Computing Hyperbolic Tangent and Sigmoid Functions using Stochastic Logic

Yin Liu and Keshab K. Parhi
University of Minnesota
Department of Electrical and Computer Engineering
Minneapolis, MN, USA

***Abstract*—This paper addresses implementations of tangent hyperbolic and sigmoid functions using stochastic logic. Stochastic computing requires simple logic gates and is inherently fault-tolerant. Thus, these structures are well suited for nanoscale CMOS technologies. Tangent hyperbolic and sigmoid functions are widely used in machine learning systems such as neural networks. This paper makes two major contributions. First, two approaches are proposed to implementing tangent hyperbolic and sigmoid functions in unipolar stochastic logic. The first approach is based on a JK flip-flop. In the second approach, the proposed designs are based on a general unipolar division. Second, we present two approaches to computing tangent hyperbolic and sigmoid functions in bipolar stochastic logic. The first approach involves format conversion from bipolar format to unipolar format. The second approach uses a general bipolar stochastic divider. Simulation and synthesis results are presented for proposed designs.**

***Keywords*—Stochastic logic, Unipolar format, Bipolar format, Stochastic division, Tangent hyperbolic, Sigmoid function.**

I. Introduction

Stochastic computing (SC), first proposed in 1960s [1] [2], has recently regained significant attention due to its fault-tolerance and extremely low-cost of arithmetic units [3]. Despite these advantages, stochastic circuits suffer from long processing latency and degradation of accuracy.

The stochastic representation is based on the fraction of 1's in bit streams. Stochastic logic was proposed by Gaines in two formats, unipolar and bipolar [2]. In the unipolar format, a real number x is represented by a stochastic bit stream X, where

$$x = p(X = 1) = p(X).$$

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

$$x = 2p(X = 1) - 1 = 2p(X) - 1,$$

where $-1 \leqslant x \leqslant 1$.

To convert a digital value x to a stochastic bit stream X, a stochastic number generator (SNG) is necessary. Fig. 1 shows an SNG circuit consisting of a comparator and a linear-feedback-shift-register (LFSR) which corresponds to a pseudo-random source [3]. This SNG generates one bit of a stochastic sequence (X) every clock cycle.

This research was supported by the National Science Foundation under grant number CCF-1319107.

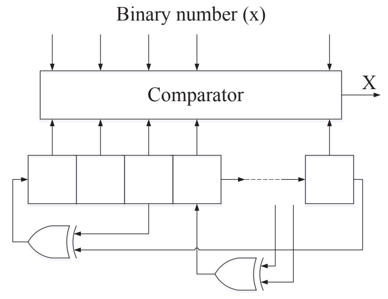


Fig. 1: The circuit diagram of a basic stochastic number generator (SNG).

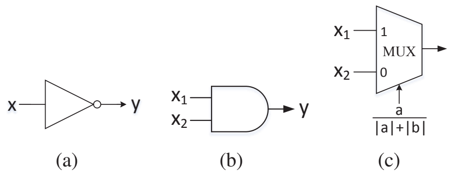


Fig. 2: Fundamental stochastic computational elements. (a) $y = 1 - x$ in unipolar format format. (b) Unsigned multiplication in unipolar format: $y = x_1 \cdot x_2$. (c) Scaled addition in unipolar/bipolar format: $y = (1 - s) \cdot x_1 + s \cdot x_2$.

Stochastic computational elements can be implemented based on simple combinational logic [1]. Fig. 2 illustrates fundamental stochastic combinational logic blocks. The NOT gate is used to implement $1 - x$ in unipolar format and $-x$ in bipolar format. The AND gate implements the multiplication in unipolar format. The scaled addition for both unipolar and bipolar formats is implemented using a multiplexer (MUX). Since stochastic computing requires that numbers in unipolar or bipolar format satisfy their range restrictions ([0,1] for unipolar and [-1,1] for bipolar), the result of addition is implicitly scaled by s to prevent overflow. The details of stochastic computational elements can be found in [1] and [4].

