2010-08

Systematic design method for two-variable numeric function generators using multiple-valued decision diagrams

Nagayama, Shinobu; Sasao, Tsutomu; Butler, Jon T.

S. Nagayama, T. Sasao and J. T. Butler, "Systematic design method for two-variable numeric function generators using multiple-valued decision diagrams," IEICE Transactions on Information and Systems, Vol. E93-D No. 8 pp. 2059-2067, Aug. 2010.
https://hdl.handle.net/10945/35832

PAPER Special Section on Multiple-Valued Logic and VLSI Computing

A Systematic Design Method for Two-Variable Numeric Function Generators Using Multiple-Valued Decision Diagrams*

Shinobu NAGAYAMA[†a)], **Tsutomu SASAO**[††b)], ***and*** **Jon T. BUTLER**[†††c)], ***Members***

SUMMARY This paper proposes a high-speed architecture to realize two-variable numeric functions. It represents the given function as an edge-valued multiple-valued decision diagram (EVMDD), and shows a systematic design method based on the EVMDD. To achieve a design, we characterize a numeric function f by the values of l and p for which f is an l-restricted Mp-monotone increasing function. Here, l is a measure of subfunctions of f and p is a measure of the rate at which f increases with an increase in the dependent variable. For the special case of an EVMDD, the EVBDD, we show an upper bound on the number of nodes needed to realize an l-restricted Mp-monotone increasing function. Experimental results show that all of the two-variable numeric functions considered in this paper can be converted into an l-restricted Mp-monotone increasing function with $p = 1$ or 3. Thus, they can be compactly realized by EVBDDs. Since EVMDDs have shorter paths and smaller memory size than EVBDDs, EVMDDs can produce fast and compact NFGs.

key words: *two-variable numeric function generators (NFGs), edge-valued multiple-valued decision diagrams (EVMDDs), edge-valued binary decision diagrams (EVBDDs), graph-based representation of numeric functions, programmable memory-based architecture*

1. Introduction

Numeric functions have wide applications including computer graphics, direct digital frequency synthesizers [5], and digital signal processing. Various design methods for numeric function generators (NFGs) have been developed [18]. However, most existing methods are intended for one-variable numeric functions [7], [16], [21], [25], [29]–[31], and only a few methods have been reported for specific multi-variable numeric functions [9], [10], [34]. Thus, different numeric functions require different methods. As far as we know, no systematic design method for generic multi-variable numeric functions has been presented.

A straightforward design method for an arbitrary multi-variable function is to use a single memory. This method produces a fast NFG, but requires a 2^{kn}-word memory to realize a k-variable function with n bits for each variable. Thus, even for a computation with a small number of bits, this method is impractical because of large memory size.

To produce practical NFGs, we consider a design method using decision diagrams (DDs) for two-variable NFGs. DDs, such as binary DDs (BDDs), can compactly realize various functions [8], [17], [35]–[37]. However, DDs are not able to represent all classes of the functions compactly. Thus, choosing a DD appropriate to a given class of functions is important. Although DDs suitable for one-variable numeric functions have been presented [21], [23], [29], [33], as far as we know, no study on graph-based representations for multi-variable numeric functions has been reported.

First, we present a DD appropriate for two-variable numeric functions. And then, we propose a design method and an architecture for two-variable NFGs using this DD. To analyze complexities for two-variable numeric functions, we introduce a new class of integer-valued functions, l-restricted Mp-monotone increasing functions. We derive an upper bound on the number of nodes in an edge-valued BDD (EVBDD) for this type of function. Theoretical analysis and experimental results show that edge-valued multiple-valued DDs (EVMDDs) can compactly represent both one- and two-variable numeric functions, and our NFGs using EVMDDs can compactly realize such functions with the same architecture.

This paper is organized as follows: Section 2 introduces a fixed-point representation to convert a real-valued numeric function into an integer-valued function. Section 3 considers representations of two-variable numeric functions using DDs. It introduces the l-restricted Mp-monotone increasing function, and derives an upper bound on the number of nodes in an EVBDD for this type of function. Section 4 presents an architecture and a design method for NFGs based on EVMDDs. Experimental results using an FPGA are also presented. Section 5 concludes the paper.

2. Preliminaries

2.1 Number Representation and Precision

Definition 1: Let $B = \{0, 1\}$, $\mathbb{Z}$ be the set of the integers, and $\mathbb{R}$ be the set of the real numbers. A logic function is $B^n \to B^m$. An integer-valued function is $B^n \to \mathbb{Z}$. A one-variable numeric (real) function is $\mathbb{R} \to \mathbb{R}$. And, a two-variable numeric function is $\mathbb{R} \times \mathbb{R} \to \mathbb{R}$.

Manuscript received November 10, 2009.
Manuscript revised March 14, 2010.

[†]The author is with the Department of Computer and Network Engineering, Hiroshima City University, Hiroshima-shi, 731–3194 Japan.
[††]The author is with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka-shi, 820–8502 Japan.
[†††]The author is with the Department of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA 93943–5121 USA.

*This paper is an extension of [22].

a) E-mail: nagayama@ieee.org
b) E-mail: sasao@cse.kyutech.ac.jp
c) E-mail: jon_butler@msn.com

DOI: 10.1587/transinf.E93.D.2059

Definition 2: A numeric function generator (NFG) is a logic circuit that computes approximate values for a numeric function. A one-variable NFG is a logic circuit for a one-variable numeric function $f(x)$, whose input is x, and output is an approximate value for $f(x)$. A two-variable NFG is a logic circuit for a two-variable numeric function $f(x, y)$, whose inputs are x and y, and output is an approximate value for $f(x, y)$.

Definition 3: The binary fixed-point representation of a number is denoted by

$$X = (x_{n\_int-1}\ x_{n\_int-2}\ \ldots\ x_1\ x_0.\ x_{-1}\ x_{-2}\ \ldots\ x_{-n\_frac})_2,$$

where $x_i \in \{0, 1\}$, n_int is the number of bits for the integer part, and n_frac is the number of bits for the fractional part of X. This is the two's complement representation: $X = -2^{n\_int-1}x_{n\_int-1} + 2^{n\_int-2}x_{n\_int-2} + \ldots + 2^{-n\_frac}x_{-n\_frac}$. To distinguish the set of binary variables from the numeric value X, we use brackets { }. Specifically, $\{X\}$ denotes the set of binary variables x_i.

Definition 4: Precision is the total number of bits in a binary fixed-point representation. Specially, n-bit precision specifies that n bits are used to represent the number; that is, $n = n\_int + n\_frac$. In this paper, an n-bit precision function $f(X, Y)$ means that both of the input variables X and Y have n-bit precision. The number of fractional bits for function values is n. This is because the range of a function can be different than its domain. Thus, function values have (n_int+n)-bit precision, where n_int is the number of integer bits for function values.

