39th International Symposium on Multiple-Valued Logic

Floating-Point Numerical Function Generators Using EVMDDs for Monotone Elementary Functions

Shinobu Nagayama[1] Tsutomu Sasao[2] Jon T. Butler[3]

[1] Department of Computer and Network Engineering, Hiroshima City University
[2] Department of Computer Science and Electronics, Kyushu Institute of Technology
[3] Department of Electrical and Computer Engineering, Naval Postgraduate School

Abstract

This paper proposes a design method for floating-point numerical function generators (NFGs) using multi-valued decision diagrams (MDDs). Our method applies to monotone elementary functions in which real values are converted into integer values that are represented by edge-valued MDDs (EVMDDs). We show that EVMDDs use fewer nodes by one or two orders of magnitude than two other types of decision diagrams, MTBDDs and BMDs. EVMDDs produce fast and compact floating-point NFGs for real-valued elementary functions, with a speed improvement of 86% *over a recently proposed floating-point implementation [4].*

1. Introduction

Elementary functions are widely used in many applications, such as computer graphics and digital signal processing [14]. Various design methods for numerical function generators (NFGs) have been proposed. Among them, fixed-point representation of elementary functions is usually adopted [3,11,18,21,23]. But, for elementary functions with a wide domain and range, a fixed-point representation requires a large number of bits to represent a large real value. This produces large NFGs. To represent a large real value with fewer bits, floating-point representation is often used. An IEEE standard for real values exists [7]. However, floating-point representation tends to produce complex and slow NFGs. Thus, the design of floating-point NFGs is especially hard, and only design methods for some elementary functions are known [4, 5, 8, 22, 26]. Since these design methods are intended only for specific functions, different functions need different design methods and architecturers. This paper proposes a systematic design method for floating-point NFGs for a wide range of elementary functions.

For design of typical digital circuits, systematic methods using various decision diagrams such as binary decision diagrams (BDDs) [6,12] have been established [13,19,25,27]. Thus, we consider a design method using decision diagrams for floating-point NFGs. Since decision diagrams are not robust enough to represent all classes of the functions compactly, choosing a decision diagram appropriate to a given class of functions is important. Although decision diagrams appropriate to a fixed-point representation of elementary functions have been presented [16, 17, 20, 24], as far as we know, no study on graph-based representations for floating-point elementary functions has been presented. As shown in Fig. 1, fixed-point and floating-point representations convert an elementary function to quite different integer functions. Even for the simple linear function such as $f(X) = 5X + 13.7$, floating-point representation converts to a complex integer function. Thus, floating-point elementary functions should be considered as a different class of functions than fixed-point ones.

This paper considers graph-based representations appropriate to floating-point elementary functions. To analyze their complexities, this paper introduces a *transition point* and a new class of integer functions, an *Mp-monotone increasing function with transition points*. Theoretical analysis and experimental results show that edge-valued multi-valued decision diagrams (EVMDDs) can compactly represent both fixed-point and floating-point elementary functions. And, by using EVMDDs, we can automatically generate fast and compact floating-point NFGs for a wide range of elementary functions.

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

Definition 2 *The* ***n-bit precision binary floating-point representation*** *of a number X is a binary n-tuple*

$$X = (s, \quad e_{a-1}, e_{a-2}, \ldots, e_0, \quad d_{b-1}, d_{b-2}, \ldots, d_0)_2,$$

where $s \in B$ is the ***sign bit***, *$E = (e_{a-1}, e_{a-2}, \ldots, e_0)_2$ is the* ***exponent***, *and $D = (d_{b-1}, d_{b-2}, \ldots, d_0)_2$ is the* ***significand***.

