where r is the largest integer satisfying the relation

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

4 Minimization of Heterogeneous MDDs

In this section, we formulate a minimization problem of heterogeneous MDDs, and present a minimization algorithm.

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 *Fig. 2.2(a) shows the heterogeneous MDD with the minimum memory for the function f , while Fig. 2.2(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 the following algorithm. Algorithm 4.1 shows a pseudo-code to solve Problem 4.1. This algorithm is based on dynamic programming. For simplicity, we assume that the variable order is x_1, x_2, \dots, x_n . This algorithm uses the index of the top variable for BDD as the argument.

Algorithm 4.1 *(Minimization of a heterogeneous MDD)*

```

1: minimize_memory (index  $i$ ) {
2:   if ( $i > n$ )
3:     return 0 ;
4:   if ( $x_i$  has been computed)
5:     return table[ $i$ ] ;
6:   min_mem = (memory for BDD) ;
7:   for( $l = 0; l \leq n - i; l++$ ) {
8:      $k = \mathbf{branch}(i, l)$  ;
9:     mdd_mem =  $(2^k + 1) \text{width}(\text{heterogeneous MDD}, j)$  ;
10:    if (mdd_mem > upper bounds)
11:      break ;
12:    next index  $i' = i + k$  ;
13:    mdd_mem += minimize_memory( $i'$ ) ;
14:    if (min_mem > mdd_mem) {
15:      min_mem = mdd_mem ;
16:      register the partition  $k$  ;
17:    }
18:  }
19:  table[ $i$ ] = min_mem ;
20:  return table[ $i$ ] ;
21:}
```

Algorithm 4.1 finds an optimum solution for Problem 4.1. In the 8th line in Algorithm 4.1, **branch** finds a k that makes the following ratio the minimum,

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

And, the 10th line uses **upper bounds**. In this part, Theorem 3.1 and Theorem 3.4 are used.

The time complexity for Algorithm 4.1 is $O(n^2)$. However, in many cases, the CPU time is proportional to n . The space complexity for Algorithm 4.1 is $O(n)$.

5 Experimental Results

5.1 Comparison with FBDDs

Table 5.1 compares the memory size needed for minimum heterogeneous MDDs with that for FBDDs. The numbers of nodes in OBDDs and FBDDs are taken from [23] and [7, 8], respectively. In Table 5.1, “MDD” denotes the minimum heterogeneous MDDs generated from OBDDs in [23]. Note that these DDs use complemented edges. The memory size needed for OBDDs and FBDDs are calculated using the formula (3.1). The column “Time” in Table 5.1 denotes the CPU time to obtain the optimum partition. We used the following environment:

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

In Table 5.1, the bottom row “Average of ratios” denotes the arithmetic average of the relative memory size, where the memory size needed for OBDD is set to 1.00. These results show that heterogeneous MDDs require comparable memory size to the FBDDs. And, Algorithm 4.1 obtains optimum solutions in a short computation time.

5.2 Comparison with BDDs and MDD(k)s

Table 5.2 compares heterogeneous MDDs with BDDs and MDD(k)s ($k = 2, 3, 4, 5$). Note that no complemented edges are used in these DDs. To obtain an average, 238 benchmark functions are used. In Table 5.2, “MDD” denotes the minimum heterogeneous MDDs obtained by Algorithm 4.1. The column “Size” shows the arithmetic averages of the relative memory size needed for each DD, where

Table 5.1. Memory sizes for heterogeneous MDDs and FBDDs

Function	#in	#out	<i>nodes(DDs)</i>		Memory Size			Time [msec]
			OBDD	FBDD	OBDD	FBDD	MDD	
C432	36	7	1063	1057	3189	3171	2910	0.01
C499	41	32	25865	25865	77595	77595	59739	20.00
C880	60	26	4052	2798	12156	8394	11934	0.01
C1908	33	25	5525	5047	16575	15141	13493	0.01
C2670	233	64	1771	1062	5313	3186	4805	0.01
C3540	50	22	23827	20999	71481	62997	65198	10.00
C5315	178	123	1718	1478	5154	4434	4855	0.01
C7552	207	107	2159	1594	6477	4782	6383	0.01
alu4	14	8	349	300	1047	900	855	0.01
apex1	45	45	1245	1177	3735	3531	3117	0.01
apex6	135	99	490	455	1470	1365	1437	0.01
cps	24	102	970	902	2910	2706	2568	0.01
dalu	75	16	688	649	2064	1947	1574	0.01
des	256	245	2944	2902	8832	8706	7589	0.01
frg2	143	139	961	920	2883	2760	2773	0.01
i3	132	6	132	132	396	396	332	0.01
i8	133	81	1275	1190	3825	3570	3825	0.01
i10	257	224	20659	18813	61977	56439	58165	0.01
k2	45	45	1245	1136	3735	3408	3119	0.01
too_large	38	3	318	286	954	858	859	0.01
vda	17	39	477	467	1431	1401	1088	0.01
Average of ratios					1.00	0.90	0.89	

the memory size needed for a BDD is set to 1.00. And, the column “APL” shows the arithmetic averages of the relative average path length. We use the method in [22] to calculate the APL.