We can convert an n-bit precision two-variable numeric function into a $2n$-input m-output logic function, where $m = n\_int + n$. The logic function can be converted into an integer-valued function by considering binary vectors as integers. That is, we can convert an n-bit precision two-variable numeric function into an integer-valued function: $B^{2n} \rightarrow P_m$, where $P_m = \{0, 1, \ldots, 2^m - 1\}$. In this paper, two-variable numeric functions are converted into integer-valued functions in this way. And, in the text that follows, we assume that x_0 and y_0 denote the least significant bits in the fixed-point representations of X and Y, respectively.

Example 1: Table 1 (a) is the function table for the Euclidean norm function $\sqrt{X^2 + Y^2}$ for a two-dimensional vector from $(0, 0)$ to (X, Y). The 2-bit precision fixed-point representation of this function is the logic function $f_b(X, Y)$ in Table 1 (b). By converting output vectors into integers, we have the integer-valued function $f(X, Y)$ of $f_b(X, Y)$ in Table 1 (c). In this paper, the 2-bit precision 2-D norm function denotes the integer-valued function $f(X, Y)$ in Table 1 (c).
(End of Example)

Table 1 Tables for 2-bit precision 2-D norm function.

(a) Table for 2-D norm.

X	Y	Norm
0.00	0.00	0.00
0.00	0.25	0.25
0.00	0.50	0.50
0.00	0.75	0.75
0.25	0.00	0.25
0.25	0.25	0.35
0.25	0.50	0.56
0.25	0.75	0.79
0.50	0.00	0.50
0.50	0.25	0.56
0.50	0.50	0.71
0.50	0.75	0.90
0.75	0.00	0.75
0.75	0.25	0.79
0.75	0.50	0.90
0.75	0.75	1.06

(b) Table for $f_b(X, Y)$.

X	Y	f_b
0.00	0.00	0.00
0.00	0.01	0.01
0.00	0.10	0.10
0.00	0.11	0.11
0.01	0.00	0.01
0.01	0.01	0.01
0.01	0.10	0.10
0.01	0.11	0.11
0.10	0.00	0.10
0.10	0.01	0.10
0.10	0.10	0.11
0.10	0.11	1.00
0.11	0.00	0.11
0.11	0.01	0.11
0.11	0.10	1.00
0.11	0.11	1.00

(c) Table for $f(X, Y)$.

X	Y	f
00	00	0
00	01	1
00	10	2
00	11	3
01	00	1
01	01	1
01	10	2
01	11	3
10	00	2
10	01	2
10	10	3
10	11	4
11	00	3
11	01	3
11	10	4
11	11	4

2.2 Decision Diagrams

This subsection summarizes the DDs used in this paper. For more detail on definitions and reduction rules, see [8], [27], [37].

Definition 5: A multi-terminal binary decision diagram (MTBDD) [6] is an extension of a BDD [3], [17], and represents an integer-valued function. In the MTBDD, the terminal nodes are labeled by integers.

Definition 6: A binary moment diagram (BMD) [4] is a rooted directed acyclic graph (DAG) representing an integer-valued function. The BMD is obtained by repeatedly applying the arithmetic transform expansion $f = f_0 + x_i(f_1 - f_0)$ to the integer-valued function, where $f_0 = f(x_i = 0)$, and $f_1 = f(x_i = 1)$. The BMD consists of terminal nodes representing the arithmetic coefficients, and non-terminal nodes representing the arithmetic transform expansions. Each non-terminal node has two edges corresponding to two terms: f_0 and $x_i(f_1 - f_0)$ in the arithmetic transform expansion.

Definition 7: An edge-valued BDD (EVBDD) [14], [15] is a variant of a BDD, and represents an integer-valued function. The EVBDD is obtained by repeatedly applying the expansion $f = \overline{x_i}f_0 + x_i(f_1' + \alpha)$ to the integer-valued function, where $f_0 = f(x_i = 0)$, $f_1' + \alpha = f_1 = f(x_i = 1)$, and α is the constant term of f_1. The EVBDD consists of only one terminal node representing 0 and non-terminal nodes with 1-edges having integer weights α. In the EVBDD, 0-edges always have zero weights. The incoming edge to the root node can have a non-zero weight. In a reduced EVBDD, each node represents a distinct sub-function.

Definition 8: For an n-bit precision number X, if $\{X\} = \{X_u\} \cup \{X_{u-1}\} \cup \ldots \cup \{X_1\}$, $\{X_i\} \neq \emptyset$, and $\{X_i\} \cap \{X_j\} = \emptyset\ (i \neq j)$, then $\{X_u, X_{u-1}, \ldots, X_1\}$ is a partition of X. Each X_i forms a super variable. Let $|X_i| = k_i$ and $k_u + k_{u-1} + \ldots + k_1 = n$. Then, by considering each super variable as a multiple-valued variable, an integer-valued function $f(X) : B^n \rightarrow \mathbb{Z}$ can be converted into a multiple-valued input integer function $f(X_u, X_{u-1}, \ldots, X_1) : P_u \times P_{u-1} \times \ldots \times P_1 \rightarrow \mathbb{Z}$, where $P_i = \{0, 1, 2, \ldots, 2^{k_i} - 1\}$.

Definition 9: An edge-valued multiple-valued decision diagram (EVMDD) [21] is an extension of an MDD [13], and represents a multiple-valued input integer function. It consists of one terminal node representing 0 and non-terminal nodes with edges having integer weights, and 0-edges always have zero weights. As shown in Fig. 1, an EVMDD is

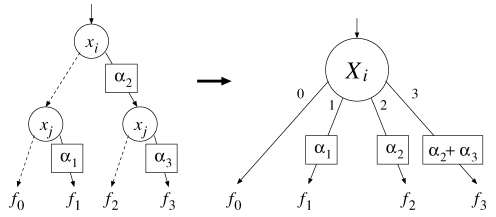
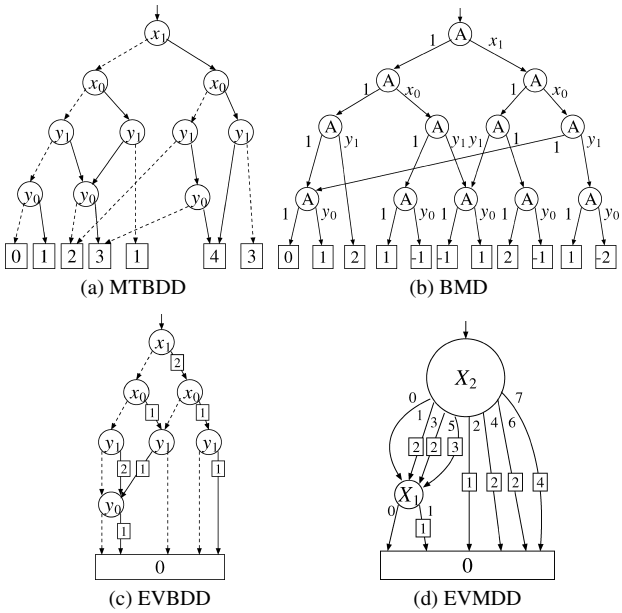


Fig. 1 Conversion of EVBDD nodes into an EVMDD node.