0195-623X/09 $25.00 © 2009 IEEE
DOI 10.1109/ISMVL.2009.24

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2009**	2. REPORT TYPE	3. DATES COVERED

4. TITLE AND SUBTITLE **Floating-Point Numerical Function Generators Using EVMDDs for Monotone Elementary Functions**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Department of Electrical and Computer Engineering,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S) 11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

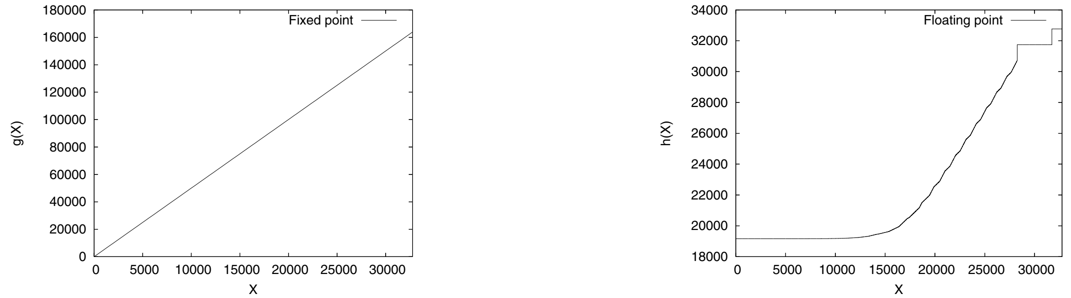
13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper proposes a design method for floating-point numerical function generators (NFGs) using multi-valued decision diagrams (MDDs). Our method applies to monotone elementary functions in which real values are converted into integer values that are represented by edgevalued MDDs (EVMDDs). We show that EVMDDs use fewer nodes by one or two orders of magnitude than two other types of decision diagrams, MTBDDs and BMDs. EVMDDs produce fast and compact floating-point NFGs for real-valued elementary functions, with a speed improvement of 86% over a recently proposed floating-point implementation [4].

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**7**	



(a) Fixed-point representation (integer bits: 15, fractional bits: 0). Domain: $0 \leq X \leq 32,767$ (smallest positive value: 1).

(b) Floating-point representation (exponent: 5 bits, significand: 10 bits). Domain: $0 \leq X < \infty$ (max: 131,040, smallest positive value: 2^{-24}).

Figure 1. Fixed-point and floating-point representations (integer functions) of $f(X) = 5X + 13.7$.

Table 1. Floating-point representation of X with a-bit exponent and b-bit significand.

Type	Exponent E	Significand D	Value of X
Zero	$(0,0,\ldots,0)_2$	$(0,0,\ldots,0)_2$	0
Subnormal no.	$(0,0,\ldots,0)_2$	$\neq (0,0,\ldots,0)_2$	$(-1)^s \times 2^{-E_s} \times 0.D$
Infinity	$(1,1,\ldots,1)_2$	$(0,0,\ldots,0)_2$	$(-1)^s \times \infty$
Not a no. (NaN)	$(1,1,\ldots,1)_2$	$\neq (0,0,\ldots,0)_2$	NaN
Normal no.	Others		$(-1)^s \times 2^{E-E_n} \times 1.D$

Bias value for subnormal numbers: $E_s = 2^{a-1} - 2$

Bias value for normal numbers: $E_n = 2^{a-1} - 1$

a and b are the numbers of bits for the exponent and the significand respectively, and $n = a + b + 1$. The value of X is shown in Table 1. When $|X| < 2^{2-2^{a-1}}$, X is a ***subnormal number****, in which the exponent E is biased by $E_s = 2^{a-1} - 2$, and the significand D represents only fractional bits of a fixed-point value smaller than 1. When $2^a \leq |X|$, X is* ***infinity****. When X cannot be defined as a number, X is represented as* ***not a number (NaN)****. In other cases, X is a* ***normal number****, in which the exponent E is biased by $E_n = 2^{a-1} - 1$, and the significand D represents only fractional bits of a fixed-point value that is larger than or equal to 1 and smaller than 2.*

According to IEEE Standard 754-2008 [7], ***half (16-bit) precision*** *has $a = 5$ and $b = 10$,* ***single (32-bit) precision*** *has $a = 8$ and $b = 23$,* ***double (64-bit) precision*** *has $a = 11$ and $b = 52$, and* ***quad (128-bit) precision*** *has $a = 15$ and $b = 112$.*

Note that an n-bit precision floating-point representation converts a real function into an n-input n-output logic function. The logic function, in turn, can be converted into an integer function by considering binary vectors as integers. That is, we can convert a real function into an integer function: $P_n \rightarrow P_n$, where $P_n = \{0, 1, \ldots, 2^n - 1\}$. This is what was done in Fig. 1 for the real function $f(X) = 5X + 13.7$. In this paper, elementary functions are converted into integer functions by using a floating-point representation, unless stated otherwise. And, for simplicity, each bit in the floating-point representation of X is denoted by x_i, where x_0 is the least significant bit.

