An Accurate and Compact Hyperbolic Tangent and Sigmoid Computation Based Stochastic Logic

Van-Tinh Nguyen
School of Information Science NARA Institue of Science and Technology, JAPAN

Tieu-Khanh Luong
Dept. of Electrical & Electronic Engineering, and MCCI, University College Cork, Ireland

Emanuel Popovici
Dept. of Electrical & Electronic Engineering, and MCCI, University College Cork, Ireland

Quang-Kien Trinh,
Dept. of Microelectronics and Microprocessing, Le Quy Don Technical University, VietNam

Renyuan Zhang
School of Information Science NARA Institue of Science and Technology, JAPAN

Yasuhiko Nakashima
School of Information Science NARA Institue of Science and Technology, JAPAN

***Abstract*— In this paper, a proof-of-concept implementation of hyperbolic $tanh(ax)$ and $sigmoid(2ax)$ functions for high-precision as well as compact computational hardware based on stochastic logic is presented. Nonlinear activation introducing the non-linearity in the learning process is one of the critical building blocks of artificial neural networks. Hyperbolic tangent and sigmoid are the most commonly used nonlinear activation functions in machine-learning system such as neural networks. This work demonstrates the stochastic computation of $tanh(ax)$ and $sigmoid(2ax)$ functions-based Bernstein polynomial using a bipolar format. The format conversion from bipolar to unipolar format is involved in our implementation. One achievement is that our proposed implementation is more accurate than the state-of-the-arts including FSM based method, JK-FF and general unipolar division. On average, 90% of improvement of this work in terms of mean square error (MAE) has been achieved while the hardware cost and power consumption are comparable to the previous approaches.**

Keywords—Stochastic logic, Unipolar format, Bipolar format, Bernstein polynomial, Tangent hyperbolic, Sigmoid function.

I. INTRODUCTION

Stochastic computing (SC), first introduced by Gaines [1], has recently regained significant attention due to its fault-tolerance and simple logic gates for arithmetic units [2]. Despite these advantages, SC still presents a trade-off for that hardware efficiency including long processing latency and degradation of accuracy.

The basic concept of stochastic computing is the representation of a number based on the fraction of the number of 1's in bitstreams. There are two main representations of a real number in SC: unipolar and bipolar format [3]. In unipolar format, a real number x is represented by a stochastic stream X, where

$$x = p(X = 1) = p(X) \quad (1)$$

Since x corresponds to a probability value, the unipolar representation must satisfy $0 \leq x \leq 1$. In the bipolar format.

$$x = p(X = 1) - 1 = 2p(X) - 1 \quad (2)$$

The equation (1) and (2) also suggests a format conversion from the unipolar format and bipolar format and vice versa. Additionally, a stochastic number generator (SNG) is necessary to convert a real number x to a stochastic bit stream X. A typical SNG consists of a comparator and a pseudo-random source [2]. Noticeably, this SNG structure generates one bit of stochastic sequence X in every single clock. To recover the output in binary format from a stochastic bit stream, a counter is employed which counts the number of 1's in the stream. Stochastic computational elements can be implemented based on simple logic gates. For example, multiplication which requires a high hardware cost in the binary format representation now is implemented just by an AND gate in unipolar format or XOR gate in bipolar format.

Brown and Card in [4] have first proposed stochastic implementations of hyperbolic tangent $tanh(ax)$ (where a is a positive interger) using finite-state-machines (FSMs). An improvement of this approach used 2-D FSMs was presented in [5]. One problem with these mean of implementation is accuracy degradation when a is decreased. An alternative approach was proposed in [6], in which hyperbolic tangent and sigmoid function were approximated by series expansion and JK-Flip Flops. However, this approach leads to even lower accuracy than FSM approach while requires a higher hardware cost due to the complexity of circuits. In [7], the authors proposed an approximated approach for those two functions based on a piecewise linear approximation which achieve good accuracy. However, this approach requires a stochastic to binary converter if using in the pipelined system as a look-up-table was used to store coefficients.

