Representations of Two-Variable Elementary Functions Using EVMDDs and Their Applications to Function Generators

Shinobu Nagayama
Department of Computer Engineering,
Hiroshima City University
Hiroshima 731-3194, Japan

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

Abstract

This paper proposes a method to represent two-variable elementary functions using edge-valued multi-valued decision diagrams (EVMDDs), and presents a design method and an architecture for function generators using EVMDDs. To show the compactness of EVMDDs, this paper introduces a new class of integer-valued functions, l-restricted Mp-monotone increasing functions, and derives an upper bound on the number of nodes in an edge-valued binary decision diagram (EVBDD) for the l-restricted Mp-monotone increasing function. EVBDDs represent l-restricted Mp-monotone increasing functions more compactly than MTBDDs and BMDs when p is small. Experimental results show that all the two-variable elementary functions considered in this paper can be converted into l-restricted Mp-monotone increasing functions with $p = 1$ or $p = 3$, and can be compactly represented by EVBDDs. Since EVMDDs have shorter paths and smaller memory size than EVBDDs, EVMDDs can produce fast and compact elementary function generators.

1. Introduction

Elementary functions have wide applications including computer graphics and digital signal processing, and various circuits (function generators) for them have been developed [16]. However, most existing methods are intended for one-variable elementary functions [5, 14, 19, 20, 24–26], and only a few methods exist for multi-variable elementary functions [7, 8, 29]. As far as we know, no study on graph-based representations for multi-variable elementary functions has been presented.

This paper proposes a method to represent two-variable elementary functions using EVMDDs, and presents an architecture and a design method for function generators. To analyze complexities for two-variable elementary functions, this paper introduces a new class of integer-valued functions, l-restricted Mp-monotone increasing functions, and derives an upper bound on the number of nodes in an edge-valued binary decision diagram (EVBDD) for the function. Theoretical analysis and experimental results show that EVMDDs can compactly represent both one- and two-variable elementary functions, and our function generators 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 elementary function into an integer-valued function, and decision diagrams used in this paper. Section 3 considers representations of two-variable elementary functions using decision diagrams. It introduces an l-restricted Mp-monotone increasing function, and derives an upper bound on the number of nodes in an EVBDD for the function. Section 4 presents an architecture and a design method for function generators based on EVMDDs. And, Section 5 concludes the paper. Proofs of theorems are omitted because of the page limitation.

2. Preliminaries

2.1. Number Representation and Precision

Definition 1 *Let $B = \{0,1\}$, Z be the set of the integers, and R be the set of the real numbers. An n-input m-output* ***logic function*** *is a mapping: $B^n \rightarrow B^m$, an* ***integer-valued function*** *is $B^n \rightarrow Z$, and a* ***two-variable real function*** *is $R \times R \rightarrow R$.*

Definition 2 *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. In this paper, $\{X\}$ denotes the set of binary variables in X.*

Definition 3 ***Precision*** *is the total number of bits for a binary fixed-point representation. Specially,* ***n-bit precision***

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

*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 precisions*[1].

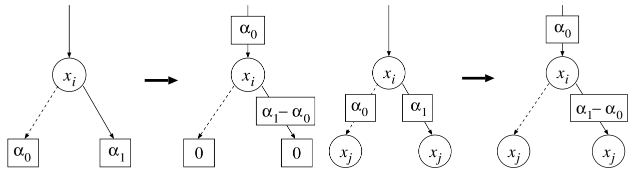
By fixed-point representation, we can convert an n-bit precision two-variable real function into a $2n$-input m-output logic function. The logic function can be converted into an integer-valued function by considering binary vectors as integers. That is, we can convert an n-bit precision two-variable real function into an integer-valued function: $B^{2n} \to P_m$, where $P_m = \{0, 1, \ldots, 2^m - 1\}$. In this paper, two-variable elementary functions are converted into integer-valued functions by using fixed-point representation, unless stated otherwise. And, for simplicity, 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 for a two-dimensional vector: $\sqrt{X^2+Y^2}$. 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)*

2.2. Decision Diagrams

This subsection defines decision diagrams used in this paper. For more detail on definition and reduction rules of each decision diagram, see [6, 22, 31, 32].

[1]Although the precision for function values can be different from the precision of the inputs, in this paper, we assume that the number of fractional bits for function values is n. Thus, function values have $(n\_int + n)$-bit precision, where n_int is the number of integer bits for function values.



(a) Terminal case. (b) Non-terminal case.

Figure 1. Conversion of an MTBDD node into an EVBDD node.