One advantage of stochastic computing is that complex computations on stochastic bit streams can be realized using extremely low-cost designs in terms of hardware complexity. Brown and Card [4] have proposed stochastic implementations of hyperbolic tangent and exponential functions using finite state machines (FSMs). It has also been illustrated in [5] that complex functions can be approximated in stochastic logic by using Bernstein polynomials. It was shown in [6] that instead of using FSMs or Bernstein polynomials, the unipolar tangent hyperbolic and sigmoid functions can be implemented by using Maclaurin series expansion and stochastic division. This paper expands the results of [6] to unipolar and bipolar cases.

This paper makes two main contributions. First, it is shown that stochastic unipolar tangent hyperbolic and sigmoid function can be implemented using two approaches. The first approach is based on a JK flip-flop. In the second approach, the proposed designs are based on a general unipolar division. Second, we present two approaches to computing tangent hyperbolic and sigmoid functions in bipolar stochastic logic. The first approach involves format conversion from bipolar format to unipolar format. The second approach uses a general bipolar stochastic divider.

This paper is organized as follows. In Section II, implementations of tangent hyperbolic and sigmoid functions in unipolar stochastic logic are proposed. Section III presents implementations of tangent hyperbolic and sigmoid functions in bipolar stochastic logic. Simulation and synthesis results of proposed designs are described in Section IV. Finally, some conclusions are given in Section V.

II. Implementations of Tangent Hyperbolic and Sigmoid Functions in Unipolar Stochastic Logic

In this section, two approaches to implementing tangent hyperbolic function in unipolar stochastic logic are presented. The inputs of target functions are assumed to be in the range of $[0, 1]$.

A. Review of Implementations of Exponential functions and division in Unipolar Stochastic Logic

Consider the following tangent hyperbolic function $\tanh ax$ $(a > 0)$:

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

The exponential function is a fundamental building block for tangent hyperbolic function. Therefore, before stepping into the unipolar implementation of $\tanh ax$, we first review implementations of e^{-ax} in stochastic *unipolar* logic, where $a > 0$.

For $0 < a \leq 1$, e^{-ax} can be implemented using Maclaurin expansion and Horner's rule [6]. For instance, the 5th-order Maclaurin polynomial of e^{-ax} transformed for stochastic implementation is given by:

$$\begin{aligned} e^{-ax} &\approx 1 - ax + \frac{a^2x^2}{2!} - \frac{a^3x^3}{3!} + \frac{a^4x^4}{4!} - \frac{a^5x^5}{5!} \\ &= 1 - ax(1 - \frac{ax}{2}(1 - \frac{ax}{3}(1 - \frac{ax}{4}(1 - \frac{ax}{5})))). \quad (2) \end{aligned}$$

The e^{-ax} is described by a 5^{th}-order truncated Maclaurin polynomial and then is transformed using Horner's rule. Given $0 < a \leq 1$, all coefficients in (2) are in the range of $[0, 1]$, which are feasible in stochastic unipolar logic. Fig. 3 shows the corresponding circuit diagram for stochastic implementation of e^{-ax} in unipolar format using equation (2) [6].

Notice that the design can not be directly used for e^{-ax} with $a > 1$ since coefficients may be greater than one, which violates the constraint of unipolar format of stochastic logic.

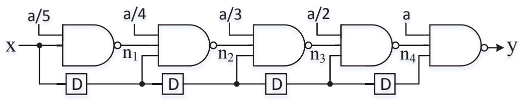


Fig. 3: The circuit diagram of stochastic implementation of e^{-ax} using the 5^{th}-order Maclaurin polynomial ().

However, for large a, e^{-ax} can be implemented based on e^{-bx} with small b. Consider the stochastic implementation of e^{-2x}, which can be written as follows:

$$e^{-2x} = e^{-x} \cdot e^{-x}.$$

Then the e^{-2x} can be implemented as shown in Fig. 4. The e^{-x} in Fig. 4 is implemented using the circuit shown in Fig. 3. The one-bit delay element is used for decorrelation. A complete decorrelation of all paths would require 5 delays in Fig. 4. However, the error due to correlation using only one AND gate is small in this example.

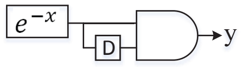


Fig. 4: The circuit diagram of stochastic implementation of e^{-2x}.