Table 2. Tables for the 8-bit precision (3-bit exponent, 4-bit significand) floating-point $\sqrt{X}$.

(a) Table for $\sqrt{X}$.

X	$\sqrt{X}$
0.000000	0.000000
0.015625	0.125000
0.031250	0.171875
0.046875	0.218750
0.062500	0.250000
0.078125	0.281250
0.093750	0.312500
0.109375	0.328125
⋮	⋮

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

X	$f_b(X)$
0 000 0000	0 000 0000
0 000 0001	0 000 1000
0 000 0010	0 000 1011
0 000 0011	0 000 1110
0 000 0100	0 001 0000
0 000 0101	0 001 0010
0 000 0110	0 001 0100
0 000 0111	0 001 0101
⋮	⋮

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

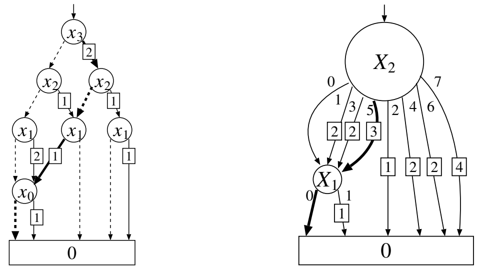
X	$f(X)$
0	0
1	8
2	11
3	14
4	16
5	18
6	20
7	21
⋮	⋮

Example 1 *Table 2 (a) is the function table for $\sqrt{X}$. The 8-bit precision (3-bit exponent and 4-bit significand) floating-point representation of this function is the logic function $f_b(X)$ in Table 2 (b). By converting binary vectors into integers, we have the integer function $f(X)$ of $f_b(X)$ in Table 2 (c). That is, our 8-bit precision floating-point representation of $f(X) = \sqrt{X}$ corresponds to the integer function of Table 2 (c).* *(End of Example)*

2.2. Edge-Valued MDDs

Definition 3 *An* **edge-valued BDD (EVBDD)** *[10] is a variant of the BDD, and represents an integer function. The EVBDD is obtained by repeatedly applying the expansion $f = \overline{x_i} f_0 + x_i(f_1' + \alpha)$ to the integer 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. The incoming edge into the root node can have a non-zero weight. The output (integer) value of an EVBDD is the sum of the weights associated with the path taken from the root node to the terminal node.*

Definition 4 *For a set of n binary variables $\{X\}$, 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 multi-valued variable, an integer function $f(X) : Z \rightarrow Z$ can be converted into a* ***multi-valued input integer function*** *$f(X_u, X_{u-1}, \ldots, X_1) : P_u \times P_{u-1} \times \ldots \times P_1 \rightarrow Z$, where $P_i = \{0, 1, 2, \ldots, 2^{k_i} - 1\}$.*



(a) EVBDD

(b) EVMDD

Figure 2. EVBDD and EVMDD for an integer function.

Definition 5 *An **edge-valued MDD (EVMDD)** [16] is an extension of the MDD [9,15], and represents a multi-valued input integer function. It consists of one terminal node representing* 0 *and non-terminal nodes. Edges have integer weights. Edges labeled by a logic* 0 *have integer* 0 *weight.*

Example 2 *Fig. 2 (a) and (b) show the EVBDD and the EVMDD, respectively, for the same integer function. In Fig. 2 (a), dashed lines and solid lines denote* 0*-edges and weighted* 1*-edges, respectively. In the EVMDD, the set of binary variables* $\{X\}$ *is partitioned into* $\{X_2\} = \{x_3, x_2, x_1\}$ *and* $\{X_1\} = \{x_0\}$*. To obtain the function value* 3 *for* $X = (1,0,1,0)_2$*, we traverse the EVBDD or the EVMDD from the root node to the terminal node according to the input values, and obtain the function value as the sum of the weights for the traversed edges. Note that we traverse the EVMDD using* $X_2 = 5$ *and* $X_1 = 0$*. (End of Example)*