(a) MTBDD (b) BMD (c) EVBDD (d) EVMDD

Fig. 2 Four types of DDs for the 2-bit precision 2-D norm function.

obtained by merging non-terminal nodes in an EVBDD according to the partition of X. In an EVMDD, each multiple-valued variable may have a different domain.

Example 2: Figure 2 (a), (b), (c), and (d) show the MTBDD, the BMD, the EVBDD, and the EVMDD of the 2-bit precision 2-D norm function in Table 1 (c). For readability of the figures, several terminal nodes are not shared. In Fig. 2 (a) and (c), dashed lines and solid lines denote 0-edges and 1-edges, respectively. Note that the EVBDD has weighted 1-edges. In Fig. 2 (b), 'A' in a circle denotes the arithmetic transform expansion. And, in Fig. 2 (d), the set of binary variables $\{X\} \cup \{Y\}$ is partitioned into $\{X_2\} = \{x_1, x_0, y_1\}$ and $\{X_1\} = \{y_0\}$. To obtain the function value 3 for $X = (10)_2$ and $Y = (10)_2$, in the MTBDD, we traverse the MTBDD from the root node to a terminal node according to the input values, and obtain the function value from the terminal node. In the BMD, we obtain the function value by computing the arithmetic transform expansion $f = f_0 + x_i(f_1 - f_0)$ recursively at each non-terminal node. And, in the EVBDD and the EVMDD, we obtain the function value as the sum of the weights for the edges traversed from the root node to the terminal node. Note that we traverse the EVMDD using $X_2 = 5$ and $X_1 = 0$. (End of Example)

3. Graph-Based Representations of Two-Variable Numeric Functions

This section introduces an l-restricted Mp-monotone increasing function, and derives an upper bound on the number of nodes in an EVBDD for the l-restricted Mp-monotone increasing function. Experimental results in this section show that EVBDDs for two-variable numeric functions are more compact than MTBDDs and BMDs.

3.1 l-Restricted Mp-Monotone Increasing Functions

Definition 10: An n-bit precision integer-valued function $f(X)$ such that $0 \le f(X+1) - f(X) \le p$ and $f(0) = 0$ is a totally Mp-monotone increasing function (or simply, Mp-monotone increasing function). Here, $X + 1$ is the binary representation of the independent variable. That is, for an Mp-monotone increasing function $f(X)$, $f(0) = 0$, and the increment of X by one increases the value of $f(X)$ by at most p.

Adding 1 as in $X + 1$ is simply incrementing the standard binary number X. It should not be confused with adding 1 to a real-valued variable x.

Definition 11: An n-bit precision integer-valued function $f(X)$ is an l-restricted Mp-monotone increasing function when, for $1 \le l < n$, all the l-bit precision sub-functions $g(X_l)$ of f are Mp-monotone increasing functions, where $\{X\} = \{x_{n-1}, x_{n-2}, \ldots, x_0\}$, $\{X_l\} = \{x_{l-1}, x_{l-2}, \ldots, x_0\}$, and $g(X_l) = f(\vec{a}, X_l)$, for all assignments $\vec{a}$ of values to $(x_{n-1}\ x_{n-2}\ \ldots\ x_l)_2$.

Theorem 1: For an n-bit precision l-restricted Mp-monotone increasing function $f(X)$, the number of nodes in the EVBDD is at most

$$2^{n-l} + \sum_{i=1}^{l} (p+1)^{2^i - 1} - l, \tag{1}$$

where l is the largest integer satisfying $2^{n-l} \ge (p+1)^{2^l-1}$, and the variable order of the EVBDD is $x_{n-1}, x_{n-2}, \ldots, x_0$ (from the root node to the terminal node).

Proof: See the appendix.

Theorem 1 also holds for EVMDDs because an EVMDD is obtained by merging non-terminal nodes in an EVBDD. Note that the upper bound for l-restricted Mp-monotone increasing functions shown in Theorem 1 is equal to the upper bound for *totally* Mp-monotone increasing functions shown in [21]. This upper bound is much smaller than the worst-case upper bound, 2^n, which is reached by EVBDDs for power functions and polynomial functions [23].

Example 3: Consider a 16-bit precision l-restricted Mp-monotone increasing function. If $p = 1$, then we have $l = 3$, and the upper bound given by (1) is 8,327. If $p = 3$, then $l = 2$, and the upper bound is 16,450. (End of Example)

Definition 12: An n-bit precision integer-valued function $f(X)$ is an extended l-restricted Mp-monotone increasing function when, for $1 \leq l < n$, all the l-bit precision sub-functions of f are Mp-monotone increasing functions $g(X_l)$ or represented by $g(X_l) + b$, where b is an integer, $\{X\} = \{x_{n-1}, x_{n-2}, \ldots, x_0\}$, $\{X_l\} = \{x_{l-1}, x_{l-2}, \ldots, x_0\}$, and sub-functions are $f(\vec{a}, X_l)$, for all assignments $\vec{a}$ to $(x_{n-1}\ x_{n-2}\ \ldots\ x_l)_2$.

Lemma 1: Let $f(X)$ be an extended l-restricted Mp-monotone increasing function. For any integer l' satisfying $1 \leq l' \leq l$, $f(X)$ is an extended l'-restricted Mp-monotone increasing function.
Proof: This follows from Definition 12. ∎

Lemma 2: Let $f(X)$ be an l-restricted Mp-monotone increasing function, and let $g(X)$ be an extended l-restricted Mp-monotone increasing function that is obtained by adding constant values to the l-bit precision sub-functions of f. Then, the EVBDDs for $f(X)$ and $g(X)$ have the same number of nodes.
Proof: See the appendix.

Corollary 1: Let $f(X)$ be an extended l-restricted Mp-monotone increasing function, and let $g(X)$ be an affine transformation of f: $g(X) = af(X) + b$, where a and b are integers. Then, the EVBDDs for $f(X)$ and $g(X)$ have the same number of nodes.