For any arbitrary a $(a > 1)$, e^{-ax} can be described as follows:

$$e^{-ax} = e^{(-\frac{a}{n}x)n} = e^{(-bx)n}, \ b = \frac{a}{n}.$$

where $0 < b \leq 1$ and n is an integer. Since $b \leq 1$, e^{-bx} can be easily implemented using Horner's rule as described in Section III. Then e^{-ax} can be implemented as shown in Fig. 5 by using e^{-bx} and $n - 1$ cascaded AND gates. Notice that the method of decomposing e^{-ax} for $a > 1$ is not unique and more details can be found in [6].

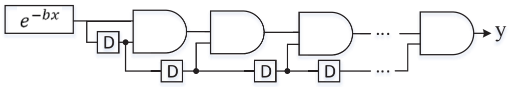
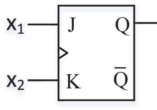
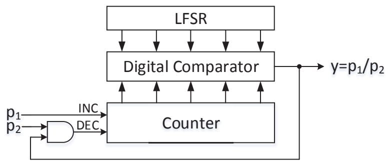


Fig. 5: The circuit diagram of stochastic implementation of e^{-ax} $(a > 1)$ by using e^{-bx} $(b \leq 1)$ and $n - 1$ cascaded AND gates.

In equation (1), besides the exponential function, a remaining critical operation is division. In stochastic logic, unipolar division can be implemented using two methods [2]. The first approach is based on a JK flip-flop as shown in Fig. 6(a). In this case, the functionality performed by the JK flip-flop is $y = x_1/(x_1 + x_2)$ rather than a general division. The second approach is based on a binary counter, which may be incremented or decremented by unit count. As shown in Fig. 6(b), this design performs a general division, where $y = p_1/p_2$ for $p_1 < p_2$. More details of stochastic unipolar division using these two approaches can be found in [2].



(a)



(b)

Fig. 6: (a) The function $y = \frac{x_1}{x_1+x_2}$ implemented using a JK flip-flop. (b) Stochastic divider $y = p_1/p_2$ in unipolar format where $p_1 < p_2$.

B. Tangent Hyperbolic and Sigmoid Functions Design using JK Flip-flop

The expression of $\tanh ax$ in (1) can be further transformed as follows:

$$\tanh ax = \frac{1 - e^{-2ax}}{1 + e^{-2ax}} = \frac{\frac{1-e^{-2ax}}{2}}{\frac{1-e^{-2ax}}{2} + e^{-2ax}}. \tag{3}$$

Fig. 7 shows the implementation of $\tanh ax$ based on (3) as presented in [6]. Two inputs of the JK flip-flop are $1 - e^{-2ax}$ and e^{-2ax}, respectively. The one-bit delay element is used for decorrelation. The e^{-2ax} is implemented using the method presented in Section II. A.

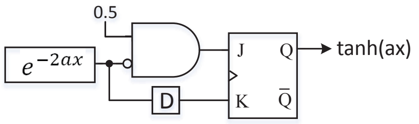


Fig. 7: The circuit diagram of stochastic implementation of $\tanh ax$ $(a > 0)$ using e^{-2ax} and a JK flip-flop.

An alternative design of $\tanh ax$ based on JK flip-flop can be implemented using the following expression:

$$\tanh ax = \frac{1}{1 + e^{-2ax}} \cdot (1 - e^{-2ax}). \tag{4}$$

The corresponding circuit is shown in Fig. 8. Two inputs of the JK flip-flop are given by 1 and e^{-2ax}. Notice that sigmoid$(2ax)$ is computed at the output of the JK flip-flop, where sigmoid$(2ax) = 1/(1 + e^{-2ax})$.

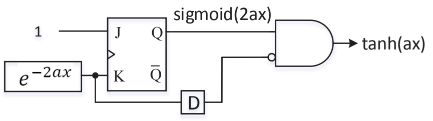


Fig. 8: An alternative design of $\tanh ax$ based on JK flip-flop in stochastic logic, with sigmoid$(2ax)$ computed at an internal node.