Our work aims to propose an implementation of hyperbolic tangent and sigmoid function-based stochastic logic to achieve high accuracy while requiring reasonable hardware cost and fit to the

stochastic end-to-end system. Bernstein polynomial [8] has been used in this work as a kernel to approximate those two functions in our proposed implementation. Format conversion from bipolar to unipolar format has been used in our work.

This paper is organized as follows. In section II, the Bernstein polynomial method used to approximate complex arithmetic function is reviewed. The proposed implementation of the hyperbolic and sigmoid function in bipolar format is presented in section III. The next section presents the simulation and experimental results of this work and compares it with the state-of-the-arts. The conclusion is given in section V.

II. Stochastic Logic Implementation Based Bernstein Polynomials

The computation of polynomial functions is typically done by using multiplications and additions. These can be effectively implemented with the stochastic arithmetic elements described in the previous section. Since the computational range of stochastic logic is limited in [0, 1], it fails for polynomials outside this range, e.g., $1.2x - 1.2x^2$ as the coefficients cannot be represented directly with stochastic bitstreams.

In [8], a method was proposed to solve this problem no matter how large the coefficients, the polynomials are synthesizable using stochastic logic. The procedure starts by transforming a power-form polynomial into a Bernstein polynomial [8] which has the form shown below.

$$B(x) = \sum_{i=0}^{n} b_i B_{i,n}(x) \tag{3}$$

Where each real number b_i is a coefficient, so-called a Bernstein coefficient, and each $B_{i,n}(x)(i = 0,1, \dots, n)$ is a Bernstein basis polynomial of the form

$$B_{i,n}(x) = \binom{n}{i} x^i (1-x)^{n-i} \tag{4}$$

For approximating nonpolynomial functions via Bernstein polynomial, a given continuous function $f(x)$ of degree n as the target, a set of coefficients $b_0, \dots, b_n$ in the interval $[0, 1]$ is considered to minimize the objective function

$$\int_0^1 \left(f(x) - \sum_{i=0}^{n} b_i B_{i,n}(x) \right)^2 dx \tag{5}$$

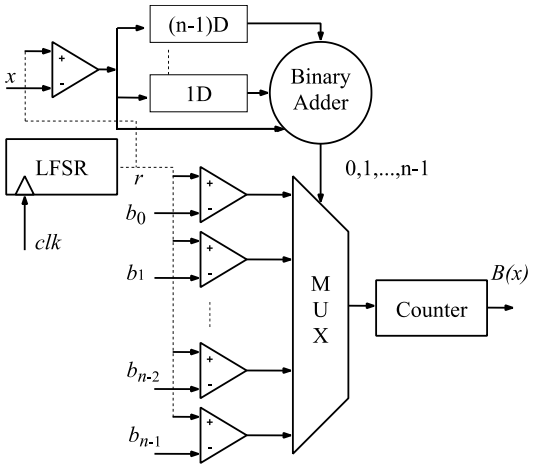


Fig. 1. Stochastic implementation of $sigmoid\,(2ax)$ via Bernstein computation.

Expanding the equation (3), an equivalent objective function can be considered as below:

$$f(b) = \frac{1}{2} b^T H b + c^T b \tag{6}$$

Where

$$b = [b_0, \dots, b_n]^T$$

$$\mathrm{c} = [-\int_0^1 f(x)\, B_{0,n}(x) dx \dots - \int_0^1 f(x)\, B_{n,n}(x) dx]^T$$

$$\mathrm{H} = \begin{bmatrix} \int_0^1 B_{0,n}(x)\, B_{0,n}(x) dx \dots \int_0^1 B_{0,n}(x)\, B_{n,n}(x) dx \\ \vdots \quad \ddots \quad \vdots \\ \int_0^1 B_{n,n}(x)\, B_{0,n}(x) dx \dots \int_0^1 B_{n,n}(x)\, B_{n,n}(x) dx \end{bmatrix}$$

The objective function in equation (6) is constrained as a quadratic programming problem where the solution can be obtained by standard techniques [8].

