Representations of Elementary Functions Using Edge-Valued MDDs

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 elementary functions such as trigonometric, logarithmic, square root, and reciprocal functions using edge-valued multi-valued decision diagrams (EVMDDs). We introduce a new class of integer functions, Mp-monotone increasing functions, and derive an upper bound on the number of nodes in an edge-valued binary decision diagram (EVBDD) for the Mp-monotone increasing function. The upper bound shows that EVBDDs represent Mp-monotone increasing functions more compactly than other decision diagrams when p is small. Experimental results using 16*-bit precision elementary functions show that: 1) standard elementary functions can be converted into Mp-monotone increasing functions with* $p = 1$ *or* $p = 2$*, or their linear transformations. And, they can be compactly represented by EVBDDs. 2) EVMDDs represent elementary functions with, on average, only* 11% *of the memory size needed for binary moment diagrams (BMDs), and only* 69% *of the memory size needed for EVBDDs.*

1. Introduction

Decision diagrams are widely used to represent functions compactly. For different types of functions, various decision diagrams have been proposed. For example, for logic functions, binary decision diagram (BDD) [1] and functional decision diagram (FDD) [7] are useful. For integer functions such as adder and multiplier, arithmetic transform decision diagram (ACDD) [16], binary moment diagram (BMD) [2], multiplicative BMD (*BMD) [2], Kronecker multiplicative BMD (K*BMD) [5], and Taylor expansion diagram (TED) [3] are compact. And, for complex matrices, quantum multiple-valued decision diagram (QMDD) [9] has been proposed recently.

This paper proposes a method to represent elementary functions [10] such as trigonometric, logarithmic, square root, and reciprocal functions using decision diagrams. In [15], we used the property that elementary functions can be expanded into polynomial functions, and we represented elementary functions using BMDs for polynomials. In this paper, however, we use the property that elementary functions are monotone functions on the standard domain, and we represent elementary functions using decision diagrams for monotone functions. Theoretical analysis and experimental results show that edge-valued multi-valued decision diagrams (EVMDDs) represent elementary functions more compactly than BMDs.

2. 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 function*** *is* $Z \rightarrow Z$*, and a* ***real function*** *is* $R \rightarrow R$*.*

Definition 2 *A value X represented by the* ***binary fixed-point representation*** *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.*

Definition 3 ***Precision*** *is the total number of bits for 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*** $f(X)$ *means that the input variable X has n-bit precision.*

By fixed-point representation, we can convert an n-bit precision real valued function into an n-input m-output logic function. The logic function can be converted into an integer function by considering binary vectors as integers. That is, we can convert an n-bit precision real valued function into an integer function: $P_n \rightarrow P_m$, where $P_n = \{0,1,\ldots,2^n-1\}$ and $P_m = \{0,1,\ldots,2^m-1\}$. In this paper, elementary functions are converted into integer functions by using n-bit fixed-point representation, unless stated

Table 1. Function table for the 3bit-precision $\sin(X)$.

(a) Function table for $\sin(X)$.

X	$\sin(X)$
0.000	0.000
0.125	0.125
0.250	0.247
0.375	0.366
0.500	0.479
0.625	0.585
0.750	0.682
0.875	0.768

(b) Truth table for logic function $f_b(X)$.

X	$f_b(X)$
0.000	0.000
0.001	0.001
0.010	0.010
0.011	0.011
0.100	0.100
0.101	0.101
0.110	0.101
0.111	0.110

(c) Function table for integer function $f(X)$.

$X = (x_2x_1x_0)_2$	$f(X)$
$0 = (000)_2$	0
$1 = (001)_2$	1
$2 = (010)_2$	2
$3 = (011)_2$	3
$4 = (100)_2$	4
$5 = (101)_2$	5
$6 = (110)_2$	5
$7 = (111)_2$	6

otherwise. And, for simplicity, x_0 denotes the least significant bit in the fixed-point representation of X.