3. Graph-Based Representation of Floating-Point Elementary Functions

3.1. Transition Points and Mp-Monotone Increasing Functions

Definition 6 *An integer function* $f(X)$ *such that* $0 \leq f(X+1) - f(X) \leq p$ *and* $f(0) = 0$ *is an **Mp-monotone increasing function**. That is, for an Mp-monotone increasing function* $f(X)$*,* $f(0) = 0$*, and increasing* X *by* 1 *increases the value of* $f(X)$ *by at most* p*.*

Definition 7 *Given an integer* p *and an integer function* $f(X)$*, a point* T *that satisfies* $f(T+1) - f(T) < 0$ *or* $p < f(T+1) - f(T)$ *is a **transition point**.*

Definition 8 *Given an integer* p*, an integer function* $f(X)$ *that satisfies* $0 \leq f(X+1) - f(X) \leq p$ *and* $f(0) = 0$ *for all* X *except for* k *transition points is an **Mp-monotone increasing function with** k **transition points**. That is, for an Mp-monotone increasing function with* k *transition points* $f(X)$*,* $f(0) = 0$*, and for all* X *except for* k *transition points, the increment of* X *by one increases the value of* $f(X)$ *by at most* p*.*

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

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

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

From Theorem 1, if the number of transition points k increases by 1, the number of nodes in the EVBDD increases by at most l. Thus, when k is small, an Mp-monotone increasing function with k transition points can be represented by an EVBDD with the size comparable to an EVBDD for Mp-monotone increasing function.

Corollary 1 *Let* $f(X)$ *be an Mp-monotone increasing function with* k *transition points, and let* $g(X)$ *be an affine 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.*

3.2. Conversion of Elementary Functions to Mp-Monotone Increasing Functions

Unlike the integer function of fixed-point representation, the integer function of floating-point representation often has discontinuities, as shown in Fig. 1. As shown in Fig. 3, both the exponent E and significand D can either oscillate or exhibit sharp transitions or both, even if the original function has only smooth transitions. Floating-point representation with a sign bit is more complex. Thus, analysis of integer functions obtained by floating-point representation of elementary functions is very difficult. However, by considering transition points defined in the previous subsection, we can convert various elementary functions into Mp-monotone increasing functions, and analyze the class of functions easily. In the following, we explain how to convert elementary functions into Mp-monotone increasing functions.

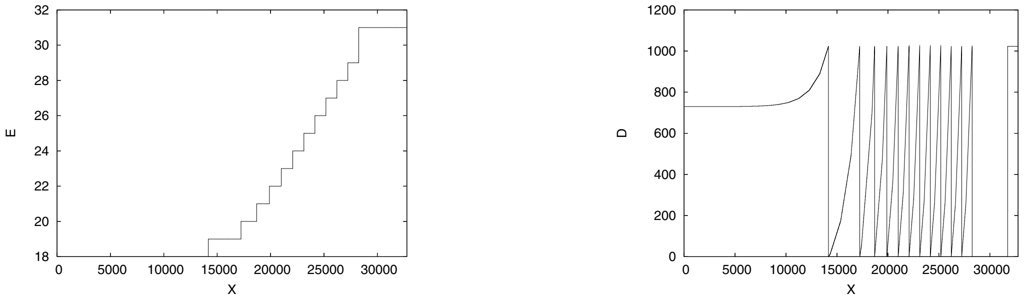
First, we consider the elementary functions whose domain includes negative numbers. To accommodate negative floating-point numbers, a sign bit is needed. Thus, for integer functions $f(X)$ obtained by floating-point representation, the magnitude relation of X for negative numbers becomes the inverse of the original one, and monotonicity of the original functions can be lost. To preserve monotonicity in the original functions, instead of $f(X)$, we use $f(Y)$, where

$$Y = (\overline{x_{n-1}}, x_{n-2} \oplus \overline{x_{n-1}}, x_{n-3} \oplus \overline{x_{n-1}}, \ldots, x_0 \oplus \overline{x_{n-1}})_2, \qquad (2)$$

for $X = (x_{n-1}, x_{n-2}, \ldots, x_0)_2$. Similarly, elementary functions whose range (function values) includes negative numbers are converted into the integer functions $g(X)$:

$$g = (\overline{f_{n-1}}, f_{n-2} \oplus f_{n-1}, f_{n-3} \oplus f_{n-1}, \ldots, f_0 \oplus f_{n-1})_2, \qquad (3)$$