C. Tangent Hyperbolic and Sigmoid Functions Design using a General Unipolar Division

Stochastic $\tanh ax$ in unipolar format can also be implemented using the general division shown in Fig. 6(b). Consider the $\tanh ax$ described in (1). The circuit diagram is illustrated in Fig. 9. The stochastic divider represents the general division shown in Fig. 6(b). The numerator of the division is computed by an AND gate, which performs $0.5(1 - e^{-2ax})$. The denominator is computed by a multiplexer, where the output of the MUX is $(1 + e^{-2ax})/2$. The final output is given by the stochastic divider.

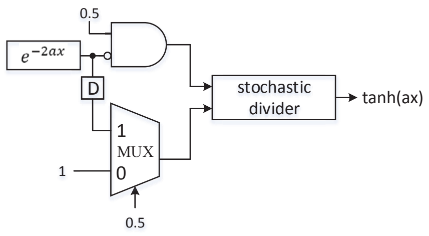


Fig. 9: An alternative design of $\tanh ax$ $(a > 0)$ using the general unipolar division.

Consider equation (4) for $\tanh ax$. Fig. 10 shows the circuit diagram of unipolar $\tanh ax$ using this equation and the general unipolar division. In this design, the MUX computes $(1 + e^{-2ax})/2$, which is the denominator of the stochastic division. The numerator is a fixed value 0.5. Therefore, the stochastic divider computes $1/(1 + e^{-2ax})$, which is sigmoid$(2ax)$. By performing a multiplication for the output of this divider with $(1 - e^{-ax})$, the final output $\tanh ax$ is generated by an AND gate.

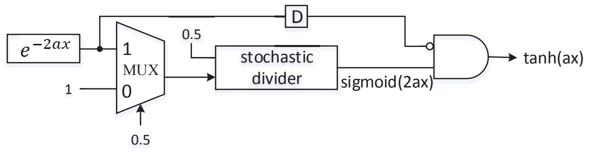


Fig. 10: The circuit diagram of stochastic implementation of $\tanh ax$ $(a > 0)$ using the general unipolar division, with sigmoid$(2ax)$ computed at an internal node.

III. Implementations of Tangent Hyperbolic and Sigmoid Functions in Bipolar Stochastic Logic

In this section, two approaches to implementing tangent hyperbolic and sigmoid functions in bipolar stochastic logic are presented. The inputs lie in the range $[-1, 1]$. The first approach is based on the format conversion from bipolar format to unipolar format. The second approach is based on a unipolar division.

A. Tangent Hyperbolic and Sigmoid Functions Design Based on Format Conversion

Consider $tanh(ax)$ $(a > 0)$ described as follows:

$$y = \tanh ax = \frac{1 - e^{-2ax}}{1 + e^{-2ax}}, \tag{5}$$

and the function sigmoid$(2ax)$ $(a > 0)$ is given by:

$$z = \text{sigmoid}(2ax) = \frac{1}{1 + e^{-2ax}}. \tag{6}$$

The relation between y and z is described as follows:

$$y = \frac{1 - e^{-2ax}}{1 + e^{-2ax}} = 2 \cdot \frac{1}{1 + e^{-2ax}} - 1 = 2z - 1. \tag{7}$$

Recall the definition of bipolar format $x = 2P_X - 1$, where x represents a bipolar value while P_X represents the number of ones in the corresponding bit-stream. Notice that P_X is also the unipolar value of this bit-stream. The ranges of x and P_X are given by $[-1, 1]$ and $[0, 1]$, respectively. We can observe that the definition of bipolar format is in the same form of equation (7). Given the input in the range $[-1, 1]$, $\text{sigmoid}(2ax) \in [0, 1]$. Therefore, we can represent $\text{sigmoid}(2ax)$ in unipolar format, i.e., z is a unipolar value and $z = P_Z$, where Z is the corresponding bit-stream. By using the same bit stream Z and considering it encoded using bipolar format, then the bipolar value is $\tanh ax$, since we have $y = 2z - 1 = 2P_Z - 1$ from equation (7). This means that we can use the same circuit to compute both functions, by considering the output sequence in ***unipolar*** format for $\text{sigmoid}(2ax)$ while considering the output sequence in ***bipolar*** format for $\tanh ax$.