Example 1 *Table 1(a) is the function table for* $\sin(X)$. *By representing this function using the* 3*-bit precision fixed-point representation, we have the logic function* $f_b(X)$ *in Table 1(b). By converting binary vectors into integers, the logic function* $f_b(X)$ *is converted into the integer function* $f(X)$ *in Table 1(c). In this paper, the* 3*-bit precision* $\sin(X)$ *denotes the integer function* $f(X)$ *in Table 1(c). (End of Example)*

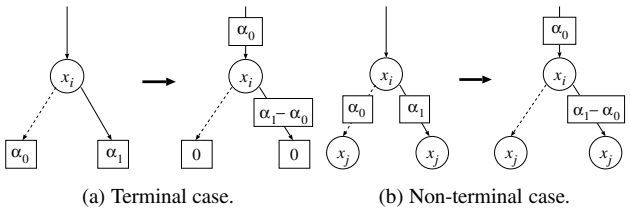
3. Edge-Valued Binary Decision Diagram

This section defines an Mp-monotone increasing function, and derives an upper bound on the number of nodes in an EVBDD for an Mp-monotone increasing function. Experimental results in this section show that EVBDDs for elementary functions are more compact than BMDs for them.

Definition 4 *A binary decision diagram (BDD) [1] is a rooted directed acyclic graph (DAG) representing a logic function. The BDD is obtained by repeatedly applying the Shannon expansion to the logic function. Each function, including the original function and all sub-functions resulting from applying the Shannon expansion, is represented by a non-terminal node, unless that function is a trivial function,* 0 *or* 1*, in which case, it is represented by a terminal node. Each non-terminal node has two outgoing edges,* 0*-edge and* 1*-edge, that correspond to the values of the input variables. Both terminal nodes have no outgoing edges.*

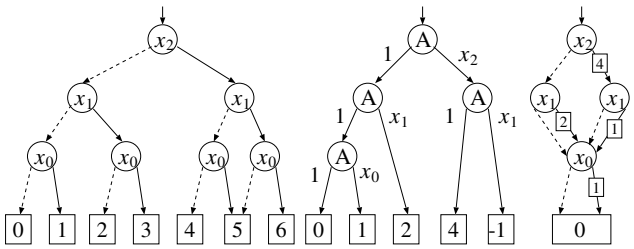
Definition 5 *A* ***multi-terminal BDD (MTBDD)*** *[4] is an extension of the BDD, and represents an integer function. In the MTBDD, the terminal nodes are labeled by integers.*

Definition 6 *An* ***edge-valued BDD (EVBDD)*** *[8] is an extension of the BDD, and represents an integer function. An EVBDD consists of only one terminal node representing* 0 *and non-terminal nodes with* 1*-edges having integer weights. In a reduced EVBDD, each node represents a different sub-function. An EVBDD is obtained 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. In the EVBDD,* 0*-edges (dashed lines) always have zero weights, and the incoming edge into the root node can have a non-zero weight.*



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

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



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

Figure 2. Three types of decision diagrams for the 3-bit precision $\sin(X)$.

Definition 7 *A binary moment diagram (BMD) [2] is a rooted DAG representing an integer function. The BMD is obtained by repeatedly applying the arithmetic transform expansion* $f = f_0 + x_i(f_1 - f_0)$ *to the integer 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.*

For more detail on these decision diagrams, refer to [13].

Example 2 *Fig. 2(a), (b) and (c) show the MTBDD, the BMD, and the EVBDD for the* 3*-bit precision* $\sin(X)$ *in Table 1(c). 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. And, in Fig. 2(b), 'A' denotes the arithmetic transform expansion. In the MTBDD, to evaluate the function, we traverse the MTBDD from the root node to a terminal node according to the input values,*

Table 2. Upper bounds on the number of nodes in MTBDDs and EVBDDs for n-bit precision Mp-monotone increasing functions.

Precision n	MTBDD	EVBDD		
		M1 ($p=1$)	M2 ($p=2$)	M3 ($p=3$)
16	131,071	8,327	10,406	16,450
17	262,143	16,519	18,598	32,833
18	524,287	32,903	34,982	49,217
19	1,048,575	65,670	67,750	81,985
20	2,097,151	98,438	133,286	147,521
21	4,194,303	163,974	264,358	278,593
22	8,388,607	295,046	526,502	540,737
23	16,777,215	557,190	1,050,790	1,065,025
24	33,554,431	1,081,478	2,099,366	2,113,601

and obtain the function value (an integer) 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, we obtain the function value as the sum of the weights for the edges traversed from the root node to the terminal node. *(End of Example)*

Definition 8 *Let I be a set of integers including 0. An integer function $f(X) : I \to Z$ such that $0 \le f(X+1) - f(X) \le p$ and $f(0) = 0$ is an **Mp-monotone increasing function** on I. 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.*

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

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

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

Example 3 *Table 2 compares the upper bounds on the number of nodes in MTBDDs and EVBDDs for n-bit precision Mp-monotone increasing functions. The upper bound for MTBDDs is $2^{n+1} - 1$ independently of p. On the other hand, the upper bound for EVBDDs decreases with p.* *(End of Example)*

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

(Proof) See Appendix.

