International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Polynomial Computation Using Multiplexer-Based Stochastic Logic in VLSI

M. Valarmathi[1], G. Rajeev[2], K. Pavan Kumar[3], K. Mohith Reddy[4]

[1]Assistant Professor, Department of ECE, SRM University Chennai
[2,3,4]Department of ECE, SRM University Chennai

Abstract

The computation of polynomials plays a vital role in various Very Large Scale Integration (VLSI) applications including digital signal processing systems, neural networks, control systems, and scientific computation. Most of the complicated mathematical operations used in these applications, including exponential and hyperbolic functions, are generally approximated using polynomial expressions. However, the implementation of these polynomial operations using binary arithmetic systems generally requires a lot of hardware resources, increased power consumption, and longer computation time. Hence, with the growing need in VLSI systems for smaller hardware implementations with low power consumption and increased computation speeds, various computational techniques have gained significant research interest.

A promising alternative is stochastic computing[1][2], in which numerical data is represented in a probabilistic rather than a definite binary form. In stochastic computing, the binary representation of a number is based on the probability of a logic '1' occurring in a random bit stream. For example, if there are 70 logic '1's in a random bit stream of 100 bits, the represented number is 0.7. Such a binary representation of a number allows complex arithmetic operations such as multiplication and polynomial evaluation to be implemented using simple logic gates without the need for complex arithmetic logic. Stochastic logic offers significant advantages in hardware, ruggedness, and power consumption.

In stochastic computing, arithmetic operations that otherwise require complex combinational logic can be performed with minimal hardware. For example, multiplication can be performed with an AND gate only, and addition and weighted operations can be performed with multiplexers. This reduced hardware requirement results in low power dissipation, making stochastic computing very attractive for low-power VLSI circuits. Moreover, stochastic computing has inherent robustness against noise and bit error, making it an even more suitable candidate for current and future integrated circuits.

In the proposed work, a multiplexer-based stochastic architecture is proposed for the efficient computation of polynomials in VLSI systems. The proposed architecture is aimed at efficiently implementing polynomial approximations of important mathematical functions using stochastic logic circuits. The proposed architecture is used in approximating various mathematical functions such as (e^x), (e^{-x}), (sinh(x)), and (cosh(x)). These mathematical functions are used in various applications including communication systems, machine learning systems, signal processing systems, and control systems. These mathematical functions are computationally intensive and cannot be efficiently implemented in digital systems using direct methods; therefore, polynomial approximations using techniques such as Taylor series expansion are used. The mathematical concept of Taylor series expansion is a method for approximating transcendental functions by a sum of polynomial terms. The use of a finite number of polynomial

coefficients for transcendental functions makes hardware implementation much easier. However, the direct implementation of a polynomial expression may require several multiplication and addition operations, leading to increased complexity and delay in the hardware implementation of the function. Another method, Horner's rule, is a powerful method for evaluating a polynomial expression by minimizing the number of arithmetic operations through a nested form of the expression.

The proposed architecture incorporates both the standard method of polynomial evaluation and Horner's method using stochastic logic circuit designs based on multiplexers. It is noted that multiplexers are the most suitable devices for implementing stochastic computing since they have the ability to execute weighted selection operations. These operations are equivalent to probabilistic arithmetic. It is possible to efficiently map the coefficients of the polynomials using multiplexers. Among the four functions discussed in the present research work, particular emphasis is given to the implementation and evaluation of the exponential decay function (e^{-x}). This function is of significant importance in a number of engineering applications, including the development of signal attenuation models, the analysis of communication systems, the development of control systems, and the development of neural network functions. Since the function (e^{-x}) has a wide range of applications in various fields of engineering, the efficient implementation of the function on a hardware platform is of significant importance. In order to test the efficiency of the proposed design, simulation is conducted using the Verilog hardware description language and FPGA-based synthesis tools. The stochastic logic is implemented and tested based on the hardware area, computation delay, and approximation accuracy. The comparison of the proposed design and the conventional design is conducted based on two different approaches. The comparison is conducted between the conventional polynomial implementation and the implementation using Horner's rule for polynomial evaluation. The simulation of the proposed stochastic logic is conducted to test its efficiency and its ability to improve the hardware utilization of the binary arithmetic logic. The simulation tests the proposed multiplexer-based stochastic logic and its efficiency in reducing the hardware complexity of the binary arithmetic logic. The simulation also tests the efficiency of the proposed stochastic logic in reducing the computation delay using the polynomial implementation of the proposed design based on Horner's rule.