EVBDDs can compactly represent not only l-restricted Mp-monotone increasing functions, but also their extended classes of functions. This property is helpful to compactly represent various two-variable numeric functions.

3.2 Two-Variable Numeric Functions

As shown in Sect. 2, n-bit precision two-variable numeric functions can be converted into $2n$-bit precision integer-valued functions. That is, n-bit precision two-variable functions $f(X, Y)$ can be converted into $2n$-bit precision *one-variable* functions $f(Z)$, where

$$Z = 2^n X + Y = (x_{n-1}\ x_{n-2}\ \ldots\ x_0\ y_{n-1}\ y_{n-2}\ \ldots\ y_0)_2.$$

When $f(Z)$ is an l-restricted Mp-monotone increasing function for the largest integer l satisfying $2^{2n-l} \geq (p+1)^{2^l-1}$, Theorem 1 gives the upper bound on the number of nodes in an EVBDD for $f(X, Y)$.

Example 4: As shown in Table 2 (a), the 2-bit precision 2-D norm function $\sqrt{X^2 + Y^2}$ can be converted into an extended 2-restricted M1-monotone increasing function $f(Z)$.

Table 2 Function tables for 2-bit precision two-variable functions.

(a) Table for 2-D norm.

Y \ X	0	1	2	3
0	0	1	2	3
1	1	1	2	3
2	2	2	3	4
3	3	3	4	4

(b) Table for $\frac{X}{Y+1}$.

Y \ X	0	1	2	3
0	0	1	2	3
1	0	1	2	2
2	0	1	1	2
3	0	1	1	2

(c) Table for $g(Z)$.

y_1y_0 \ x_1x_0	00	01	10	11
00	0	−1	−2	−3
01	0	−1	−2	−2
10	0	−1	−1	−2
11	0	−1	−1	−2

Note that, in the table, values increase by at most one for each column. Similarly, the 2-bit precision two-variable function $\frac{X}{Y+1}$ shown in Table 2 (b) can be converted into an affine transformation of the extended 2-restricted M1-monotone increasing function $g(Z)$ shown in Table 2 (c): $-1 \times g(Z)$. (End of Example)

We now show two-variable numeric functions that can be converted into integer-valued functions of classes discussed in Sect. 3.1. As a result, their EVBDDs are small.

Lemma 3: Let $h(Y)$ be an n-bit precision Mp-monotone increasing function. Then, for an arbitrary one-variable function $g(X)$, a two-variable function $f(X, Y) = g(X) + h(Y)$ is an extended n-restricted Mp-monotone increasing function.
Proof: See the appendix.

Lemma 4: Let $h(Y)$ be an n-bit precision Mp-monotone increasing function. Then, for an arbitrary one-variable function $g(X)$, the two-variable function $f(X, Y) = g(X) - h(Y)$ can be converted into an affine transformation of an extended n-restricted Mp-monotone increasing function.
Proof: See the appendix.

Lemma 5: Let $h(Y)$ be an n-bit precision Mp-monotone increasing function, and let $g(X)$ be a real function satisfying $0 \leq g(X) \leq 1$. Then, an n-bit precision two-variable function $f(X, Y) = g(X) \cdot h(Y)$ is an n-restricted Mp-monotone increasing function.
Proof: See the appendix.

In Lemma 5, if the range of $g(X)$ is large, then the EVBDD can be large. For example, the n-bit multiplier requires $O(2^n)$ nodes [36].

Lemma 6: Let $h(Y)$ be an affine transformation of an n-bit precision Mp-monotone increasing function, and let $g(X)$ be a real function satisfying $0 \leq g(X) \leq 1$. Then, an n-bit precision two-variable function $f(X, Y) = g(X) \cdot h(Y)$ can be converted into an affine transformation of an extended n-restricted Mp-monotone increasing function.
Proof: See the appendix.

Example 5: 2-bit precision function $\frac{1}{Y+1}$ is an affine transformation of an M1-monotone increasing function [21]. As shown in Example 4, $f(X, Y) = \frac{X}{Y+1}$ can be converted into an affine transformation of an extended 2-restricted M1-monotone increasing function. (End of Example)

Since many common one-variable numeric functions can be converted into Mp-monotone increasing functions [23], many two-variable numeric functions obtained by four arithmetic operations of them can be converted into extended l-restricted Mp-monotone increasing functions as shown in the above lemmas.

In the following, we show that various two-variable numeric functions, as well as the above functions, can be converted into extended l-restricted Mp-monotone increasing functions. Table 3 compares the numbers of nodes in MTBDDs, BMDs, and EVBDDs for certain 8-bit precision

Table 3 Numbers of nodes in MTBDDs, BMDs, and EVBDDs for 8-bit precision two-variable numeric functions.

Numeric functions	Type of function	Number of nodes			R_1	R_2
		MTBDD	BMD	EVBDD		
$\sqrt{X^2+Y^2}$	M1	12,969	25,084	2,566	20	10
$\arctan(\frac{X}{Y+1})$	M1$^+$	8,997	26,158	3,134	35	12
$\ln(X+1)\sin(Y)$	M1	9,776	25,994	3,444	35	13
$\sqrt{X}\sin(Y)$	M1	11,543	26,542	3,483	30	13
$\sin(\sqrt{X^2+Y^2})$	M1	11,521	27,858	4,013	35	14
$\sin(XY)$	M1	11,282	21,746	3,789	34	17
$X/(Y+1)$	M1$^+$	9,664	25,878	3,162	33	12
$XY/\sqrt{X^2+Y^2}$	M1	9,325	23,634	2,269	24	10
WaveRings	M3$^+$	17,423	27,691	5,047	29	18
Average		11,389	25,621	3,434	30	13

Domain of the functions is $0 \le X < 1$ and $0 \le Y < 1$.
Number of fractional bits for function values is 8.
Mp^+: the function is an affine transformation of an extended 8-restricted Mp-monotone increasing function.
R_1 = (EVBDD) / (MTBDD) × 100. R_2 = (EVBDD) / (BMD) × 100.
Variable orders of DDs are produced by the sifting algorithm [26].

two-variable numeric functions [2]. *WaveRings* in the table is

$$WaveRings = \frac{\cos\left(\sqrt{X^2+Y^2}\right)}{\sqrt{X^2+Y^2+0.25}}.$$

In the column labeled "Type of function" of Table 3, Mp denotes an extended 8-restricted Mp-monotone increasing function, while Mp^+ denotes an affine transformation of an extended 8-restricted Mp-monotone increasing function.