Definition 4 *A binary decision diagram (BDD) [2, 15] is a rooted directed acyclic graph (DAG) representing a logic function. The BDD is obtained by repeatedly applying the Shannon expansion $f = \overline{x_i} f_0 + x_i f_1$ to the logic function. It consists of two terminal nodes representing function values 0 and 1 respectively, and non-terminal nodes representing input variables. Each non-terminal node has two unweighted outgoing edges, 0-edge and 1-edge, that correspond to the values of the input variables. Both terminal nodes have no outgoing edges.*

Definition 5 *A **multi-terminal BDD (MTBDD)** [4] is an extension of a BDD, and represents an integer-valued function. In the MTBDD, the terminal nodes are labeled by integers.*

Definition 6 *A **binary moment diagram (BMD)** [3] is a rooted 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. 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)** [12, 13] 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 $f_1 = f_1' + \alpha$, 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, and the incoming edge into the root node can have a non-zero weight. In a reduced EVBDD, each node represents a distinct sub-function. An EVBDD is obtained from an MTBDD by recursively applying the conversion shown in Fig. 1 to each non-terminal node in an MTBDD, where in Fig. 1, dashed lines and solid lines denote 0-edges and 1-edges, respectively.*

Definition 8 *For an n-bit precision number X, if $\{X\} = \{X_u\} \cup \{X_{u-1}\} \cup \ldots \cup \{X_1\}$, $\{X_i\} \neq \phi$, and $\{X_i\} \cap \{X_j\} =$*

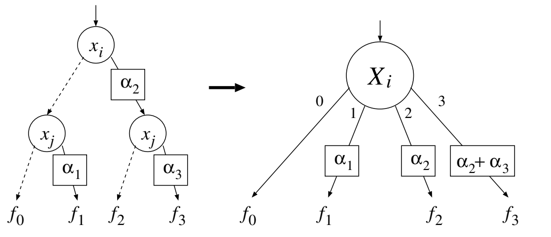


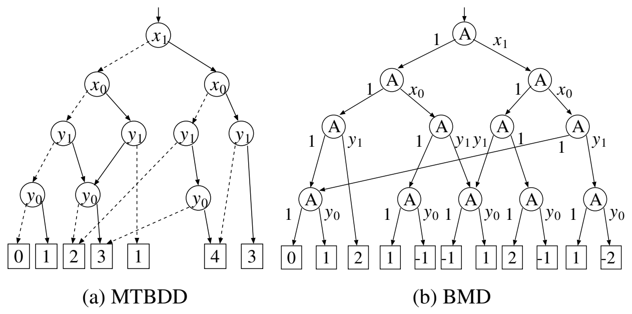
Figure 2. Conversion of EVBDD nodes into an EVMDD node.

ϕ $(i \neq j)$, 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 multi-valued variable, an integer-valued function $f(X) : B^n \to Z$ can be converted into a* ***multi-valued input integer function*** *$f(X_u, X_{u-1}, \ldots, X_1) : P_u \times P_{u-1} \times \ldots \times P_1 \to Z$, where $P_i = \{0, 1, 2, \ldots, 2^{k_i} - 1\}$.*

Definition 9 *A multi-valued decision diagram (MDD) is a rooted DAG representing a multi-valued input integer function. The MDD is obtained by repeatedly applying the Shannon expansion to the multi-valued input integer function [11]. It consists of terminal nodes representing function values and non-terminal nodes representing multi-valued variables. Each non-terminal node has multiple unweighted outgoing edges that correspond to the values of multi-valued variable. When an MDD represents a function for which multi-valued variables have different domains, it is a* ***heterogeneous MDD*** *[17, 18]. In the following, the heterogeneous MDD is simply denoted by the MDD.*

Definition 10 *An* ***edge-valued MDD (EVMDD)*** *is an extension of an MDD, and represents a multi-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. 2, an EVMDD is obtained by merging non-terminal nodes in an EVBDD according to the partition of X.*

Example 2 *Fig. 3 (a), (b), (c), and (d) show the MTBDD, the BMD, the EVBDD, and the EVMDD for the 2-bit precision 2-D norm function in Table 1 (c), respectively. Note that for readability of the figures, several terminal nodes are not shared. In Fig. 3 (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. 3 (b), 'A' in a circle denotes the arithmetic transform expansion. And, in Fig. 3 (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,*



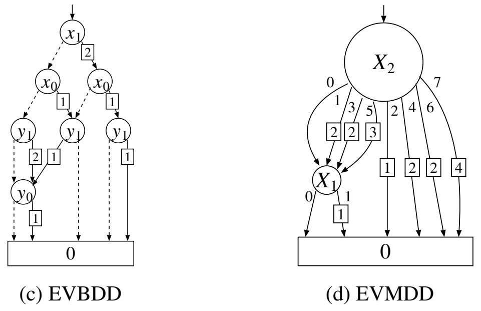


Figure 3. Four types of decision diagrams for the 2-bit precision 2-D norm function.