Consider the implementation of $z = \text{sigmoid}(2ax)$ using bipolar input and unipolar output, i.e., x is a bipolar value while z is a unipolar value. The function can be transformed as follows:

$$\begin{aligned}
z =& \text{sigmoid}(2ax) = \frac{1}{1 + e^{-2ax}} && (8)\\
=& \frac{1}{1 + e^{-2a(2P_X - 1)}} && (9)\\
=& \frac{1}{1 + e^{-4aP_X} \cdot e^{2a}} && (10)\\
=& \frac{e^{-2a}}{e^{-2a} + e^{-4aP_X}}. && (11)
\end{aligned}$$

From (8) to (9), x is substituted by $2P_X - 1$, where P_X is the unipolar value of the input bit stream X. Since z is also in unipolar format, (11) can be implemented in unipolar logic, where the input is still the original bit stream X. The circuit diagram is shown in Fig. 11. In this circuit, the coefficient e^{-2a} is in the range of $[0, 1]$ given $a > 0$ and is represented in unipolar format. The e^{-4aP_X} is implemented using the approach presented in Section II. A. Two inputs of the JK flip-flop are given by e^{-2ax} and e^{-4aP_X}.

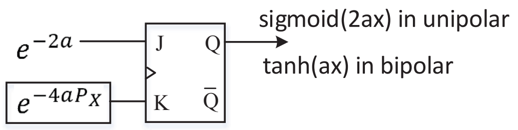


Fig. 11: The circuit diagram of stochastic implementation of $\tanh ax$ and $\text{sigmoid}(2ax)$ $(a > 0)$ with bipolar input using format conversion.

Due to the equivalence of (8) and (11), the circuit performs the computation in (8) by considering the input bit stream in bipolar format. Therefore, $\text{sigmoid}(2ax)$ with bipolar input and unipolar output is implemented using the design shown in Fig. 11. As explained above, this circuit also performs $\tanh ax$ with bipolar input and bipolar output.

For more details of stochastic circuits using format conversion, the reader is referred to [6] [7].

B. Tangent Hyperbolic Function Design using Bipolar Division

Besides the implementation involving format conversion, tangent hyperbolic function can also be implemented using bipolar division. A typical bipolar division is shown in Fig. 12. In this circuit, a LFSR is used as a pseudo-random number

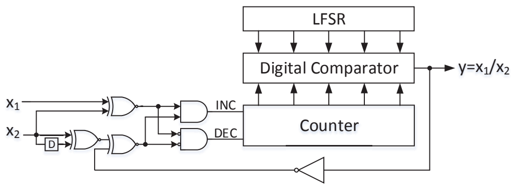


Fig. 12: The circuit diagram of stochastic bipolar division [2].

generator. A comparator is used to compare the value of the LFSR and a binary counter, which may be incremented or decremented by unit count. The bit stream generated by the comparator represents the division result in bipolar format. Notice that inputs x_1 and x_2 are also represented in bipolar format. More details of this bipolar division can be found in [2].

Consider the $\tanh ax$ $(a > 0)$ described as follows:

$$\tanh ax = \frac{e^{ax} - e^{-ax}}{e^{ax} + e^{-ax}}. \tag{12}$$

The Maclaurin expansion of e^{-ax} is given by;

$$\begin{aligned}
e^{-ax} \approx& 1 - ax + \frac{a^2x^2}{2!} - \frac{a^3x^3}{3!} + \frac{a^4x^4}{4!} - \frac{a^5x^5}{5!} \cdots\\
=& (1 + \frac{a^2x^2}{2!} + \frac{a^4x^4}{4!} + \cdots)\\
&- (ax + \frac{a^3x^3}{3!} + \frac{a^5x^5}{5!} + \cdots)\\
=& P(x) - Q(x), && (13)
\end{aligned}$$

where all even-order order terms are grouped as $P(x)$ while all odd-order terms are grouped as $Q(x)$. Notice that all coefficients in $P(x)$ and $Q(x)$ are positive. Accordingly, the Maclaurin expansion of e^{ax} can be described as follows:

$$\begin{aligned}
e^{ax} \approx& 1 + ax + \frac{a^2x^2}{2!} + \frac{a^3x^3}{3!} + \frac{a^4x^4}{4!} + \frac{a^5x^5}{5!} \cdots\\
=& (1 + \frac{a^2x^2}{2!} + \frac{a^4x^4}{4!} + \cdots)\\
&+ (ax + \frac{a^3x^3}{3!} + \frac{a^5x^5}{5!} + \cdots)\\
=& P(x) + Q(x). && (14)
\end{aligned}$$