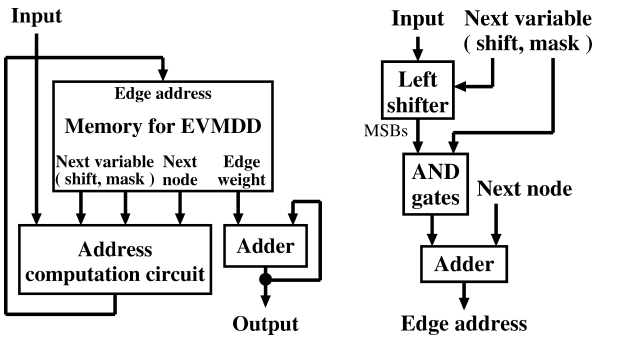
Two-variable numeric functions whose range changes smoothly on a given domain can be converted into extended l-restricted Mp-monotone increasing functions with small p. As shown in Theorem 1, such functions have small EVBDDs. In fact, the two-variable numeric functions in Table 3 are converted into 8-restricted M1 or M3-monotone increasing functions, and EVBDDs have many fewer nodes than MTBDDs and BMDs. Since non-terminal nodes of EVBDD have weighted 1-edges, a non-terminal node of EVBDD requires larger memory size than a non-terminal node of MTBDD and BMD. However, the increase due to the weighted edges is negligible, because EVBDDs have many fewer non-terminal nodes than MTBDDs and BMDs [21].

As shown in [21], by converting EVBDDs into EVMDDs, we can often reduce memory size and path length of decision diagrams. In the next section, we present a design method that takes advantage of EVMDDs.

4. Two-Variable NFGs Based on EVMDDs

4.1 Architecture for NFGs

In DDs based on the Shannon expansion, function values can be obtained by traversing the DDs from the root node to a terminal node [11], [12]. In EVBDDs and EVMDDs, function values can be obtained as the sum of the weights for traversed edges. Figure 3 shows an NFG based on an EVMDD. It consists of a memory to store an EVMDD, an address computation circuit to traverse the EVMDD, and an accumulator to compute the sum of edge weights. Note that, for readability, registers, circuits for initialization, and some signals are omitted from Fig. 3.



(a) Architecture for NFGs. (b) Address computation circuit.

Fig. 3 Architecture for NFGs based on EVMDDs.

In Fig. 3 (a), the block labeled "Memory for EVMDD" stores data for edges in an EVMDD. Data for an edge consists of a pointer to the next node, data for the variable of the next node, and an edge weight. From the memory, a pointer to the next node and data for the next variable are read and fed to the address computation circuit. And, an edge weight is fed to the accumulator. The address computation circuit produces an address of the next edge from an address of the node and a value of the input variable.

Figure 3 (b) shows the address computation circuit. Data for the next variable consist of shift data and mask data. A value of the corresponding input variable is retrieved by the left shifter and the AND gates. And, the value is added to the address of next node to generate the edge address. This circuit just selects an edge to be traversed. Thus, we could use a multiplexer. However, it is inefficient because, in EVMDDs, each input variable can have a different domain. Note that EVBDDs require neither the AND gates nor the adder because in EVBDDs, all input variables are binary.

Since the circuit shown in Fig. 3 just traverses an EVMDD and computes the sum of edge weights, it can evaluate both one- and two-variable functions with the same architecture.

4.2 Design Method for NFGs Using EVMDDs

For a given numeric function, its domain, and precision, we can systematically design the circuit in Fig. 3. First, convert a given numeric function into an n-bit precision integer-valued function. Next represent the integer-valued function using an EVMDD, and finally generate HDL code for the circuit in Fig. 3 from the EVMDD. Since our NFG directly realizes the function table, it is more accurate than existing NFGs using polynomial approximation [7], [16], [25], [30], [31].

To generate memory data for an NFG in Fig. 3, we first assign an address to each edge in an EVMDD. For each non-

terminal node, we assign addresses to edges in ascending order from 0-edges. Thus, addresses assigned to 0-edges correspond to addresses of non-terminal nodes. Next, we generate shift data and mask data of each edge. For an EVMDD for $f(X_u, X_{u-1}, \ldots, X_1)$, shift data and mask data of an edge are computed as follows:

$$\text{shift data} = \sum_{j=i}^{u-1} k_j, \qquad \text{mask data} = 2^{k_i} - 1, \text{ and}$$

$$\text{bit size for mask} = \max_{1 \le j \le u}(k_j),$$

where the edge points to a node representing X_i, variable order of the EVMDD is $X_u, X_{u-1}, \ldots, X_1$ from the root, and $k_j = |X_j|$ $(j = 1, 2, \ldots, u)$.

Example 6: Table 4 shows memory data and initial values for the NFG produced from the EVMDD in Fig. 2 (d). This example shows how to compute the function value for $X = (10)_2$ and $Y = (10)_2$ using Table 4.

First, the address computation circuit produces an edge address from the initial values. Since the initial shift data is 0, the bitwise AND between the most significant 3 bits of input variable $(x_1\ x_0\ y_1)_2 = (101)_2$ and the initial mask data $(111)_2$ is computed to produce $(101)_2$. Adding the initial address of node 0 to the result of bitwise AND yields the first edge address $(101)_2 = 5$, which corresponds to the edge 5 of the root node in Fig. 2 (d).

Next, data for address 5 is read from the memory and fed to the address computation circuit and the accumulator. The accumulator obtains the sum of the edge weight 3 given by the memory and the initial edge weight 0. In the address computation circuit, the value of input variable is $(x_0\ y_1\ y_0)_2 = (010)_2$, which is shifted 1-bit left, and the bitwise AND is performed with the mask data $(001)_2$. Adding the result of bitwise AND 0 to the address of next node 8 yields the second edge address 8, which corresponds to the edge 0 of the node for X_1 in Fig. 2 (d).

Since the mask data 0 at the address 8 means arrival at the terminal node, adding the edge weight 0 to the previous sum of edge weights 3 yields the function value 3. (End of Example)

Table 4 Memory data for the NFG for 2-bit precision 2-D norm function.

Edge address	Shift data	Mask data (binary)	Address of next node	Edge weight
0	1	001	8	0
1	1	001	8	2
2	0	000	0	1
3	1	001	8	2
4	0	000	0	2
5	1	001	8	3
6	0	000	0	3
7	0	000	0	4
8	0	000	0	0
9	0	000	0	1

Initial values
Shift data: 0 Mask data: 111
Address of node: 0 Edge weight: 0

Memory size and delay time of our NFG depend largely on memory size and path length of EVMDD. Therefore, the memory minimization algorithm and the APL minimization algorithm for MDDs [19], [20] are useful for producing fast and compact NFGs.