III. The Proposed Stochastic Implementation of Hyperbolic Tangent and Sigmoid Functions

In this section, an approach to implementing hyperbolic tangent and sigmoid function in the bipolar format are presented. The input lies in the range $[-1, 1]$. The format conversion is embedded in our approach.

The mathematical equation of $tanh(ax)\,(a > 0)$ is described as follows:

$$\tanh(ax) = \frac{1 - e^{-2ax}}{1 + e^{-2ax}} \tag{7}$$

Additionally, the sigmoid function is given by:

$$\text{sigmoid}(2ax) = \frac{1}{1+e^{-2ax}} \quad (8)$$

From the two equations above, a relation between $\tanh(ax)$ and $\text{sigmoid}(2ax)$ is illustrated as equation below:

$$\tanh(ax) = 2\frac{1}{1+e^{-2ax}} - 1 = 2sigmoid(2ax) - 1 \quad (9)$$

The bipolar format defines $x = 2P_x - 1$ in which x represents a bipolar value while P_x represent the number of ones in the corresponding bitstream. Clearly, x is in the range $[-1, 1]$ and P_x is in the range $[0, 1]$. The definition of bipolar format also suggests that a format conversion between unipolar and bipolar format is possible.

Given the input in the range $[-1, 1]$, the output of $sigmoid(2ax) \in [0, 1]$. Hence, the output of $sigmoid(2ax)$ can be represented in unipolar format. By using the same bitstream of $sigmoid(2ax)$, and applying to equation (9) which can be considered as format conversion, then the bipolar output is $tanh(ax)$. This analysis means that the same circuit can be used to implement bot functions, by considering the output bitstreams in unipolar format for $sigmoid(2ax)$ while the same output in bipolar format for $tanh(ax)$.

The implementation of $sigmoid(2ax)$ can be done by using bipolar stochastic logic elements. However, with some simple mathematical transformations below, $sigmoid(2ax)$ can be implemented by using unipolar stochastic logic elements.

$$\text{sigmoid}(2ax) = \frac{1}{1+e^{-2ax}} \quad (10)$$

$$= \frac{1}{1+e^{-2a(2P_x-1)}} \quad (11)$$

$$= \frac{1}{1+e^{-4aP_x}e^{-2a}} \quad (12)$$

$$= \frac{e^{-2a}}{e^{-2a}+e^{-4aP_x}} \quad (13)$$

From (10) to (11) x is substituted by $2P_x - 1$, where P_x is the unipolar value of the input bitstream X. Therefore, $\text{sigmoid}(2ax)$ can be implemented by unipolar stochastic logic while the input is still original bitstream as equation (13).

The approximation of equation (13) can be made by using Bernstein computation. The circuit diagram approximating $\text{sigmoid}(2ax)$ is shown in Fig. 1. In the circuit, the set of Bernstein coefficients and input x are represented in unipolar format. Binary addition is used whose output is fed to the input of the multiplexer to select which input is connected to the output. To reduce the complexity of the circuit, only one LFSR is used to generate the pseudo-random number. The uncorrelated requirement of bitstreams is solved by inserting a set of delays showed in Fig. 1.

IV. Experimental Results

This section gives the experimental results of the performance of our proposed implementation comparing to the state-of-the-arts.

$Sigmoid(2x)$ and $sigmoid(4x)$, $tanh(x)$ and $tanh(2x)$ were respectively simulated to evaluate the accuracy. Solving the quadratic programming problem in equation (6) for $sigmoid(2ax)$ applied equation (13), a set of Bernstein coefficients of 5^{th} order Bernstein polynomial is obtained as being shown in Table I.