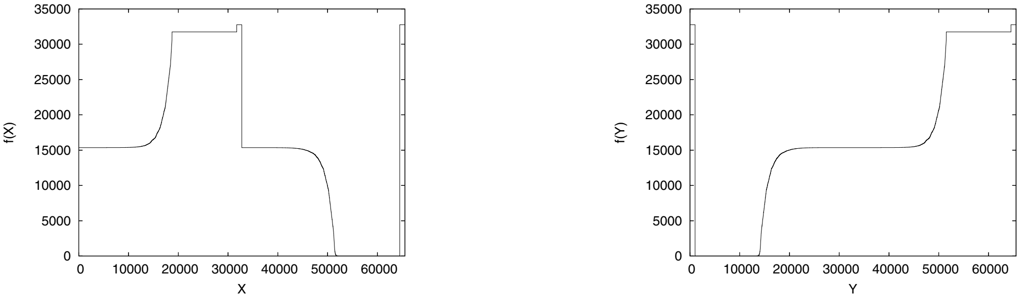
where each f_i denotes each bit in floating-point representation of function values $f(X)$.



(a) Exponent E of $f(X)$.

(b) Significand D of $f(X)$.

Figure 3. Half precision floating-point representation of $f(X) = 5X + 13.7$ ($0 \leq X < \infty$).



(a) Floating-point representation $f(X)$.

(b) Integer function $f(Y)$ using (2).

Figure 4. Conversion of $\exp(X)$ ($-\infty < X < \infty$) into integer functions.

Table 3. Class of half precision floating-point elementary functions.

Functions $f(X)$	p	k	Theorem 1	Upper bound only for Mp [16]
$5X+13.7$	2	1,306	14,324	10,406
$\sin^{-1}(X)$	1	507	5,752	4,231
$\tan^{-1}(X)$	1	83	4,480	4,231
$\exp(X)$	8	483	18,086	17,120
$\ln(X)$	7	397	9,504	8,710
$1/X$	2	141	6,733	6,310
$\sqrt{X}$	1	514	5,773	4,231

k: Number of transition points.
Upper bound only for Mp: Upper bound on the number of nodes for Mp-monotone increasing function without transition points.

Example 3 *Fig. 4 (a) shows the half precision floating-point representation $f(X)$ of $\exp(X)$. Since the domain includes negative numbers, $f(X)$ is not monotone even though the original function $\exp(X)$ is monotone. On the other hand, the integer function $f(Y)$ using (2) is monotone, except for a few transition points as shown in Fig. 4 (b). (End of Example)*

Second, we consider the value of p, used in the Mp-monotone increasing function representation. Larger p yields fewer transition points k, while smaller p yields more values of k. Thus, choosing a p appropriate to a given function is important. Since Theorem 1 holds for any p, we choose the p that makes (1) minimum to obtain a tight upper bound on the number of nodes.

Table 3 shows for various floating-point elementary functions, the appropriate value of p, the number of transition points k, and our upper bound on the number of nodes in an EVBDD for the Mp-monotone increasing function with k transition points. And, for comparison, this table shows the upper bound on the number of nodes in an EVBDD for the Mp-monotone increasing function *without transition points*. It has been derived to analyze the fixed-point elementary functions [16]. Note that, in this table, floating-point representations of the functions whose domain includes only non-negative numbers and the *odd functions* satisfying $f(-X) = -f(X)$ have no sign bit in X (i.e., $n = 15$). And, monotone decreasing functions are converted into monotone increasing functions using the affine transformation (Corollary 1).