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 Elementary 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 elementary functions are more compact than MTBDDs and BMDs for them.

3.1. l-restricted Mp-monotone Increasing Functions

Definition 11 *An n-bit precision integer-valued function $f(X)$ such that $0 \leq f(X+1) - f(X) \leq p$ and $f(0) = 0$ is a* ***totally Mp-monotone increasing function*** *(or simply,* ***Mp-monotone increasing function****). 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.*

Definition 12 *An n-bit precision integer-valued function $f(X)$ is an **l-restricted Mp-monotone increasing function** when for $1 \leq l < n$, all the l-bit precision sub-functions $g(X_l)$ of f are Mp-monotone increasing functions, where $g(X_l) = f(\vec{a}, X_l)$, $\{X\} = \{x_{n-1}, x_{n-2}, \ldots, x_0\}$, $\{X_l\} = \{x_{l-1}, x_{l-2}, \ldots, x_0\}$, and $\vec{a}$ is an assignment 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*

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

where l is the largest integer satisfying $2^{n-l} \geq (p+1)^{2^l - 1}$, 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).

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 [19].

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

Definition 13 *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 a sub-function is $f(\vec{a}, X_l)$, 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 $\vec{a}$ is an assignment 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.*

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 which is obtained by adding constant values to l-bit precision sub-functions of f. Then, the EVBDDs for $f(X)$ and $g(X)$ have the same number of nodes.*

Corollary 1 *Let $f(X)$ be an extended l-restricted Mp-monotone increasing function, and let $g(X)$ be a linear 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.*

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

(a) Table for 2-D norm.

	X			
Y	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 $\frac{X}{Y+1}$.

	X			
Y	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 $g(Z)$.

	$x_1 x_0$			
$y_1 y_0$	00	01	10	11
00	0	-1	-2	-3
01	0	-1	-2	-2
10	0	-1	-1	-2
11	0	-1	-1	-2

3.2. Two-Variable Elementary Functions

As shown in Section 2, n-bit precision two-variable elementary 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 the extended 2-restricted M1-monotone increasing function $f(Z)$. 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 a linear transformation of the extended 2-restricted M1-monotone increasing function $g(Z)$ shown in Table 2 (c): $-1 \times g(Z)$. (End of Example)*

From here, we are going to show some classes of two-variable elementary functions whose EVBDDs are small.

Lemma 3 *Let $h(Y)$ be an n-bit precision Mp-monotone increasing function. Then, for arbitrary one-variable function $g(X)$, two-variable functions $f(X,Y) = g(X) + h(Y)$ are the extended n-restricted Mp-monotone increasing functions.*

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

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.*

In Lemma 5, if the dynamic range of $g(X)$ is large, then the EVBDD can be large. For example, the n-bit multiplier requires O(2^n) nodes [31].

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

Elementary functions	Type of functions	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 $0 \leq X < 1$ and $0 \leq Y < 1$.
Number of fractional bits for function values is 8.
Mp^+: the function is a linear transformation of an extended 8-restricted Mp-monotone increasing function.
R_1 = (EVBDD) / (MTBDD) $\times$ 100. R_2 = (EVBDD) / (BMD) $\times$ 100.
Variable orders of decision diagrams are produced by the sifting algorithm [21].

Lemma 6 *Let $h(Y)$ be a linear 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 a linear transformation of an extended n-restricted Mp-monotone increasing function.*

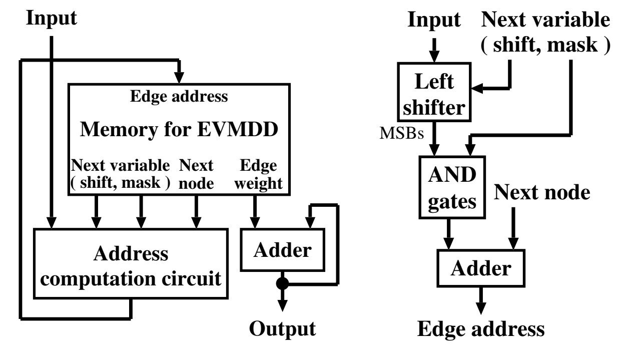
Example 5 *2-bit precision function $\frac{1}{Y+1}$ is a linear transformation of an M1-monotone increasing function [19]. As shown in Example 4, $f(X,Y) = \frac{X}{Y+1}$ can be converted into a linear transformation of an extended 2-restricted M1-monotone increasing function. (End of Example)*

In the following, we will show that various two-variable elementary 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 8-bit precision two-variable elementary functions. These functions are arbitrarily selected from books on multivariable calculus such as [1]. *WaveRings* in the table is defined by

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

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

Two-variable elementary functions whose function values smoothly change 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 elementary 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 [19].