Table 5.2. Comparison of heterogeneous MDDs, BDDs, and MDD(*k*)s

DDs	Size	APL
BDD	1.00	1.00
MDD	0.84	0.69
MDD(2)	1.08	0.69
MDD(3)	1.54	0.53
MDD(4)	2.39	0.50
MDD(5)	4.10	0.46

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.

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 algorithm for the memory size. 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 average path length in the BDDs; 2) Heterogeneous MDDs require comparable memory size to Free Binary Decision Diagrams (FBDDs); 3) The computation time to obtain the minimum heterogeneous MDDs is short.

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

In this paper, the variable order is fixed. To obtain the optimum heterogeneous MDDs considering both partitioning and ordering of the input variables, we need to improve the algorithms and heuristics.

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. Dr. Alan Mishchenko improved English paper.

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. Jurcska, L. Lavagno, A. Sangiovanni-Vincentelli, E. M. Sentovich, and K. Suzuki, "Synthesis of software programs for embedded control applications," *IEEE Trans. CAD*, 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. of 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," *Journal of the 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*, June 1985, pp. 663-698.
- [6] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, Vol. C-35, No. 8, pp. 677-691, Aug. 1986.
- [7] W. Günther and R. Drechsler, "Minimization of free BDDs," *Asia and South Pacific Design Automation Conference (ASP-DAC'99)*, pp.323-326, Jan. 18-21, 1999, Wanchai, Hong Kong.
- [8] W. Günther, "Minimization of free BDDs using evolutionary techniques," *International Workshop on Logic Synthesis 2000 (IWLS-2000)*, pp.167-172, May 31-June 2, 2000, Loguna Cliffs Marriott, Dana Point, CA.
- [9] 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)*, Jan. 26-28, Yokohama, Japan.
- [10] Y. Iguchi, T. Sasao, M. Matsuura, "Implementation of multiple-output functions using PQMDDs," *International Symposium on Multiple-Valued Logic*, pp.199-205, May 2000.
- [11] T. Kam, T. Villa, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Multi-valued decision diagrams: Theory and Applications," *Multiple-Valued Logic*, 1988, Vol. 4, No. 1-2, pp. 9-62, 1998.
- [12] C. Kim, L. Lavagno, and A. S-Vincentelli, "Free BDD-based software optimization techniques for embedded systems," *Design, Automation and Test in Europe (DATE2000)*, Paris, pp.14-19, March 2000.
- [13] H.-T. Liaw, and C.-S. Lin. "On the OBDD-representation of general Boolean function," *IEEE Transactions on Computers*, Vol. 4, No. 6, pp. 661-664, June 1992.
- [14] 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.
- [15] 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.
- [16] D. M Miller, "Multiple-valued logic design tools," *Proc. of International Symposium on Multiple Valued Logic*, pp. 2-11, May 1993.
- [17] 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 15-18, 2002.
- [18] S. Nagayama and T. Sasao, "Code generation for embedded systems using heterogeneous MDDs," *Synthesis And System Integration of Mixed Information technologies (SASIMI2003)*, Hiroshima, Japan, April 3-4, 2003 (accepted for publication).
- [19] T. Sasao and M. Fujita (ed.), *Representations of Discrete Functions*, Kluwer Academic Publishers 1996.
- [20] 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 29-31, 1996.
- [21] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.
- [22] T. Sasao, Y.Iguchi, and M. Matsuura, "Comparison of decision diagrams for multiple-output logic functions," *International Workshop on Logic and Synthesis*, New Orleans, Louisiana, June 4-7, 2002, pp.379-384.
- [23] F. Somenzi, "CUDD: CU Decision Diagram Package Release 2.3.1," University of Colorado at Boulder, 2001.
- [24] C. D. Thompson, "Area-Time complexity for VLSI," *Ann. Symp. on Theory of Computing*, May 1979.
- [25] S. Yang, *Logic synthesis and optimization benchmark user guide version 3.0*, MCNC, Jan. 1991.