This table suggests that many elementary functions can be converted into Mp-monotone increasing functions with a small number of transition points and a small p, and can be represented compactly by an EVBDD with the size comparable to an EVBDD for Mp-monotone increasing function.

Table 4 compares the number of nodes in MTBDDs [2], BMDs [1], and EVBDDs for half precision floating-point elementary functions. From this table, we can see that EVBDDs have fewer nodes by one or two orders of magnitude than MTBDDs and BMDs. In addition to the functions shown in Table 4, we compared the size of DDs for other 18 elementary functions. For all the functions we investigated, the size of the EVBDDs was smaller than the MTBDDs and the BMDs.

As shown in [16], by converting EVBDDs into EVMDDs, we can further reduce memory size of deci-

Table 4. Number of nodes in DDs for half precision floating-point elementary functions.

Function	Number of nodes			R_1	R_2
$f(X)$	MTBDD	BMD	EVBDD		
$5X + 13.7$	49,669	12,470	726	1	6
$\sin^{-1}(X)$	30,652	3,618	397	1	11
$\tan^{-1}(X)$	33,245	10,039	926	3	9
$\exp(X)$	23,035	23,121	2,356	10	10
$\ln(X)$	27,838	26,173	2,500	9	10
$1/X$	52,750	4,259	567	1	13
$\sqrt{X}$	40,145	5,619	518	1	9
Average	36,762	12,186	1,143	4	10

$R_1 = (EVBDD)/(MTBDD) \times 100$
$R_2 = (EVBDD)/(BMD) \times 100$
Variable orders of DDs are $x_{n-1}, x_{n-2}, \ldots, x_0$.

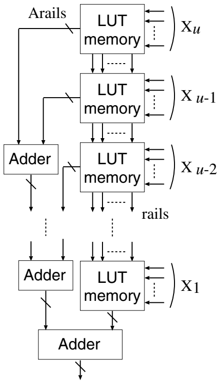


Figure 5. Architecture for floating-point NFGs based on EVMDDs.

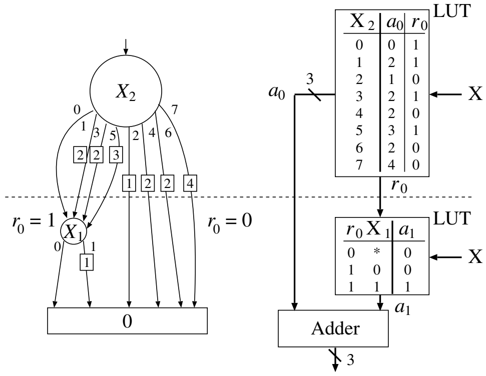
sion diagrams, and obtain more compact representations. In the next section, we present an architecture and a design method for floating-point NFGs taking advantage of EVMDDs.

4. Floating-Point Function Generators

4.1. Architecture for Floating-Point NFGs

In EVMDDs, function values can be obtained by traversing the EVMDDs from the root node to the terminal node, and accumulating the weights of traversed edges. Thus, we can compute function values by the architecture consisting of only memories and integer adders shown in Fig. 5. In the architecture shown in Fig. 5, an EVMDD is decomposed with each multi-valued variable X_i, and stored in LUT memories for X_i. The interconnecting lines between adjacent LUT memories are called *rails*, and they represent sub-functions in the EVMDD. And, the outputs from each LUT memory to an adder (called *Arails*) represent the sum of edge weights.

Example 4 *By realizing the EVMDD in Fig. 2 (b) with the architecture in Fig. 5, we have the circuit shown in Fig. 6. In this figure, r_0 denotes the rails that represent sub-functions*



X_2	a_0	r_0
0	0	1
1	2	1
2	1	0
3	2	1
4	2	0
5	3	1
6	2	0
7	4	0

r_0	X_1	a_1
0	*	0
1	0	0
1	1	1

Figure 6. Example of floating-point NFGs based on EVMDDs.