TABLE I. The Bernstein Coefficients $b_i (0 \le i \le 5)$ synthesizing $Sigmoid(2x)$ and $Sigmoid(4x)$

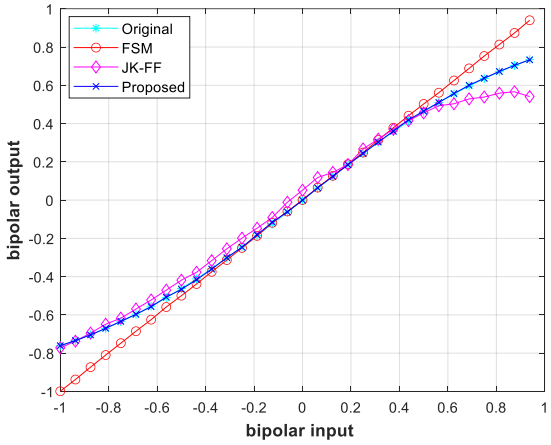
Coefficients	b_0	b_1	b_2	b_3	b_4	b_5
$sigmoid(2x)$	0.12	0.2	0.34	0.66	0.8	0.87
$sigmoid(4x)$	0.03	0.02	0	1	0.98	0.96

The length pf stochastic bitstream is 1024, which means that 10-bit LFSR is used for SNG. In our simulation, the inputs of target functions are given by 0:0.03:1. The output results are collected through Monte Carlo experiments. The accuracy is evaluated via Mean Absolute Error (MAE). Fig. 2 shows the simulation results of approximated functions in different approaches and target functions.

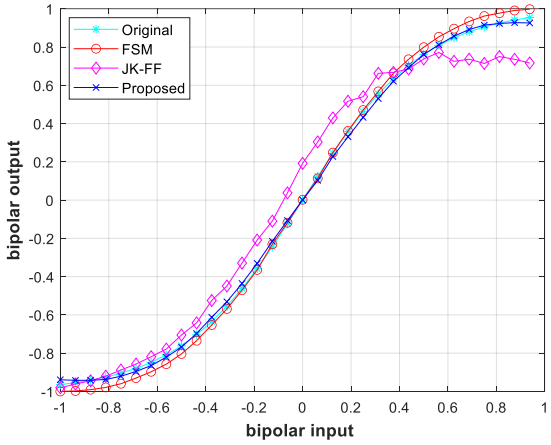
Synthesize results of the proposed function and state-of-the-arts for $Sigmoid(2x)$ and $sigmoid(4x)$, $tanh(x)$ and $tanh(2x)$ are considered. All architectures are implemented using 180nm CMOS technology node and synthesized Synopsys Design Compiler. A summarized table of power consumption, area, delay and MAE is shown in Table II. It is noted that the same circuit can be used to implement both $sigmoid(2ax)$ and $tanh(ax)$, then the same hardware cost, power consumption and MAE are achieved. In terms of accuracy, our proposed implementation is roughly 60 times and 20 times more accurate than FSM-based method and JK-FF based method, respectively, for $sigmoid(2x)$ and $tanh(x)$. Additionally, 10 and 16 times of

TABLE II. MAE EVALUATION AND HARDWARE SYNTHESIZE FOR HYPERBOLIC TANGENT AND SIMOIG FUCNTION BASED STOCHASTIC LOGIC

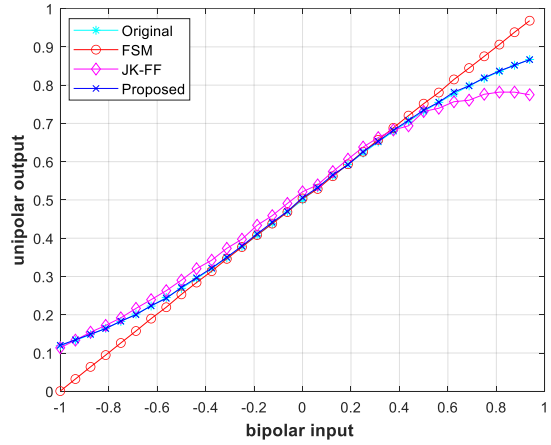
Function	$\tanh(x)$ and $\text{sigmoid}(2x)$				$\tanh(2x)$ and $\text{sigmoid}(4x)$			
Method	Proposed		FSM [4]	JK-FF [6]	Proposed		FSM [4]	JK-FF [6]
	n=3	n=5	2 states	-	n=3	n=5	2 states	-
Area ($(\mu m)^2$)	1554	1777	1345	10121	1777	2106	1551	10476
Latency (ns)	2.25	2.33	2.38	3.42	2.33	3.3	3.07	3.07
Power (mW)	0.07	0.08	0.06	0.4	0.11	0.11	0.08	0.08
MAE	0.003	0.001	0.06	0.02	0.007	0.003	0.03	0.05