Table 3. Numbers of nodes in MTBDDs, BMDs, and EVBDDs for 16-bit precision elementary functions.

Elementary functions	Type of functions	Number of nodes			R_1	R_2
		MTBDD	BMD	EVBDD		
$2^X - 1$	M2	122,659	29,634	3,469	2.8	12
$\frac{1}{\sqrt{X+1}} - \frac{1}{\sqrt{2}}$	M1$^+$	58,412	28,446	2,857	4.9	10
$\ln(X+1)$	M1	100,880	28,442	3,187	3.2	11
$\log_2(X+1)$	M2	122,542	29,553	3,465	2.8	12
$\sqrt{X+1} - 1$	M1	73,406	26,149	2,383	3.2	9
$\frac{2}{X+1} - 1$	M2$^+$	114,093	28,348	4,079	3.6	14
$\sin(X)$	M1	115,450	22,638	2,853	2.5	13
Average		101,063	27,601	3,185	3.3	12

Number of fractional bits for function values is 16.
Domain of the functions is $0 \le X < 1$.
R_1 = (EVBDD) / (MTBDD) $\times$ 100 [%].
R_2 = (EVBDD) / (BMD) $\times$ 100 [%].
Mp^+: the function is a linear transform of an Mp-monotone increasing function.

Example 4 *The 3-bit precision $\sin(X) = [0,1,2,3,4,5,5,6]^t$ is an M1-monotone increasing function. The 3-bit precision $\frac{1}{X+1} = [8,7,6,6,5,5,5,4]^t$ is a linear transformation of the M1-monotone increasing function $f(X) = [0,1,2,2,3,3,3,4]^t : -1 \times f(X) + 8$.* *(End of Example)*

Table 3 compares the numbers of nodes in MTBDDs, BMDs, and EVBDDs for 16-bit precision elementary functions. In the column labeled with "Type of functions" of Table 3, Mp denotes an Mp-monotone increasing function, while Mp^+ denotes a linear transformation of an Mp-monotone increasing function.

Standard elementary functions such as trigonometric and logarithmic functions on $0 \le X < 1$ are M1 or M2-monotone increasing functions. Thus, as shown in Table 2, the upper bounds on the number of nodes in their EVBDDs are small. From Table 3, we can see that the upper bounds are smaller than the numbers of nodes in their MTBDDs and BMDs. When EVBDDs are partitioned into two parts as shown in Fig. A.1, for many elementary functions, 1) the upper part does not form a complete binary tree, and 2) the lower part represents only a subset of the Mp-monotone increasing functions. Therefore, the numbers of nodes in EVBDDs for elementary functions are smaller than the upper bounds, and they are much smaller than the numbers of nodes in the MTBDDs and BMDs.

4. Edge-Valued MDD

This section expands EVBDDs shown in the previous section into multi-valued decision diagrams to represent elementary functions more compactly.

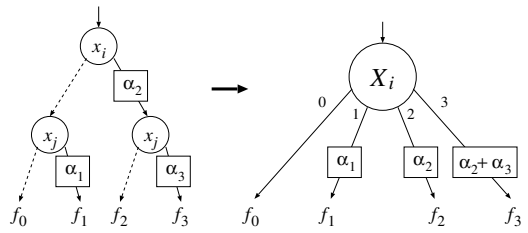
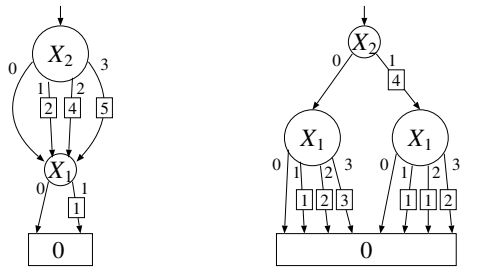


Figure 3. Conversion of EVBDD nodes into an EVMDD node.