This is because the stochastic circuits use simple logic gates. This implies that the architecture is expandable to use higher-order polynomial approximations. Therefore, the architecture is useful in the implementation of various VLSI applications that involve function evaluations.

The proposed multiplexer-based stochastic circuit architecture is thus a viable option for the implementation of polynomial computation in VLSI systems. This is due to the fact that the architecture is based on stochastic computing principles. The principles of the architecture are also based on the use of Taylor series approximation and optimization using Horner's rule. This implies that the architecture is efficient in terms of hardware area requirements, power consumption, and computation speed. Therefore, the architecture is highly viable for use in modern digital systems that emphasize low power and speed. The results presented in the work also show the viability of stochastic logic in the implementation of complex mathematical functions. Therefore, the work presented in the paper shows the potential of stochastic computing techniques in the implementation of efficient and compact VLSI architectures.

Keywords: Polynomial approximation, Horner's rule, Taylor series, VLSI architecture, exponential and hyperbolic functions, FPGA implementation, stochastic computing, multiplexer-based design, and low-power digital design.

I. INTRODUCTION

The advanced digital technologies like signal processing systems, artificial intelligence systems, communication systems, and control systems, efficient evaluation of various mathematical functions like exponential functions and hyperbolic functions is necessary[9]. These functions can be evaluated efficiently using polynomial approximations based on Taylor series expansion[6]. In VLSI technology, arithmetic evaluation of polynomial functions using arithmetic logic circuits results in an increase in hardware complexities due to an increase in the number of multipliers and adders.

Recently, stochastic logic has been proposed, and it can efficiently implement various arithmetic operations[4] efficiently. In stochastic logic-based systems, numbers are represented in probability bit stream form[3]. Arithmetic operations like multiplication and addition can be performed efficiently by using logic gates like AND logic gates. This results in reduced hardware complexities and maximizes fault tolerance, making these types of stochastic logic-based arithmetic operations efficient and suitable for low-power and high-speed VLSI technology.

Multiplexers play an important role in implementing stochastic logic-based arithmetic operations[7], and they can be efficiently utilized in achieving weighted addition operations. These types of arithmetic operations can be efficiently performed in evaluating polynomial functions efficiently based on Horner's rule[8].

In this work, polynomial computation with multiplexer-based stochastic logic is implemented for the implementation of four significant mathematical functions: e^x, e^{-x}, sinh(x), and cosh(x). These functions have wide applicability in communication systems and various control systems and signal processing systems. Among these four functions, the implementation and analysis of e^{-x} have been discussed in detail because of its applications in attenuation systems and filter systems.

II. MATHEMATICAL BACKGROUND

The exponential and hyperbolic functions used in this work are approximated using Taylor series expansion[9] around x = 0.The first few terms of the series are used to balance hardware complexity and accuracy.

Exponential Functions

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots \quad (1)$$

$$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \cdots \quad (2)$$

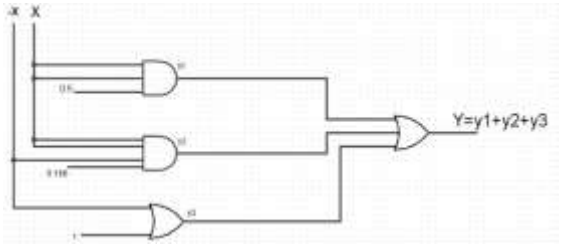


Fig. 1. Logical gate implementation of e^{-x}

Hyperbolic Functions

Hyperbolic sine and cosine are defined using exponential functions.

$$\sinh(x) = \frac{e^x - e^{-x}}{2} \quad (3)$$

$$\cosh(x) = \frac{e^x + e^{-x}}{2} \quad (4)$$

Expanding into Taylor series:

$$\sinh(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots \quad (5)$$

$$\cosh(x) = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots \quad (6)$$

Using these equations, polynomial approximation can be implemented in digital circuits using multiplexer-based stochastic logic.