in the EVMDD, and a_0 and a_1 denote the Arails that represent the sum of edge weights. Since this EVMDD has only two sub-functions with respect to X_2, r_0 takes a value 0 or 1 to represent the sub-functions. When $r_0 = 0$, the represented sub-function is a constant function 0, and thus the edge weight a_1 is 0 independently of the value of X_1. On the other hand, when $r_0 = 1$, a_1 depends on the value of X_1. (End of Example)

The proposed NFGs have the following features:

1. Since the proposed NFG just traverses an EVMDD and computes the sum of edge weights (integers), it requires only the integer adders to compute function values of a floating-point function. That is, it requires neither the *rounding circuit* nor the *normalization circuit*.
2. By changing the data for the LUT memories, a wide range of elementary functions can be realized by the same architecture.
3. Since the proposed NFG directly realizes the function table of a floating-point function using an EVMDD, it is more accurate than existing NFGs using polynomial approximation [4, 5, 8, 22, 26].
4. The proposed NFG is suitable for pipeline processing, and thus it can achieve a high throughput.

4.2. Design Method for Floating-Point NFGs Using EVMDDs

For a given elementary function, its domain, and precision n, we can automatically generate the circuit in Fig. 5. First, convert a given elementary function represented in n-bit precision floating-point into an integer function, next represent the integer function using an EVMDD, and finally generate HDL code for the circuit in Fig. 5 from the EVMDD. Memory size and delay time of the generated circuit mainly depend on size and path length of EVMDD. Therefore, the memory minimization algorithm and the APL minimization algorithm for MDDs [15] are useful to produce fast and compact floating-point NFGs.

As mentioned in Section 3.2, the elementary functions whose domain or range includes negative numbers are converted into integer functions $f(Y)$ or $g(X)$ using (2) or (3).

Table 5. Comparison results with the existing floating-point NFG.

Floating-point NFGs	Throughput [MHz]	Delay [nsec.]
Circuit for $\ln(X)$ [4]	100	39
Our NFG using EVMDD	186	27

$f(Y)$ or $g(X)$ is realized with the architecture in Fig. 5. To restore $f(Y)$ to the original function $f(X)$, our synthesis system for NFG automatically inserts the circuit realizing (3) in the inputs of the NFG in Fig. 5. Similarly, to restore $g(X)$ to $f(X)$, our synthesis system inserts the circuit realizing (2) in the outputs of the NFG. For the odd functions satisfying $f(-X) = -f(X)$, $f_s = x_s$ is realized, where f_s and x_s denote the sign bits of the function value and the input variable X, respectively. And, for the functions whose domain includes only non-negative numbers, such as $\sqrt{X}$ and $\ln(X)$, the circuit that produces *NaN* when $x_s = 1$ is inserted in the outputs of the NFG automatically.

4.3. FPGA Implementation Results

To show the usefulness of our floating-point NFGs, we implemented a half precision floating-point NFG for $\ln(x)$ using the Xilinx Virtex-II FPGA (XC2V1000-4), and compared it with the dedicated circuit for $\ln(X)$ proposed in [4] in terms of throughput and delay. Table 5 shows that our NFG has 86% greater throughput.

Our NFG is a general-purpose circuit that can realize a wide range of elementary functions. Nevertheless, it achieves a higher throughput and a shorter delay time than the special-purpose circuit. Therefore, EVMDDs are useful to design fast and compact floating-point NFGs for elementary functions.

5. Conclusion and Comments

This paper has introduced a new type of integer function, called an Mp-monotone increasing function with transition points. We also derived an upper bound on the number of nodes in an EVBDD to represent the function. Experimental results showed that many monotone elementary functions can be converted into Mp-monotone increasing functions with a small number of transition points and a small p, and can be represented by EVBDDs with fewer nodes by one or two orders of magnitude than MTBDDs and BMDs. This paper has also presented a design method for floating-point NFGs based on EVMDDs. By using EVMDDs, we can automatically generate higher performance NFGs than existing NFG design techniques.

This paper showed that EVMDDs are promising to design floating-point NFGs. Our future work includes studying the proposed NFGs with high precisions in more detail.

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, 2008, and Hiroshima City University Grant for Special Academic Research (General Studies), 8108, 2008.

References

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

[2] 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.

[3] 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.