Substituting e^{ax} and e^{-ax} in (12) using (13) and (14), $\tanh ax$ can be described as follows:

$$\tanh ax = \frac{(P(x) + Q(x)) - (P(x) - Q(x))}{(P(x) + Q(x)) + (P(x) - Q(x))} = \frac{Q(x)}{P(x)} \tag{15}$$

In this design, all bit streams are encoded in bipolar format. $P(x)$ and $Q(x)$ can be easily implemented using multiple levels of multiplexers [8]. Then $P(x)$ and $Q(x)$ are used as inputs of stochastic bipolar divider. The circuit is shown in Fig. 13, where the bipolar divider is implemented as shown in Fig. 12.

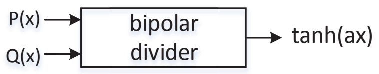


Fig. 13: The circuit diagram of stochastic implementation of $\tanh ax$ $(a > 0)$ using the general bipolar division.

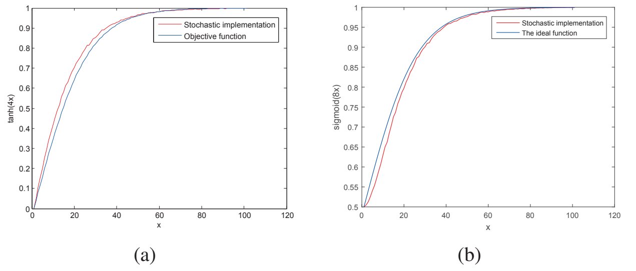


Fig. 14: Simulation results of unipolar (a) $\tanh 4x$ and (b) sigmoid$(8x)$ implemented using the proposed design shown in Fig. 8, with 5^{th}-order truncated Maclaurin polynomial for exponential function.

IV. Experimental results

In this section, we present the experimental results of performance test and synthesis results for unipolar and bipolar implementations of tangent hyperbolic and sigmoid functions.

A. Tangent Hyperbolic and Sigmoid Functions in Unipolar Logic

Simulations were performed to test accuracies of $\tanh 4x$ and sigmoid$(8x)$ using the unipolar design shown in Fig. 8, where a JK flip-flop is used to perform the division. The length of stochastic bit streams is 1024. A 10-bit LFSR is used in SNG. In our simulations, the inputs of target functions are given by 0:0.01:1. The output results are obtained using Monte Carlo experiments for different inputs. 1000 Monte Carlo runs were performed for each input. Fig. 14 shows simulations results and target functions, where the 5^{th}-order Maclaurin polynomial of the exponential function is used in the proposed implementations.

Table I presents the output mean absolute error (MAE) of the proposed design for $\tanh 4x$ using JK flip-flop in stochastic unipolar logic as shown in Fig. 8.

Synthesis results are considered for stochastic unipolar implementations of $\tanh 4x$ using JK flip-flop. The architectures are implemented using 65nm libraries and synthesized using Synopsys Design Compiler. The length of the stochastic sequence is 1024 and all required SNGs including 10-bit LFSRs as random number generators are considered in our synthesis. The operating conditions for each implementation are specified by a supply voltage of 1.05 V and a temperature of 25 degree Celsius. Table II describes synthesis results of the proposed design presented in Fig. 8. The hardware complexity results are given in terms of equivalent 2-input NAND gates. All SNGs are included in synthesis. The delay of critical path of the proposed design is also presented.

TABLE I: The output mean absolute error (MAE) of the proposed design for $\tanh 4x$ using JK flip-flop in stochastic unipolar logic as shown in Fig. 8, where exponential functions are implemented using Maclaurin polynomials with different orders.