Polynomial Approximation Order Selection

Higher-order polynomials provide better accuracy but require more hardware resources[7] such as adders, multipliers and stochastic bit-stream length. Therefore, in practical VLSI design[8], 3-term or 4-term approximations are commonly used.

Example 4-Term Approximations

$$e^x \approx 1 + x + \frac{x^2}{2} + \frac{x^3}{6} \quad (7)$$

$$e^{-x} \approx 1 - x + \frac{x^2}{2} - \frac{x^3}{6} \quad (8)$$

$$sinh\,(x) \approx x + \frac{x^3}{6} \quad (9)$$

$$cosh\,(x) \approx 1 + \frac{x^2}{2} + \frac{x^4}{24} \quad (10)$$

detailed analysis is performed for e^{-x}, while the same approach is validated for the other three functions.

Horner's Rule for Polynomial Evaluation

Horner's rule rewrites a polynomial in nested form to reduce the number of multiplications and additions.

General polynomial:

$$P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 \quad (11)$$

Horner form:

$$P(x) = a_0 + x(a_1 + x(a_2 + x(a_3))) \quad (12)$$

Horner Forms of All Four Functions

1. e^x

$$e^x \approx 1 + x\left(1 + x\left(\frac{1}{2} + x\left(\frac{1}{6}\right)\right)\right) \quad (13)$$

2. e^{-x}

$$e^{-x} \approx 1 + x\left(-1 + x\left(\frac{1}{2} + x\left(x - \frac{1}{6}\right)\right)\right) \quad (14)$$

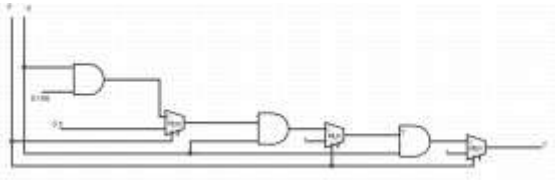


Fig. 2. Horner's logic gate implementation of e^{-x}

3. $sinh\,(x)$

$$sinh\,(x) \approx x\left(1 + x^2\left(\frac{1}{6}\right)\right) \quad (15)$$

4. $cosh\,(x)$

$$cosh\ (x) \approx 1 + x^2\left(\frac{1}{2} + x^2\left(\frac{1}{24}\right)\right) \quad (16)$$

These forms reduce hardware complexity and are suitable for multiplexer-based stochastic implementation.

Direct Polynomial Evaluation (Without Horner's Rule)

In conventional polynomial computation, each term is evaluated separately.

Direct Forms of All Four Functions

$$e^x \approx 1 + x + \frac{x^2}{2} + \frac{x^3}{6} \quad (17)$$

$$e^{-x} \approx 1 - x + \frac{x^2}{2} - \frac{x^3}{6} \quad (18)$$

$$sinh\ (x) \approx x + \frac{x^3}{6} \quad (19)$$

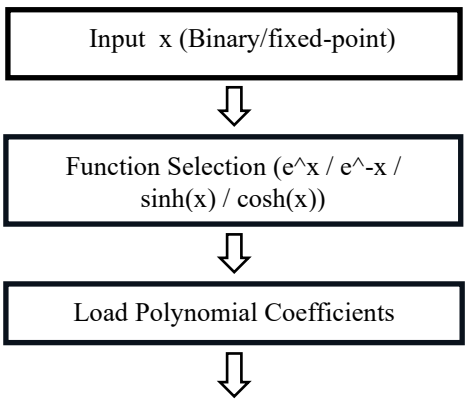
$$cosh\ (x) \approx 1 + \frac{x^2}{2} + \frac{x^4}{24} \quad (20)$$

This method requires more multipliers and adders, increasing circuit area and delay. However, it is useful as a reference to compare with Horner's rule implementation.

III. STOCHASTIC LOGIC FUNDAMENTALS

Stochastic computing represents numerical values using probability-based bit streams[1] instead of conventional binary numbers. In this representation, the value of a number is equal to the proportion of 1s in the bit stream, i.e., $x = P(X = 1)$for unipolar encoding. Arithmetic operations can then be implemented using simple logic gates[3]. For example, multiplication of two stochastic numbers is realised using an AND gate, $z = x \cdot y$, while weighted addition is performed using a multiplexer, $z = px + (1 - p)y$. Because complex multipliers and adders are avoided, stochastic logic significantly reduces hardware area and power consumption[2].