(a) $X_1 = (x_0)_2, X_2 = (x_2\ x_1)_2$. (b) $X_1 = (x_1\ x_0)_2, X_2 = (x_2)_2$.

Figure 4. EVMDDs for the 3-bit precision $\sin(X)$.

Definition 9 *For an n-bit precision integer X, let $\{X\}$ be the set of binary variables in X. If $\{X\} = \{X_1\} \cup \{X_2\} \cup \ldots \cup \{X_u\}$, $\{X_i\} \neq \phi$, and $\{X_i\} \cap \{X_j\} = \phi$ $(i \neq j)$, then $(X_1, X_2, \ldots, X_u)$ is a **partition of** X. Each X_i is called a **super variable**. Let $|X_i| = k_i$ and $k_1 + k_2 + \ldots + k_u = n$. Then, by considering each super variable as a multi-valued variable, an integer function $f(X) : Z \rightarrow Z$ can be converted into a **multi-valued input integer function** $f(X_1, X_2, \ldots, X_u) : P_1 \times P_2 \times \ldots \times P_u \rightarrow Z$, where $P_i = \{0, 1, 2, \ldots, 2^{k_i} - 1\}$.*

Definition 10 *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 [6]. It consists of terminal nodes representing function values and non-terminal nodes representing multi-valued variables. Each non-terminal node has multiple 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** [11, 12]. In the following, the heterogeneous MDD is simply denoted by the MDD.*

Definition 11 *An **edge-valued MDD (EVMDD)** is an extension of the 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. 3, an EVMDD is obtained by grouping non-terminal nodes in an EVBDD according to the partition of X.

Example 5 *Fig. 4 shows EVMDDs for the 3-bit precision $\sin(X)$. In Fig. 4(a), $X = (x_2\ x_1\ x_0)_2$ is partitioned into $X_1 = (x_0)_2$ and $X_2 = (x_2\ x_1)_2$. On the other hand, in Fig. 4(b), it is partitioned into $X_1 = (x_1\ x_0)_2$ and $X_2 = (x_2)_2$. To obtain the function value $f = 5$ for $X = (101)_2$, in Fig. 4(a), we traverse the EVMDD using $X_1 = 1$ and $X_2 = 2$, and obtain the function value by summing the weights of the traversed edges. In Fig. 4(b), we traverse the EVMDD using $X_1 = 1$ and $X_2 = 1$, and obtain the function value $f = 5$.*
(End of Example)

Table 3 used the number of nodes to compare the sizes of three types of decision diagrams. However, comparing just the number of nodes in an EVMDD with them is nonsense because non-terminal nodes in an EVMDD are obtained by grouping non-terminal nodes in an EVBDD. Thus, this section introduces another measure, memory size, for meaningful comparison of different types of decision diagrams.

Definition 12 *The **memory size of a decision diagram** is the number of words needed to represent all non-terminal nodes in the decision diagram in a memory, where each attribute requires one word.*

Since each non-terminal node in a BMD has three attributes: an index of binary variable used for the arithmetic transform expansion, and two pointers for edges, it requires three words. Thus, the memory size of a BMD is

$$3 \times (\text{Number of non-terminal nodes in the BMD}).$$

Similarly, since each non-terminal node in an EVBDD has four attributes: an index of binary variable used for the Shannon expansion, two pointers for 0 and 1-edges, and a weight of 1-edge, the memory size of an EVBDD is

$$4 \times (\text{Number of non-terminal nodes in the EVBDD}).$$

Each non-terminal node in an EVMDD may have different number of edges, depending on the domain of multi-valued variable represented by the node. Each non-terminal node representing p_i-valued variable has $2p_i$ attributes: an index of p_i-valued variable X_i, p_i pointers for edges, and $p_i - 1$ weights of edges except for 0-edge. When the number of multi-valued variables is u, and each variable X_i is p_i-valued variable, the memory size of an EVMDD is

$$\sum_{i=1}^{u} 2p_i \times w_i,$$

where w_i denotes the number of non-terminal nodes for X_i in the EVMDD.

Table 4. Numbers of non-terminal nodes and memory sizes of BMDs, EVBDDs, and EVMDDs for 16-bit precision elementary functions.