	Order	4	5	6
$\tanh 4x$	Error	0.0199	0.0192	0.0191

TABLE II: The hardware complexity and critical path delay (ns) of the proposed design for $\tanh 4x$ using JK flip-flop in stochastic unipolar logic as shown in Fig. 8, where exponential functions are implemented using Maclaurin polynomials with different orders.

	Order	4	5	6
$\tanh 4x$	Area	583.4	649.5	728.5
	Delay (ns)	3.39	3.63	4.07

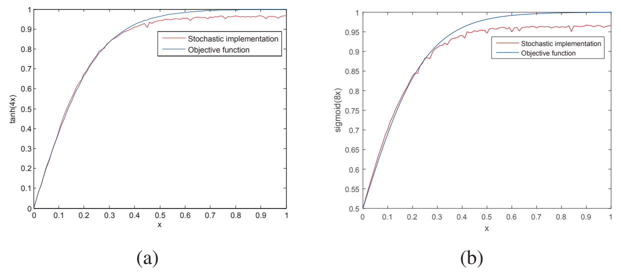


Fig. 15: Simulation results of unipolar (a) $\tanh 4x$ and (b) sigmoid$(8x)$ implemented using the proposed design shown in Fig. 10, with 5^{th}-order truncated Maclaurin polynomial for exponential function.

We performed simulations for $\tanh 4x$ and sigmoid$(8x)$ using the unipolar design shown in Fig. 10, where a general unipolar divider is used. Fig. 15 shows simulations results and target functions, where the 5^{th}-order Maclaurin polynomial of the exponential function is used in the proposed implementations.

Table III presents the output mean absolute error (MAE) of the proposed design for $\tanh 4x$ using a general unipolar divider as shown in Fig. 10, where exponential functions are implemented using Maclaurin polynomials with different orders. Synthesis results are considered for stochastic unipolar implementations of $\tanh 4x$ using a general unipolar divider. Table IV describes synthesis results of the proposed design presented in Fig. 10. The hardware complexity results are given in terms of equivalent 2-input NAND gates. All SNGs are included in synthesis. The delay of critical path of the proposed design is also presented.

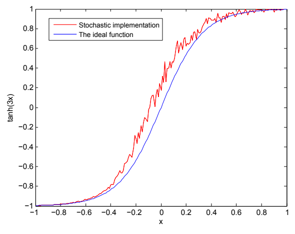
Comparing Table II and Table IV, we observe that hardware complexity of the stochastic implementation using JK flip-flop is less than the design based on a general unipolar divider.

TABLE III: The output mean absolute error (MAE) of the proposed design for $\tanh 4x$ using a general unipolar divider as shown in Fig. 10, where exponential functions are implemented using Maclaurin polynomials with different orders.

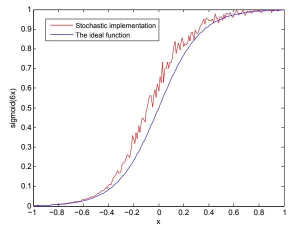
	Order	4	5	6
$\tanh 4x$	Error	0.0178	0.0175	0.0140

TABLE IV: The hardware complexity and critical path delay (ns) of the proposed design for $\tanh 4x$ using a general unipolar divider as shown in Fig. 10, where exponential functions are implemented using Maclaurin polynomials with different orders.

	Order	4	5	6
$\tanh 4x$	Area	1033.2	1100.8	1181.4
	Delay (ns)	7.50	7.38	7.43



(a)



(b)

Fig. 16: Simulation results of (a) $\tanh 3x$ and (b) sigmoid$(6x)$ implemented using the proposed design shown in Fig. 11 based on format conversion, with 5^{th}-order truncated Maclaurin polynomial for exponential function.

B. Tangent Hyperbolic and Sigmoid Functions in Bipolar Logic

Simulations were performed to test accuracies of $\tanh 3x$ and sigmoid$(6x)$ using the design shown in Fig. 11, which involves format conversion. The length of stochastic bit streams is 1024. A 10-bit LFSR is used in SNG. In our simulations, the inputs of target functions are given by 0:0.01:1. Fig. 16 shows simulations results and target functions, where the 5^{th}-order Maclaurin polynomial of the exponential function is used in the proposed implementation. The output MAE of $\tanh 3x$ is given by 0.0683 and the output MAE of sigmoid$(6x)$ is given by 0.0341.