4.3 FPGA Implementation Results

We implemented the NFGs in Fig. 3 for the two-variable functions shown in Table 3 on an Altera FPGA (EP1S25F672C8). To show the effectiveness of MDDs, NFGs for EVBDDs and EVMDDs were compared. Table 5 shows the results.

EVBDD-based NFGs can achieve higher operating frequency than EVMDD-based NFGs, because the EVBDD-based NFGs require neither AND gates nor an adder for the address computation circuit. However, as for the time to obtain the function value (i.e., latency), EVMDD-based

Table 5 FPGA implementation of 8-bit precision two-variable numeric functions.

FPGA device: Altera Stratix EP1S25F672C8
Logic synthesis tool: Altera QuartusII 7.1 (speed optimization, timing requirement of 100 MHz)

Numeric functions	EVBDD					EVMDD					Ratio [%]	
	LE	Mem [bits]	Freq. [MHz]	LPL	Delay [nsec]	LE	Mem [bits]	Freq. [MHz]	LPL	Delay [nsec]	Mem	Delay
$\sqrt{X^2+Y^2}$	367	155,736	80	16	201	96	103,080	66	5	76	66	38
$\arctan(\frac{X}{Y+1})$	256	132,174	79	16	204	94	88,760	66	6	91	67	44
$\ln(X+1)\sin(Y)$	169	142,640	79	16	202	91	86,256	66	5	76	60	37
$\sqrt{X}\sin(Y)$	317	157,080	81	16	198	91	91,404	67	5	74	58	38
$\sin(\sqrt{X^2+Y^2})$	204	184,968	69	16	231	93	101,916	65	5	77	55	33
$\sin(XY)$	365	151,520	81	16	197	91	90,828	67	5	74	60	38
$X/(Y+1)$	338	145,782	82	16	195	91	88,236	67	5	75	61	38
$XY/\sqrt{X^2+Y^2}$	269	145,200	80	16	200	94	99,826	67	5	75	69	37
WaveRings	114	235,934	68	16	234	101	144,892	65	5	77	61	33
Average	267	161,226	78	16	207	94	99,466	66	5	77	62	37

LE: Number of logic elements Mem: Memory size Freq.: Operating frequency
LPL: Longest path length of DDs Delay: maximum delay time = LPL / Freq.
Ratio for Mem = Mem for EVMDD / Mem for EVBDD×100
Ratio for Delay = Delay for EVMDD / Delay for EVBDD×100

Table 6 Performance comparison with CORDIC for 16-bit sin(X).

FPGA: Xilinx Virtex-II XC2V1000-6 Logic synthesis tool: Synplify Premier Ver. 8.5		
NFGs	Freq. [MHz]	Delay [nsec]
CORDIC (RTL) [1]	102	157
CORDIC (structural) [1]	222	72
EVBDD-based	193	83
EVMDD-based	141	43

CORDIC (RTL): The circuit is synthesized from RTL.
CORDIC (structural): The circuit is manually implemented to make it suitable for the FPGA structure.

NFGs are shorter than EVBDD-based NFGs, because the path length of EVMDD is shorter than that of EVBDD. The maximum delay time needed to obtain a function value for EVMDD-based NFGs is, on the average, only 37% of that for EVBDD-based NFGs. And, EVMDD-based NFGs require, on the average, only 62% of the memory size needed for EVBDD-based NFGs. These results show that EVMDDs produce fast and compact two-variable NFGs.

Two-variable functions are often designed using a combination of one-variable NFGs, multipliers, and adders. For example, the compound function $\sin(\sqrt{X^2+Y^2})$ can be designed using two circuits realizing a^2, an adder, a square root circuit, and a sine function circuit. But, it can produce a slow implementation due to long path delays. For such a compound function, two-variable NFGs that can directly realize two-variable functions are much faster [24]. To compare the EVMDD-based NFGs with the previous two-variable NFGs based on polynomial approximation [24], we implemented the EVMDD-based NFGs for $\sqrt{X^2+Y^2}$ and $XY/\sqrt{X^2+Y^2}$ using the same Altera FPGA (EP3SL340F1517C2) and QuartusII 9.0.

For both functions, the maximum delay times of the EVMDD-based NFGs are 21 nsec., while those of the polynomial-based two-variable NFGs [24] are 35 and 37 nsec., respectively. Since the EVMDD-based NFGs are faster than the polynomial-based NFGs, and they can directly realize two-variable functions, the proposed two-variable NFGs are faster than the NFGs designed by any other existing approaches.

As mentioned previously, our NFGs can also realize one-variable functions. To show the effectiveness of our NFGs for one-variable functions, we compare our NFGs with a CORDIC shown in [1], which is well known as a standard one-variable NFG for FPGA implementation, in terms of performance. Table 6 shows the results. From this table, we can see that the EVMDD-based NFG that is automatically generated from the function table evaluates the sine function faster than a manually implemented CORDIC. And, the EVBDD-based NFG is faster than the CORDIC synthesized from RTL.

In this way, we can implement fast and compact one-variable and two-variable NFGs by using EVMDDs.

5. Conclusion and Comments

This paper introduces a new class of integer-valued functions, called an l-restricted Mp-monotone increasing function. It also derives an upper bound on the number of nodes in an EVBDD to represent the function. EVBDDs represent l-restricted Mp-monotone increasing functions or their affine transformations more compactly than MTBDDs and BMDs when p is small.

This paper also presents a design method for NFGs based on EVMDDs. With EVMDDs, we can design accurate, fast, and compact NFGs for one- and two-variable numeric functions. In FPGA implementations for two-variable numeric functions, we show that EVMDD-based NFGs require, on the average, only 37% of the delay time and 62% of the memory size needed for EVBDD-based NFGs.

Acknowledgments

This research is partly supported by the Grant in Aid for Scientific Research of the Japan Society for the Promotion of Science (JSPS), funds from Ministry of Education, Culture, Sports, Science, and Technology (MEXT) via Knowledge Cluster Project, the MEXT Grant-in-Aid for Young Scientists (B), 200700051, 2009, and Hiroshima City University Grant for Special Academic Research (General Studies), 8108, 2009. The comments of reviewers were helpful in improving this paper.