Elementary functions	No. of non-terminal nodes			Memory size			Ratio [%]	
	BMD	EVBDD	EVMDD	BMD	EVBDD	EVMDD	MDD / BMD	MDD / BDD
$2^X - 1$	29,486	3,468	353	88,458	13,872	9,600	11	69
$\frac{1}{\sqrt{X+1}} - \frac{1}{\sqrt{2}}$	28,272	2,856	950	84,816	11,424	7,892	9	69
$\ln(X+1)$	28,277	3,186	1,240	84,831	12,744	9,052	11	71
$\log_2(X+1)$	29,393	3,464	362	88,179	13,856	9,636	11	70
$\sqrt{X+1} - 1$	26,011	2,382	552	78,033	9,528	6,300	8	66
$\frac{2}{X+1} - 1$	28,168	4,078	675	84,504	16,312	10,888	13	67
$\sin(X)$	22,497	2,852	1,091	67,491	11,408	8,456	13	74
Average	27,443	3,184	746	82,330	12,735	8,832	11	69

Number of fractional bits for function values is 16. Domain of the functions is $0 \leq X < 1$.
MDD / BMD = (memory size of EVMDD) / (memory size of BMD) $\times$ 100 [%].
MDD / BDD = (memory size of EVMDD) / (memory size of EVBDD) $\times$ 100 [%].

Example 6 *The memory size of the BMD in Fig. 2(b) is* $3 \times 4 = 12$. *The memory size of the EVBDD in Fig. 2(c) is* $4 \times 4 = 16$. *And, the memory sizes of the EVMDDs in Fig. 4(a) and (b) are* $8 + 4 = 12$, *and* $4 + 2 \times 8 = 20$, *respectively. (End of Example)*

When we represent an integer function $f(X)$ using an EVMDD by a partition of X, the memory size of the EVMDD depends on the partition of X as shown in Example 6. In this paper, to find the partition of X that minimizes the memory size, we used the algorithm for heterogeneous MDDs proposed in [11, 12].

Table 4 compares the numbers of non-terminal nodes and memory sizes of BMDs, EVBDDs, and EVMDDs for 16-bit precision elementary functions. Since non-terminal nodes in EVBDDs have weighted 1-edges, the memory size of each non-terminal node in an EVBDD is larger than that of a non-terminal node in the BMD. However, by using weighted edges, to represent an elementary function, an EVBDD requires many fewer nodes than the BMD, and thus an EVBDD requires smaller memory size than the BMD. In an EVMDD, the memory size of each non-terminal node becomes larger than that of a non-terminal node in the EVBDD, but the number of nodes becomes much smaller than that in the EVBDD. Thus, an EVMDD requires smaller memory size than the EVBDD. Table 4 shows that, on average, EVMDDs require only 11% of memory size needed for BMDs, and only 69% of memory size needed for EVBDDs.

5. Conclusion and Comments

This paper has presented a new class of integer functions, called an Mp-monotone increasing function, and derived an upper bound on the number of nodes in an EVBDD for the Mp-monotone increasing function. The upper bound shows that EVBDDs represent Mp-monotone increasing functions or their linear transformations more compactly than MTBDDs and BMDs when p is small. By converting integer functions into multi-valued input integer functions, and representing them by EVMDDs, memory sizes needed for decision diagrams can be reduced further. Experimental results show that EVMDDs represent elementary functions more compactly than BMDs and EVBDDs.

EVBDDs form a subset of K*BMDs [5] and factored EVBDDs (FEVBDDs) [19] that allow additive and multiplicative edge weights. Thus, K*BMDs and FEVBDDs also represent elementary functions compactly. However, function evaluations using them are more complex than those using EVBDDs. In EVBDDs, we can evaluate functions only by summing the weights of the edges traversed from the root node to the terminal node. This evaluation method can be realized with a simple hardware using a memory and an adder. Since EVMDDs have shorter paths and smaller memory size than EVBDDs, EVMDDs are promising for fast and compact elementary 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 Kitakyushu innovative cluster project, the MEXT Grant-in-Aid for Young Scientists (B), 18700048, 2006, and Hiroshima City University Grant for Special Academic Research (General Studies), 6101, 2006.

References

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

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

[3] M. J. Ciesielski, P. Kalla, Z. Zheng, and B. Rouzeyre, "Taylor expansion diagrams: A compact canonical representation with applications to symbolic verification," *Design, Automation and Test in Europe (DATE2002)*, pp. 285–289, 2002.

[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] R. Drechsler, B. Becker, and S. Ruppertz, "K*BMDs: A new data structure for verification," *European Design & Test Conf.*, pp. 2–8, 1996.