In this work, stochastic logic is used to compute polynomial approximations of e^x, e^{-x}, $sinh\ (x)$, and $cosh\ (x)$[12]. Binary inputs are converted into stochastic bit streams using linear feedback shift register (LFSR)–based random number generators. Polynomial evaluation is then performed using cascaded multiplexer stages following Horner's rule. Although stochastic computing may require longer bit streams[16] for higher accuracy, it offers advantages such as low hardware complexity, fault tolerance, and efficient VLSI implementation for nonlinear mathematical functions.



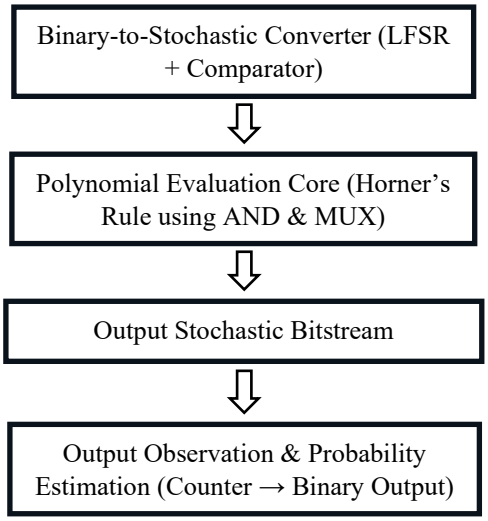


Fig. 3. Flowchart of the proposed multiplexer-based stochastic polynomial computation system.

IV. RESULTS AND DISCUSSION

The performance of the proposed multiplexer-based stochastic polynomial computation architecture was evaluated and compared with the conventional method of implementing the polynomial function. The comparison of the proposed method and the conventional method was based on various parameters of the VLSI design, i.e., power consumption, hardware area, and timing delay. The polynomial function was implemented using arithmetic logic circuits that include various multipliers and adders in the conventional method. The hardware of the conventional method is complex due to the use of various multipliers and adders. On the other hand, the stochastic architecture was employed to implement the polynomial function in the proposed method. The proposed method requires simple logic circuits that include multiplexers and AND logic gates. The proposed method is simple compared to the conventional method. The proposed method was evaluated using various tools in the hardware design environment. The functions such as e^{x}, e^{-x}, sinh(x), and cosh(x) were implemented and evaluated to measure the performance of the proposed method.

TABLE I
Power, area and timing results using conventional method

Function	Power (µW)	Area (µm²)	Timing (ns)
cosh(x)	383.48	4539.12	2.82
sinh(x)	154.08	4846.43	2.83
e^{x}	191.99	5929.55	2.86
e^{-x}	204.04	6089.26	6.12

Table I shows the synthesis results obtained using the conventional polynomial computation method. The results indicate higher power consumption and larger hardware area due to the use of multiple multipliers and adders.

TABLE II
Power, Area and Timing Results Using Stochastic Method

Function	Power (µW)	Area (µm²)	Timing (ns)
cosh(x)	17.74	186.95	0.69
sinh(x)	17.81	190.73	0.69
e^x	17.75	185.44	0.69
e^{-x}	28.18	244.47	5.45

Table II summarises the synthesis results obtained using the proposed multiplexer-based stochastic architecture. Compared with the conventional implementation, the stochastic approach considerably decreases both power consumption and hardware area because complex arithmetic units are replaced with simple logic gates and multiplexers. The measured power consumption varies between 17 µW and 28 µW for the functions e^x, e^{-x}, $sinh(x)$, and $cosh(x)$. In addition, the hardware area is reduced to nearly 185-244 µm², showing a compact circuit design. From the timing result, it is clear that a delay of 0.69 ns is required for most functions, while a slightly higher delay is required for the e^(-x) function, considering the additional polynomial stages. From these observations, it is clear that the stochastic implementation is an efficient solution for low-power and area-optimised VLSI systems.