References

[1] F. Angarita, A. Perez-Pascual, T. Sansaloni, and J. Valls, "Efficient FPGA implementation of CORDIC algorithm for circular and linear coordinates," Proc. Inter. Conf. on Field Programmable Logic and Applications (FPL'05), pp.535–538, Tampere, Finland, Aug. 2005.
[2] H. Anton, Multivariable Calculus, John Wiley & Sons, 1995.
[3] R.E. Bryant, "Graph-based algorithms for boolean function manipulation," IEEE Trans. Comput., vol.C-35, no.8, pp.677–691, Aug. 1986.
[4] R.E. Bryant and Y.-A. Chen, "Verification of arithmetic circuits with binary moment diagrams," Design Automation Conference, pp.535–541, 1995.
[5] D. De Caro and A.G.M. Strollo, "High-performance direct digital frequency synthesizers using piecewise-polynomial approximation," IEEE Trans. Circuits Syst., vol.52, no.2, pp.324–337, Feb. 2005.
[6] E.M. Clarke, K.L. McMillan, X. Zhao, M. Fujita, and J. Yang, "Spectral transforms for large Boolean functions with applications to technology mapping," Proc. 30th ACM/IEEE Design Automation Conference, pp.54–60, June 1993.
[7] J. Detrey and F. de Dinechin, "Table-based polynomials for fast hardware function evaluation," 16th IEEE Inter. Conf. on Application-Specific Systems, Architectures, and Processors (ASAP'05), pp.328–333, 2005.
[8] R. Drechsler and B. Becker, Binary Decision Diagrams: Theory and Implementation, Kluwer Academic Publishers, 1998.
[9] R. Gutierrez and J. Valls, "Implementation on FPGA of a LUT based $atan(y/x)$ operator suitable for synchronization algorithms," Proc. IEEE Conf. on Field Programmable Logic and Applizations, pp.472–475, Aug. 2007.
[10] Z. Huang and M.D. Ercegovac, "FPGA implementation of pipelined on-line scheme for 3-D vector normalization," Proc. 9th Annual IEEE Symp. on Field-Programmable Custom Computing Machines (FCCM'01), pp.61–70, April 2001.
[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. 2000.
[12] Y. Iguchi, T. Sasao, and M. Matsuura, "Evaluation of multiple-output logic functions using decision diagrams," Asia and South Pacific Design Automation Conference (ASP-DAC'2003), pp.312–315, Kitakyushu, Jan. 2003.
[13] T. Kam, T. Villa, R.K. Brayton, and A.L. Sangiovanni-Vincentelli, "Multi-valued decision diagrams: Theory and applications," Multiple-Valued Logic: An International Journal, vol.4, no.1-2, pp.9–62, 1998.
[14] Y.-T. Lai and S. Sastry, "Edge-valued binary decision diagrams for multi-level hierarchical verification," Proc. 29th ACM/IEEE Design Automation Conference, pp.608–613, 1992.
[15] Y.-T. Lai, M. Pedram, and S.B. Vrudhula, "EVBDD-based algorithms for linear integer programming, spectral transformation and functional decomposition," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.13, no.8, pp.959–975, Aug. 1994.
[16] D.-U. Lee, W. Luk, J. Villasenor, and P.Y.K. Cheung, "Hierarchical segmentation schemes for function evaluation," Proc. IEEE Conf. on Field-Programmable Technology, pp.92–99, Tokyo, Dec. 2003.
[17] C. Meinel and T. Theobald, Algorithms and Data Structures in VLSI Design: OBDD – Foundations and Applications, Springer, 1998.
[18] J.-M. Muller, Elementary Function: Algorithms and Implementation, Birkhauser Boston, Secaucus, NJ, 1997.
[19] S. Nagayama and T. Sasao, "Compact representations of logic functions using heterogeneous MDDs," IEICE Trans. Fundamentals, vol.E86-A, no.12, pp.3168–3175, Dec. 2003.
[20] S. Nagayama and T. Sasao, "On the optimization of heterogeneous MDDs," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.24, no.11, pp.1645–1659, Nov. 2005.
[21] S. Nagayama and T. Sasao, "Representations of elementary functions using edge-valued MDDs," 37th International Symposium on Multiple-Valued Logic, Oslo, Norway, May 2007.
[22] S. Nagayama and T. Sasao, "Representations of two-variable elementary functions using EVMDDs and their applications to function generators," 38th International Symposium on Multiple-Valued Logic, pp.50–56, Dallas, U.S.A., May 2008.
[23] S. Nagayama and T. Sasao, "Complexities of graph-based representations for elementary functions," IEEE Trans. Comput., vol.58, no.1, pp.106–119, Jan. 2009.
[24] S. Nagayama, T. Sasao, and J.T. Butler, "Programmable architectures and design methods for two-variable numeric function generators," IPSJ Trans. System LSI Design Methodology, vol.3, pp.118–129, Feb. 2010.
[25] J.-A. Piñeiro, S.F. Oberman, J.-M. Muller, and J.D. Bruguera, "High-speed function approximation using a minimax quadratic interpolator," IEEE Trans. Comput., vol.54, no.3, pp.304–318, March 2005.
[26] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," International Conference on Computer-Aided Design (ICCAD'93), pp.42–47, Nov. 1993.
[27] T. Sasao and M. Fujita (eds.), Representations of Discrete Functions, Kluwer Academic Publishers, 1996.
[28] T. Sasao, Switching Theory for Logic Synthesis, Kluwer Academic Publishers, 1999.
[29] T. Sasao and S. Nagayama "Representations of elementary functions using binary moment diagrams," 36th International Symposium on Multiple-Valued Logic, Singapore, May 2006.
[30] T. Sasao, S. Nagayama, and J.T. Butler, "Numerical function generators using LUT cascades," IEEE Trans. Comput., vol.56, no.6, pp.826–838, June 2007.
[31] M.J. Schulte and J.E. Stine, "Approximating elementary functions with symmetric bipartite tables," IEEE Trans. Comput., vol.48, no.8, pp.842–847, Aug. 1999.
[32] R. Stankovic and J. Astola, Spectral Interpretation of Decision Diagrams, Springer Verlag, New York, 2003.
[33] R. Stankovic and J. Astola, "Remarks on the complexity of arithmetic representations of elementary functions for circuit design," Workshop on Applications of the Reed-Muller Expansion in Circuit Design and Representations and Methodology of Future Computing Technology, pp.5–11, May 2007.
[34] N. Takagi and S. Kuwahara, "A VLSI algorithm for computing the Euclidean norm of a 3D vector," IEEE Trans. Comput., vol.49, no.10, pp.1074–1082, Oct. 2000.
[35] M.A. Thornton, R. Drechsler, and D.M. Miller, Spectral Techniques in VLSI CAD, Springer, 2001.
[36] I. Wegener, Branching Programs and Binary Decision Diagrams: Theory and Applications, SIAM, 2000.
[37] S.N. Yanushkevich, D.M. Miller, V.P. Shmerko, and R.S. Stankovic, Decision Diagram Techniques for Micro- and Nanoelectronic Design, CRC Press, Taylor & Francis Group, 2006.

Appendix

Lemma A: [21] The number of distinct n-bit precision Mp-monotone increasing functions is $(p+1)^{2^n-1}$.