[6] 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.

[7] U. Kebschull, E. Schubert, and W. Rosenstiel, "Multilevel logic synthesis based on functional decision diagrams," *European Conference on Design Automation (EDAC)*, pp. 43–47, 1992.

[8] 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.

[9] D. M. Miller and M. A. Thornton, "QMDD: A decision diagram structure for reversible and quantum circuits," *36th International Symposium on Multiple-Valued Logic*, Singapore, May 17-20, 2006.

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

[11] 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.

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

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

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

[15] 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.

[16] R. Stankovic, T. Sasao, and C. Moraga, "Spectral transform decision diagrams," Chapter 3 in [13].

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

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

[19] S. B. K. Vrudhula, M. Pedram, and Y.-T. Lai, "Edge valued binary decision diagrams," Chapter 5 in [13].

Appendix

Definition A.1 *A two-valued input multi-valued output function $h : B^n \to \{0,1,\ldots,p\}$ such that $h(0,0,\ldots,0) = 0$ is a $(p+1)$-**valued** 0-**preserving function**. This is an extension of the 0-preserving function for logic function [14].*

Lemma A.1 *The number of different n-bit precision Mp-monotone increasing functions is*

$$(p+1)^{2^n-1}.$$

(Proof) Let $h(Y)$ be an n-bit input $(p+1)$-valued 0-preserving function, where $Y = (y_{n-1}\, y_{n-2} \ldots y_0)_2$. For each $h(Y)$, there exists an n-bit precision Mp-monotone increasing function $f(X)$. And, any $f(X)$ can be derived from the corresponding $h(Y)$ as

$$f(X) = \sum_{Y=0}^{X} h(Y).$$

The number of different h's is

$$(p+1)^{2^n-1}.$$

Therefore, we have the lemma. ∎

Example A.1 *Let H be the function-vector of a 2-bit input two-valued 0-preserving function h. And, let F be the function-vector of a 2-bit precision M1-monotone increasing function f. The following shows all possible F's and the corresponding H's.*

H	F	H	F
$[0,0,0,0]^t$	$[0,0,0,0]^t$	$[0,0,0,1]^t$	$[0,0,0,1]^t$
$[0,0,1,0]^t$	$[0,0,1,1]^t$	$[0,0,1,1]^t$	$[0,0,1,2]^t$
$[0,1,0,0]^t$	$[0,1,1,1]^t$	$[0,1,0,1]^t$	$[0,1,1,2]^t$
$[0,1,1,0]^t$	$[0,1,2,2]^t$	$[0,1,1,1]^t$	$[0,1,2,3]^t$

As shown above, the number of the different 2-bit precision M1-monotone increasing functions is $2^{2^2-1} = 8$. *(End of Example)*

Example A.2 *Let H be the function-vector of a 2-bit input three-valued 0-preserving function h. And, let F be the function-vector of a 2-bit precision M2-monotone increasing function f. The following shows all possible F's and the corresponding H's.*

H	F	H	F	H	F
$[0,0,0,0]^t$	$[0,0,0,0]^t$	$[0,0,0,1]^t$	$[0,0,0,1]^t$	$[0,0,0,2]^t$	$[0,0,0,2]^t$
$[0,0,1,0]^t$	$[0,0,1,1]^t$	$[0,0,1,1]^t$	$[0,0,1,2]^t$	$[0,0,1,2]^t$	$[0,0,1,3]^t$
$[0,0,2,0]^t$	$[0,0,2,2]^t$	$[0,0,2,1]^t$	$[0,0,2,3]^t$	$[0,0,2,2]^t$	$[0,0,2,4]^t$
$[0,1,0,0]^t$	$[0,1,1,1]^t$	$[0,1,0,1]^t$	$[0,1,1,2]^t$	$[0,1,0,2]^t$	$[0,1,1,3]^t$
$[0,1,1,0]^t$	$[0,1,2,2]^t$	$[0,1,1,1]^t$	$[0,1,2,3]^t$	$[0,1,1,2]^t$	$[0,1,2,4]^t$
$[0,1,2,0]^t$	$[0,1,3,3]^t$	$[0,1,2,1]^t$	$[0,1,3,4]^t$	$[0,1,2,2]^t$	$[0,1,3,5]^t$
$[0,2,0,0]^t$	$[0,2,2,2]^t$	$[0,2,0,1]^t$	$[0,2,2,3]^t$	$[0,2,0,2]^t$	$[0,2,2,4]^t$
$[0,2,1,0]^t$	$[0,2,3,3]^t$	$[0,2,1,1]^t$	$[0,2,3,4]^t$	$[0,2,1,2]^t$	$[0,2,3,5]^t$
$[0,2,2,0]^t$	$[0,2,4,4]^t$	$[0,2,2,1]^t$	$[0,2,4,5]^t$	$[0,2,2,2]^t$	$[0,2,4,6]^t$