TABLE III
Percentage improvement of stochastic implementation

Function	Power Reduction	Area Reduction
cosh(x)	95.4% ↓	95.9% ↓
sinh(x)	88.4% ↓	96.1% ↓
e^x	90.8% ↓	96.9% ↓
e^{-x}	86.2% ↓	96.0% ↓

Table III indicates that the proposed stochastic architecture provides a significant improvement over the conventional approach. The results show a power reduction ranging from approximately 86% to 95%, while the hardware area is reduced by nearly 96%, demonstrating the efficiency of the stochastic implementation.

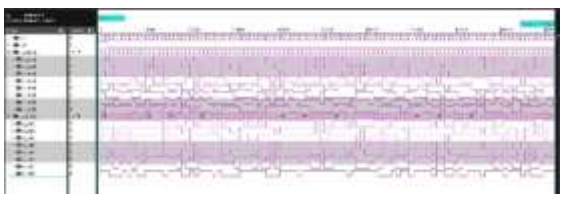


Fig. 4. Cadence simulation waveform of the stochastic implementation of the e^{-x}function.

The waveform presented in Fig. 4 describes the Cadence simulation outcomes of the proposed stochastic architecture for the computation of the exponential decay function given by e^{-x}. The proposed stochastic architecture uses stochastic bit streams to represent numerical values. The stochastic bit streams are produced using LFSR-based random number generators. The arithmetic logic needed for the computation of polynomial functions is implemented using simple logic gates such as multiplexers and AND gates.

The waveform describes the stochastic nature of the input and output bit streams during the computation of the polynomial function. The simulation outcomes validate the functional correctness of the proposed multiplexer-based stochastic logic architecture and its ability to evaluate nonlinear mathematical functions using less hardware.

Fig. 5. Cadence simulation waveform of the conventional polynomial implementation of the e^{-x}function.

The waveform produced by the Cadence simulator for the conventional method of implementation of the function 'e^{-x}' using the polynomial method is as given in Fig. 5 below. In the conventional method of implementing the function ' e^{-x} 'using the polynomial method, the evaluation of the equations of the polynomials is carried out by using arithmetic logic circuits such as 'multipliers' and 'adders.' The waveform as given in Fig. 5 represents the relationship between the input 'x' and the corresponding output 'y' produced by the system. From the waveform as given in Fig. 5, it is clear that the complexity of the arithmetic logic circuits in the conventional method is more compared to the stochastic method.

The stochastic waveform demonstrates probabilistic bit-stream characteristics because numerical values are represented using stochastic encoding. On the other hand, the normal waveform demonstrates deterministic characteristics of a signal transition using arithmetic logic circuitry. Although both methods accurately implement the e^{-x} function calculation, the stochastic method uses simple hardware components such as multiplexers and logic gate circuitry.

V. CONCLUSION

The paper proposed an efficient architecture based on multiplexers for stochastic logic in polynomial computations in VLSI systems. Polynomial approximations of functions like e^x, e^(-x), sinh(x), and cosh(x) have been implemented using conventional polynomial evaluation and Horner's method. In the proposed stochastic method, arithmetic operations are performed using simple logic blocks like multiplexers and AND gates[20].

The synthesis results demonstrate that the stochastic architecture greatly reduces power consumption and hardware area while maintaining good timing performance. Comparing to conventional arithmetic circuits, the proposed architecture greatly outperforms in power efficiency and hardware compactness. These results demonstrate that multiplexer-based stochastic computing is a promising solution to efficiently implement nonlinear mathematical functions[19] in low-power VLSI systems. Future work can be devoted to hardware evaluation using FPGA-based prototyping, enhancing computational accuracy by minimizing stochastic bit stream length, and extending the proposed architecture to support more mathematical functions and machine learning accelerators.