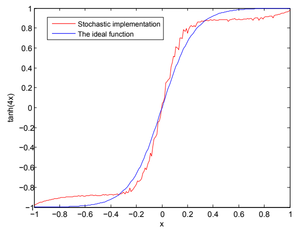
Synthesis results are considered for stochastic implementations of $\tanh 3x$ based on format conversion using JK flip-flop. Table V describes synthesis results of the proposed design presented in Fig. 11.

We performed simulations for $\tanh 4x$ and $\tanh 8x$ using the bipolar design shown in Fig. 13, where a general bipolar divider is used. Fig. 17 shows simulations results and target functions, where the 5^{th}-order Maclaurin polynomial of the exponential function is used in the proposed implementations. The output MAE of $\tanh 4x$ is given by 0.0746 and the output MAE of $\tanh 8x$ is given by 0.0879.

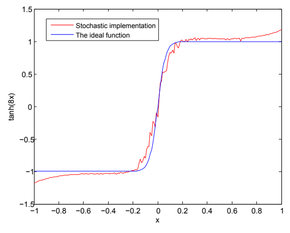
Synthesis results are considered for stochastic unipolar implementations of $\tanh 4x$ using a general bipolar divider. Table VI describes synthesis results of the proposed design presented in Fig. 13.

TABLE V: The hardware complexity and critical path delay (ns) of the proposed design for $\tanh 3x$ based on format conversion using JK flip-flop as shown in Fig. 11.

	Order	4	5	6
$\tanh 3x$	Area	576.2	621.9	720.4
	Delay (ns)	3.31	3.53	4.02



(a)



(b)

Fig. 17: Simulation results of bipolar (a) $\tanh 4x$ and (b) $\tanh 8x$ implemented using the proposed design shown in Fig. 13.

TABLE VI: The hardware complexity and critical path delay (ns) of the proposed design for $\tanh 4x$ using a general bipolar divider as shown in Fig. 13, where exponential functions are implemented using Maclaurin polynomials with different orders.

	Order	4	5	6
$\tanh 4x$	Area	1081.4	1174.5	1242.6
	Delay (ns)	7.42	7.39	7.48

V. Conclusion

Stochastic logic based tangent hyperbolic and sigmoid functions in unipolar and bipolar format have been presented in this paper. Two approaches are described for implementations in unipolar format, which are based on a JK flip-flop and unipolar stochastic division, respectively. Two approaches are proposed for implementations in bipolar format, which are based on format conversion and bipolar stochastic division, respectively. However, the proposed approach for $\tanh ax$ and sigmoid(ax) requires more hardware and leads to more error than FSM based implementations [4].

References

[1] B. R. Gaines, "Stochastic computing," in *Proceedings of AFIPS spring joint computer conference*, pp. 149–156, ACM, 1967.
[2] B. Gaines, "Stochastic computing systems," in *Advances in information systems science*, pp. 37–172, Springer, 1969.
[3] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Transactions on Embedded computing systems (TECS)*, vol. 12, no. 2s, p. 92, 2013.
[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] W. Qian and M. D. Riedel, "The synthesis of robust polynomial arithmetic with stochastic logic," in *45th ACM/IEEE Design Automation Conference*, pp. 648–653, 2008.
[6] K. K. Parhi and Y. Liu, "Computing arithmetic functions using stochastic logic by series expansion," *IEEE Transactions on Emerging Topics in Computing*, DOI: 10.1109/TETC.2016.2618750.
[7] Y. Liu and K. K. Parhi, "Computing RBF kernel for SVM classification using stochastic logic," in *Proceedings of 2016 IEEE Workshop on Signal Processing Systems*, Dallas, Oct. 2016.
[8] Y.-N. Chang and K. K. Parhi, "Architectures for digital filters using stochastic computing," in *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2697–2701, 2013.