Definition A: A shared EVBDD (SEVBDD) is an extension of an EVBDD, and it has multiple root nodes to represent multiple integer-valued functions. The SEVBDD is obtained by sharing equivalent sub-graphs in EVBDDs for the integer-valued functions.

Lemma B: [21] Let $\eta(l, p)$ be the number of non-terminal nodes in the SEVBDD representing all the l-bit precision Mp-monotone increasing functions, where the variable order of the SEVBDD is $x_{l-1}, x_{l-2}, \ldots, x_0$ (from the root nodes to the terminal node). Then,

$$\eta(l, p) = \sum_{i=1}^{l} (p+1)^{2^i-1} - l.$$

Proof for Theorem 1: Suppose that an EVBDD for $f(X)$ is partitioned into two parts: the upper and the lower parts as shown in Fig. A· 1. In this case, the lower part represents l-bit precision Mp-monotone increasing functions, and the upper part represents the selector function. The upper part has the maximum number of nodes when it forms a complete binary tree. That is, the maximum number of nodes in the upper part is $2^{n-l} - 1$.

The lower part has the maximum number of nodes when it represents all the l-bit precision Mp-monotone increasing functions. From Lemma B, the maximum number of nodes in the lower part is $\eta(l, p)$.

Therefore, the number of non-terminal nodes in the EVBDD for $f(X)$ is at most

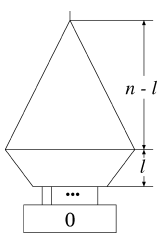


Fig. A· 1 Partition of EVBDD.

$$2^{n-1} - 1 + \eta(l, p) = 2^{n-l} + \sum_{i=1}^{l} (p+1)^{2^{i-1}} - l - 1.$$

By adding 1 to account for the terminal node to this, we have (1). From Lemma A, the number of Mp-monotone increasing functions that can be represented in the lower part is $(p+1)^{2^l-1}$. It does not exceed the number of functions which can be selected by the upper part: 2^{n-l}. Therefore, we have the relation: $(p+1)^{2^l-1} \leq 2^{n-l}$. ∎

Proof for Lemma 2: In the EVBDD for $f(X)$, adding constant values to the weights of edges pointing to the corresponding sub-functions can produce the EVBDD for $g(X)$. This conversion of EVBDDs does not change the number of nodes. ∎

Proof for Lemma 3: For $f(X, Y)$, any sub-function with respect to an assignment to X can be represented by $h(Y)+b$, where b is an integer. Thus, from Definition 12, the lemma holds. ∎

Proof for Lemma 4:

$$f(X, Y) = g(X) - h(Y) = -(-g(X) + h(Y))$$

From Lemma 3, $(-g(X) + h(Y))$ is an extended n-restricted Mp-monotone increasing function. Therefore, we have the lemma. ∎

Proof for Lemma 5: For $f(X, Y)$, any sub-function with respect to an assignment to X can be represented by $a \cdot h(Y)$, where a is a real number satisfying $0 \leq a \leq 1$. Since $a \cdot h(0) = 0$ and $0 \leq a \cdot h(Y+1) - a \cdot h(Y) \leq p$ hold, from Definition 11, the lemma holds. ∎

Proof for Lemma 6: Let $h(Y) = a \cdot h_0(Y) + b$. Then,

$$\begin{aligned} f(X, Y) &= g(X)(a \cdot h_0(Y) + b) \\ &= a\left(g(X) \cdot h_0(Y) + \frac{b}{a} \cdot g(X)\right). \end{aligned}$$

From Lemmas 3 and 5, $g(X) \cdot h_0(Y) + \frac{b}{a} \cdot g(X)$ is an extended n-restricted Mp-monotone increasing function. Thus, we have the lemma. ∎

Shinobu Nagayama received the B.S. and M.E. degrees from the Meiji University, Kanagawa, Japan, in 2000 and 2002, respectively, and the Ph.D. degree in computer science from the Kyushu Institute of Technology, Iizuka, Japan, in 2004. He is now an Associate Professor at the Hiroshima City University, Hiroshima, Japan. He received the Outstanding Contribution Paper Award from the IEEE Computer Society Technical Committee on Multiple-Valued Logic (MVL-TC) in 2005 for a paper presented at the International Symposium on Multiple-Valued Logic in 2004, and the Excellent Paper Award from the Information Processing Society of Japan (IPS) in 2006. His research interest includes numeric function generators, 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, the IBM T.J. Watson Research Center, Yorktown Heights, New York, and the Naval Postgraduate School, Monterey, California. He is now a Professor of the Department of Computer Science and Electronics 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 nine books on logic design, including Logic Synthesis and Optimization, Representation of Discrete Functions, Switching Theory for Logic Synthesis, and Logic Synthesis and Verification, Kluwer Academic Publishers, 1993, 1996, 1999, and 2001, respectively. He has served as Program Chairman for the IEEE International Symposium on Multiple-Valued Logic (ISMVL) many times. Also, he was the Symposium Chairman of the 28th ISMVL held in Fukuoka, Japan, in 1998. He received the NIWA Memorial Award in 1979, Distinctive Contribution Awards from the IEEE Computer Society MVL-TC for papers presented at ISMVLs in 1986, 1996, 2003 and 2004, and Takeda Techno-Entrepreneurship Award in 2001. He has served as an Associate Editor of the IEEE Transactions on Computers. He is a fellow of the IEEE.

Jon T. Butler received the B.E.E. and M.Engr. degrees from Rensselaer Polytechnic Institute, Troy, New York, in 1966 and 1967, respectively. He received the Ph.D. degree from The Ohio State University, Columbus, in 1973. Since 1987, he has been a professor at the Naval Postgraduate School, Monterey, California. From 1974 to 1987, he was at Northwestern University, Evanston, Illinois. During that time, he served two periods of leave at the Naval Postgraduate School, first as a National Research Council Senior Postdoctoral Associate (1980–1981) and second as the NAVALEX Chair Professor (1985–1987). He served one period of leave as a foreign visiting professor at the Kyushu Institute of Technology, Iizuka, Japan. His research interests include logic optimization and multiple-valued logic. He has served on the editorial boards of the IEEE Transactions on Computers, Computer, and IEEE Computer Society Press. He has served as the editor-in-chief of Computer and IEEE Computer Society Press. He received the Award of Excellence, the Outstanding Contributed Paper Award, and a Distinctive Contributed Paper Award for papers presented at the International Symposium on Multiple-Valued Logic. He received the Distinguished Service Award, two Meritorious Awards, and nine Certificates of Appreciation for service to the IEEE Computer Society. He is a fellow of the IEEE.