(a) Architecture for function generators. (b) Address computation circuit.

Figure 4. Architecture for function generators based on EVMDDs.

As shown in [19], by converting EVBDDs into EVMDDs, we can often reduce memory size and path length of decision diagrams. In the next section, we present the function generator taking advantage of EVMDDs.

4. Function Generators for Two-Variable Elementary Functions

This section presents an architecture and a design method for function generators based on EVMDDs and EVBDDs.

4.1. Architecture for Function Generators

In decision diagrams based on the Shannon expansion, function values can be obtained by traversing the decision diagrams from the root node to a terminal node [9, 10]. In EVBDDs and EVMDDs, function values can be obtained as the sum of the weights for traversed edges. Fig. 4 shows the function generator based on EVMDD. It consists of a memory to store an EVMDD, an address computation circuit to traverse an EVMDD, and an adder to compute sum of edge weights. Note that for readability of the figures, registers, circuits for initialization, and some signals are omitted from Fig. 4.

In Fig. 4 (a), the memory for EVMDD stores data for edges in an EVMDD. Data for an edge consist of a pointer to the next node, data for next variable of the next node,

Table 4. Memory data for the function generator 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

and an edge weight. From the memory, a pointer to the next node and data for next variable are read and fed to the address computation circuit. And, an edge weight is fed to the adder. The address computation circuit produces an address of the next edge, from an address of node and a value of the input variable.

Fig. 4 (b) shows the address computation circuit. Data for 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.

Example 6 *Table 4 shows memory data and initial values for the function generator produced from the EVMDD in Fig. 3 (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 of 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 $(101)_2 = 5$.

Next, data for the address 5 are read from the memory and fed to the address computation circuit and the adder. The adder 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 to the 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 node 8 yields the second edge address 8.

Since in data for the address 8, the mask data 0 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)

Since the circuit shown in Fig. 4 just traverses an EVMDD and computes sum of edge weights, it can evaluate both one- and two-variable functions with the same architecture.

4.2. Design Method for Function Generators Using EVMDDs

For given elementary function, its domain, and precision, we can systematically design the circuit in Fig. 4. First, convert a given elementary 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. 4 from the EVMDD. Since our function generator directly realizes the function table, it is more accurate than the existing function generators using polynomial approximation [5, 14, 20, 25, 26].

To generate memory data like Table 4, 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-edge. 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:

$$\begin{aligned} \text{shift data} &= \sum_{j=i}^{u-1} k_j \qquad \text{mask data} = 2^{k_i} - 1 \\ \text{bit size for mask} &= \max_{1 \le j \le u} (k_j) \end{aligned}$$

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 $k_j = |X_j|$ $(j = 1, 2, \ldots, u)$.

Memory size and delay time of our function generator depend largely on memory size and path length of EVMDD. Therefore, the memory minimization algorithm and the APL minimization algorithm for MDDs [17, 18] are useful to produce fast and compact function generators.

5. Conclusion and Comments

This paper has introduced a new class of integer-valued functions, called an l-restricted Mp-monotone increasing function. It has also derived 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 linear transformations more compactly than MTBDDs and BMDs when p is small.

This paper has also presented a design method for function generators based on EVMDDs. With EVMDDs, we can realize accurate, fast, and compact function generators for two-variable elementary functions. In FPGA implementations for two-variable elementary functions, we confirmed that EVMDD-based function generators require, on the average, only 37% of the delay time and 62% of the memory size needed for EVBDD-based function generators.

Our future works include deriving lower bounds on the number of nodes in decision diagrams, and designing faster and more compact function generators.

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 Initiative, the MEXT Grant-in-Aid for Young Scientists (B), and Hiroshima City University Grant for Special Academic Research (General Studies).

References

[1] H. Anton, *Multivariable Calculus*, John Wiley & Sons, Inc., 1995.