REFERENCES

1. A. Alaghi and J. P. Hayes,“Survey of stochastic computing,”ACM Trans. Embedded Comput. Syst.,

vol. 12,no. 2s,pp. 92:1–92:19,May 2013.
2. A. Alaghi and J. P. Hayes,“Principles of stochastic computing,”IEEE Design & Test,vol. 33,no. 2,pp. 6–15,Apr. 2016.
3. S. Li,J. Han,and F. Lombardi,“A survey of stochastic computing,”IEEE Trans. Emerging Topics Comput.,vol. 7,no. 4,pp. 513–528,Oct.–Dec. 2019.
4. S. Liu,B. Li,and J. Han,“Stochastic computing: A more than 50-year journey,”IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.,vol. 38,no. 10,pp. 1878–1890,Oct. 2019.
5. Y. Liu et al.,“Review of stochastic computing architecture and applications,”IEEE Access,vol. 8,pp. 187240–187260,2020.
6. W. Qian and M. D. Riedel,“The synthesis of robust polynomial arithmetic with stochastic logic,”in Proc. IEEE Design Automation Conf.(DAC),2008,pp. 648–653.
7. K. K. Parhi,“Stochastic logic implementations of polynomials with all positive coefficients by expansion methods,”IEEE Trans. Circuits Syst. I,vol. 65,no. 3,pp. 987–998,Mar. 2018.
8. Y. Liu and K. K. Parhi,“Computing polynomials using unipolar stochastic logic,”ACM Trans. Design Autom. Electron. Syst.,vol. 21,no. 4,pp. 1–23,Jul. 2016.
9. K. K. Parhi and Y. Liu,“Computing arithmetic functions using stochastic logic by series expansion,”IEEE Trans. Circuits Syst. I,vol. 66,no. 5,pp. 1819–1831,May 2019.
10. J. Li and H. Zhou,“Area-efficient stochastic multipliers for VLSI applications,”IEEE Trans. Circuits Syst. II,vol. 64,no. 5,pp. 525–529,May 2017.
11. J. Haselmayr,M. Huemer,and C. Pacher,“High-accuracy stochastic inner product design,”IEEE Trans. Circuits Syst. I,vol. 65,no. 9,pp. 2860–2873,Sep. 2018.
12. M. Najafi,D. Jenson,and K. Bazargan,“Reliable and energy-efficient stochastic logic design,”IEEE Trans. Nanotechnology,vol. 16,no. 3,pp. 456–466,May 2017.
13. J. Kim,D. Kim,and S. Kang,“Energy-efficient FPGA implementation of stochastic computing,”IEEE Trans. Very Large Scale Integr.(VLSI) Syst.,vol. 26,no. 6,pp. 1230–1242,Jun. 2018.
14. H. Li,A. Alaghi,and J. P. Hayes,“Low-power stochastic computing circuits for DSP applications,”IEEE Trans. Circuits Syst. I,vol. 66,no. 10,pp. 3890–3902,Oct. 2019.
15. A. Ardakani,M. Ardakani,and W. J. Gross,“VLSI implementation of deep neural network using integral stochastic computing,”IEEE Trans. Very Large Scale Integr.(VLSI) Syst.,vol. 25,no. 10,pp. 2688–2699,Oct. 2017.
16. D. Li,Y. Wang,and J. Han,“Correlation-aware stochastic computing circuits,”IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.,vol. 39,no. 8,pp. 1565–1578,Aug. 2020.
17. F. Ren,J. Li,J. Han,and F. Lombardi,“Accuracy improvement techniques in stochastic computing,”IEEE Trans. Computers,vol. 69,no. 3,pp. 451–463,Mar. 2020.
18. Y. Chen,X. Wang,and J. Han,“Low-latency stochastic arithmetic units for FPGA,”IEEE Access,vol. 9,pp. 122456–122468,2021.
19. Z. Li,S. Liu,and J. Han,“Hybrid binary–stochastic computing architectures,”IEEE Trans. Emerging Topics Comput.,vol. 10,no. 2,pp. 945–957,Apr.–Jun. 2022.
20. Y.-N. Chang and K. K. Parhi,“Architectures for digital filters using stochastic computing,”in Proc. IEEE Int. Conf. Acoust.,Speech Signal Process.(ICASSP),Vancouver,BC,Canada,May 2013,pp. 2697–2701.
21. S. Agwa et al.,“Digital in-memory stochastic computing architecture for vector–matrix multiplication,”Frontiers in Nanotechnology,vol. 5,2023.

22. S. Liu,Y. Wang,and J. Han,"Design of arithmetic circuits in stochastic computing: A review,"IEEE Nanotechnology Magazine,vol. 18,no. 2,pp. 45–55,Apr. 2024.