[4] J. Detrey and F. de Dinechin, "A parameterizable floating-point logarithm operator for FPGAs," *39th Asilomar Conf. on Signals, Systems and Computers,* pp. 1186–1190, 2005.

[5] J. Detrey and F. de Dinechin, "A parameterized floating-point exponential function for FPGAs," *IEEE Inter. Conf. on Field-Programmable Technology (ICFPT'05),* pp. 27–34, 2005.

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

[7] ANSI/IEEE Standard 754-2008, *IEEE Standard for Floating-Point Arithmetic*, 2008.

[8] V. K. Jain and L. Lin, "High-speed double precision computation of nonlinear functions," *Proc. of the 12th IEEE Symp. on Computer Arithmetic (ARITH'95)*, Bath, England, pp. 107–114, July 1995.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

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

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

[16] 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.

[17] S. Nagayama and T. Sasao, "Complexities of graph-based representations for elementary functions," *IEEE Trans. on Computers*, Vol. 58, No. 1, pp. 106–119, Jan. 2009.

[18] 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.

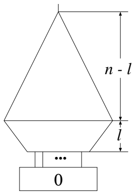


Figure A. Partition of an EVBDD.

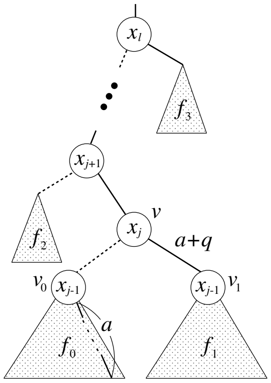


Figure B. EVBDD for a transition point.

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

[20] 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.

[21] 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.

[22] M. J. Schulte and E. E. Swartzlander, Jr., "Hardware designs for exactly rounded elementary functions," *IEEE Trans. on Comp.*, Vol. 43, No. 8, pp. 964–973, Aug. 1994.

[23] 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.

[24] 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.

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

[26] W. Wong and E. Goto, "Fast evaluation of the elementary functions in single precision," *IEEE Trans. on Comp.*, Vol. 44, No. 3, pp. 453–457, Mar. 1995.

[27] 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

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. In this case, the lower part represents l-bit Mp-monotone increasing functions and transition points, 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 Mp-monotone increasing functions and the transition points. As proven in [16, 17], the number of nodes needed to represent all the l-bit Mp-monotone increasing functions is

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

Thus, in the following, we show the maximum number of nodes needed to represent all the transition points in the lower part.

First, we assume the case where $k=1$ transition point is included. Let the transition point T be

$$T=(t_{n-1},t_{n-2},\ldots,t_{j+1},t_j,t_{j-1},\ldots,t_0)_2.$$

Then, for an integer j, we have the following:

$$\begin{aligned} T &= (t_{n-1},t_{n-2},\ldots,t_{j+1},0,1,1,\ldots,1)_2 \\ T+1 &= (t_{n-1},t_{n-2},\ldots,t_{j+1},1,0,0,\ldots,0)_2. \end{aligned}$$

Let $q=f(T+1)-f(T)$, then $q<0$ or $p<q$ holds, and it can be represented by the EVBDD shown in Fig. B. In Fig. B, a is the sum of the weights of 1-edges traversed from the node v_0 to the terminal node, and the sub-functions f_0, f_1, f_2, f_3 are Mp-monotone increasing. As shown in Fig. B, the node v and its parent nodes have only one incoming edge. If there is a node with more than one incoming edge, it contradicts the assumption of $k=1$. Since the nodes other than v and its parent nodes represent Mp-monotone increasing functions, only v and its parent nodes are required to represent a transition point. When $j=0$, the number of required nodes is maximum, and is l. That is, l nodes are needed to represent a transition point.

When k transition points are represented without shared nodes, the lower part of EVBDD has the maximum number of nodes:

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

By adding the number of nodes in the upper part and one terminal node to this, we have the theorem. The numbers of Mp-monotone increasing functions and transition points that can be represented in the lower part are $(p+1)^{2^l-1}$ [16, 17] and k, respectively. The sum of them never exceeds the number of functions which can be selected by the upper part: 2^{n-l}. Therefore, we have

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

∎