[2] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, Vol. C-35, No. 8, pp. 677–691, Aug. 1986.

[3] R. E. Bryant and Y-A. Chen, "Verification of arithmetic circuits with binary moment diagrams," *Design Automation Conference*, pp. 535–541, 1995.

[4] 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. of 30th ACM/IEEE Design Automation Conference*, pp. 54–60, June 1993.

[5] 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.

[6] R. Drechsler and B. Becker, *Binary Decision Diagrams: Theory and Implementation*, Kluwer Academic Publishers, 1998.

[7] R. Gutierrez and J. Valls, "Implementation on FPGA of a LUT based $atan(y/x)$ operator suitable for synchronization algorithms," *Proc. of the IEEE Conf. on Field Programmable Logic and Applizations*, pp. 472–475, Aug. 2007.

[8] Z. Huang and M. D. Ercegovac, "FPGA implementation of pipelined on-line scheme for 3-D vector normalization," *Proc. of the 9th Annual IEEE Symp. on Field-Programmable Custom Computing Machines (FCCM'01)*, pp. 61–70, Apr. 2001.

[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)*, pp. 73–76, Yokohama, Japan, Jan. 2000.

[10] 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, Japan, Jan. 2003.

[11] 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.

[12] Y-T. Lai and S. Sastry, "Edge-valued binary decision diagrams for multi-level hierarchical verification," *Proc. of 29th ACM/IEEE Design Automation Conference*, pp. 608–613, 1992.

[13] 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.

[14] D.-U. Lee, W. Luk, J. Villasenor, and P. Y. K. Cheung, "Hierarchical segmentation schemes for function evaluation," *Proc. of the IEEE Conf. on Field-Programmable Technology*, Tokyo, Japan, pp. 92–99, Dec. 2003.

[15] C. Meinel and T. Theobald, *Algorithms and Data Structures in VLSI Design: OBDD – Foundations and Applications*, Springer, 1998.

[16] J.-M. Muller, *Elementary Function: Algorithms and Implementation*, Birkhauser Boston, Inc., Secaucus, NJ, 1997.

[17] S. Nagayama and T. Sasao, "Compact representations of logic functions using heterogeneous MDDs," *IEICE Trans. on Fundamentals*, Vol. E86-A, No. 12, pp. 3168–3175, Dec. 2003.

[18] S. Nagayama and T. Sasao, "On the optimization of heterogeneous MDDs," *IEEE Trans. on CAD*, Vol. 24, No. 11, pp. 1645–1659, Nov. 2005.

[19] S. Nagayama and T. Sasao, "Representations of elementary functions using edge-valued MDDs," *37th International Symposium on Multiple-Valued Logic*, Oslo, Norway, May 13-16, 2007.

[20] J.-A. Piñeiro, S. F. Oberman, J.-M. Muller, and J. D. Bruguera, "High-speed function approximation using a minimax quadratic interpolator," *IEEE Trans. on Comp.*, Vol. 54, No. 3, pp. 304–318, Mar. 2005.

[21] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," *International Conference on Computer-Aided Design (ICCAD'93)*, pp. 42–47, Nov. 1993.

[22] T. Sasao and M. Fujita (eds.), *Representations of Discrete Functions*, Kluwer Academic Publishers, 1996.

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

[24] T. Sasao and S. Nagayama "Representations of elementary functions using binary moment diagrams," *36th International Symposium on Multiple-Valued Logic*, Singapore, May 17-20, 2006.

[25] T. Sasao, S. Nagayama, and J. T. Butler, "Numerical function generators using LUT cascades," *IEEE Transactions on Computers*, Vol. 56, No. 6, pp. 826–838, Jun. 2007.

[26] M. J. Schulte and J. E. Stine, "Approximating elementary functions with symmetric bipartite tables," *IEEE Trans. on Comp.*, Vol. 48, No. 8, pp. 842–847, Aug. 1999.

[27] R. Stankovic and J. Astola, *Spectral Interpretation of Decision Diagrams*, Springer Verlag, New York, 2003.

[28] 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.

[29] N. Takagi and S. Kuwahara, "A VLSI algorithm for computing the Euclidean norm of a 3D vector," *IEEE Transactions on Computers*, Vol. 49, No. 10, pp. 1074–1082, Oct. 2000.

[30] M. A. Thornton, R. Drechsler, and D. M. Miller, *Spectral Techniques in VLSI CAD*, Springer, 2001.

[31] I. Wegener, *Branching Programs and Binary Decision Diagrams: Theory and Applications*, SIAM, 2000.

[32] 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.