As shown above, the number of the different 2-bit precision M2-monotone increasing functions is $3^{2^2-1} = 27$. *(End of Example)*

Definition A.2 *A shared EVBDD (SEVBDD) is an extension of the EVBDD, and it has multiple root nodes to represent multiple integer functions. The SEVBDD is obtained by sharing equivalent sub-graphs in EVBDDs for the integer functions.*

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

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

(Proof) We prove the lemma by the mathematical induction. When $k = 1$, the function-vectors of all the Mp-monotone increasing functions are $F = [0,0]^t, F = [0,1]^t, \ldots$, and $F = [0,p]^t$. Since $F = [0,0]^t$ is the constant function 0, there is no non-terminal node representing it. As for the other p function-vectors, there exists a non-terminal node for each function-vector. Thus, when $k = 1$, the lemma holds. Next, we assume that the lemma holds when $k = n$. And, we prove that the lemma holds when $k = n+1$.

Each non-terminal node in an EVBDD represents the following expansion [8]: $f = \overline{x_i} f_0 + x_i(f_1 + \alpha)$. When $f_0 = f_1 + \alpha$, however, the non-terminal node for x_i is eliminated because of the reduction rules for EVBDD. Conversely, the non-terminal node for x_i is not eliminated when $f_0 \neq f_1 + \alpha$. When f is an $(n+1)$-bit precision Mp-monotone increasing function except for the constant function 0, $f_0 \neq f_1 + \alpha$ holds in the expansion with respect to x_n. From Lemma A.1, the number of different $(n+1)$-bit precision Mp-monotone increasing functions except for the constant function 0 is $(p+1)^{2^{n+1}-1} - 1$. Thus, in an SEVBDD, there exist $(p+1)^{2^{n+1}-1} - 1$ non-terminal nodes representing the expansions with respect to x_n. Since f_0's and f_1's produced by these expansions are the n-bit precision Mp-monotone increasing functions, from the assumption for $k = n$, we have

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

Therefore, when $k = n+1$, the number of non-terminal nodes is as follows:

$$\begin{aligned} &\eta(n,p) + (p+1)^{2^{n+1}-1} - 1 \\ &= \sum_{i=1}^{n} (p+1)^{2^i-1} - n + (p+1)^{2^{n+1}-1} - 1 \\ &= \sum_{i=1}^{n+1} (p+1)^{2^i-1} - (n+1) = \eta(n+1,p) \end{aligned}$$

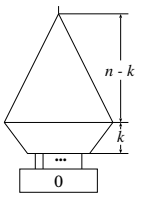


Figure A.1. Partition of EVBDD.

Therefore, the lemma holds. ∎

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 k-bit precision Mp-monotone increasing functions, and the upper part represents the function which chooses one from them. 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

$$2^{n-k} - 1. \tag{A.1}$$

The lower part has the maximum number of nodes when it represents all the k-bit precision Mp-monotone increasing functions. From Lemma A.2, the maximum number of nodes in the lower part is

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

From (A.1) and (A.2), the number of non-terminal nodes in the EVBDD for $f(X)$ is at most

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

By adding one terminal node to this, we have the theorem. The number of Mp-monotone increasing functions which can be represented in the lower part is $(p+1)^{2^k-1}$. It does not exceed the number of functions which can be chosen by the upper part: 2^{n-k}. Therefore, we have the relation:

$$2^{n-k} \geq (p+1)^{2^k-1}.$$

∎

Proof for Lemma 1 In EVBDDs, the sum of weights of the traversed edges shows the function value. Thus, in the EVBDD for $f(X)$, multiplying each weight by a and adding b to the weight of edge to the root node can produce the EVBDD for $g(X)$. This conversion of EVBDDs does not change the number of nodes. ∎