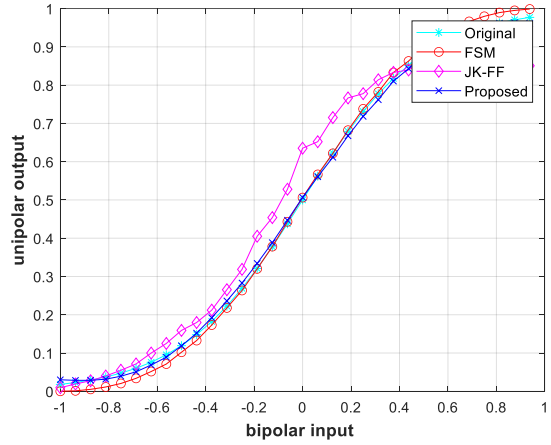
(a)



(b)



(c)



(d)

Fig. 2. Simulation results compared different approaches with target functions: (a) $\tanh(x)$, (b) $\tanh(2x)$, (c) $\text{sigmoid}(2x)$, (d) $\text{sigmoid}(4x)$

improvement of accuracy, on average, are achieved by our proposed method in comparison to FSM and JK-FF based method, respectively, $sigmoid(4x)$ and $tanh(2x)$.

Our proposed implementations are 80% and 85% less area and power consumption that JK-FF approach. The proposed circuit employing 3^{th} and 5^{th} order Bernstein polynomial reduced roughly 20% of critical path delay in comparison with FSM and JK-FF-based implementation.

V. CONCLUSIONS

In this paper, an approached computation of $sigmoid(2ax)$ and $tanh(ax)$ in a bipolar format based Bernstein polynomial has been proposed. The results showed that an improvement of accuracy had been achieved while maintained a comparable hardware cost in comparing to state-of-the-art.

ACKNOWLEDGEMENT

This work supported by JST, PRESTO Grant Number JPMJPR18M7, Japan, and the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2018.310. The authors would like to thank the VLSI Design and Education Center (VDEC) of the University of Tokyo in collaboration with Rohm Corporation, Toppan Printing Corporation, Synopsys, Inc., Cadence Design Systems, Inc., and Mentor Graphics, Inc.comparable hardware cost in comparing to state-of-the-art.

REFERENCES

[1] B. R. Gaines, "Stochastic computing," in *Proceedings of AFIPS spring joint computer conference*, pp. 149–156, ACM, 1967.

[2] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Transactions on Embedded computing systems (TECS)*, vol. 12, no. 2s, p. 92, 2013

[3] K. K. Parhi, "Analysis of stochastic logic circuits in unipolar, bipolar and hybrid formats," *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, Baltimore, MD, 2017, pp. 1-4.

[4] B. D. Brown and H. C. Card, "*Stochastic neural computation. I. computational elements*," IEEE Transactions on Computers, vol. 50, no. 9, pp. 891–905, 2001.

[5] P. Li, D. J. Lilja, K. Bazargan and M. Riedel, "The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic," in *IEEE/ACM (ICCAD)*, San Jose, CA, USA, pp. 480-487, Dec. 2012.

[6] Y. Liu and K. K. Parhi, "Computing hyperbolic tangent and sigmoid functions using stochastic logic," 2016 50th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 2016, pp. 1580-1585.

[7] K. T. Luong, V. Nguyen, A. Nguyen and E. Popovici, "Efficient Architectures and Implementation of Arithmetic Functions Approximation Based Stochastic Computing," *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, New York, 2019, pp. 281-287.

[8] W. Qian, X. Li, M. D. Riedel, K. Bazargan and D. J. Lilja, "An Architecture for Fault-Tolerant Computation with Stochastic Logic," in IEEE Transactions on Computers, vol. 60, no. 1, pp. 93-105, Jan. 2011