

Article

A High-Accuracy Normalization Unit Using Multi-Bit Random Variables

Yubin Zhu , Kaining Han *  and Jianhao Hu

National Key Laboratory of Wireless Communications, University of Electronic Science and Technology of China, Chengdu 611731, China

* Correspondence: hkaining2014@gmail.com

Abstract

Stochastic computing (SC) has the characteristics of low complexity and is expected to solve the bottleneck problem of conventional binary computing. Stochastic normalization units are widely used in stochastic decoding and stochastic signal detection, and have achieved hardware efficiency far exceeding conventional methods. However, they also have problems such as 1-bit representation and low calculation accuracy, calculation overflow, and fluctuation in the sum of normalized probabilities, which lead to prolonged processing latency and degraded hardware efficiency. Thus, this paper proposes a novel stochastic normalization unit using multi-bit random variables. Benefiting from the high representation accuracy of multi-bit random variables, the accuracy of the proposed unit is greatly improved. Meanwhile, the proposed unit completely solves the problems of calculation overflow and fluctuation in the sum of normalized probabilities. Simulation results show that the proposed 3-bit unit achieves a fourfold improvement in convergence speed and 2 times higher hardware efficiency compared to the state-of-the-art stochastic normalization unit. Finally, we verify that the proposed 3-bit unit demonstrates a 75% improvement in hardware efficiency for stochastic sparse-code multiple-access (SCMA) detection.

Keywords: stochastic computing; stochastic circuits; stochastic normalization unit; sparse code multiple access (SCMA); hardware efficiency



Academic Editor: Fabio Postiglione

Received: 4 September 2025

Revised: 7 October 2025

Accepted: 9 October 2025

Published: 14 October 2025

Citation: Zhu, Y.; Han, K.; Hu, J. A High-Accuracy Normalization Unit Using Multi-Bit Random Variables. *Electronics* **2025**, *14*, 4042. <https://doi.org/10.3390/electronics14204042>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As a novel computing paradigm, stochastic computing (SC) [1] has the characteristics of low complexity and is expected to solve the bottleneck problem of conventional binary computing [2]. Stochastic computing is a unary representation and computing system that represents a number with an unweighted stochastic bit stream, where the proportion of 1s in the stream represents the numerical value. The most attractive feature of stochastic computing is that complex mathematical operations in conventional binary computing systems can be implemented with basic logic gates. For linear operations, AND gates and multiplexers (MUXs) can conduct multiplication and multiply-accumulate (MAC) operations, respectively [3]. For nonlinear operations, the JK flip-flop can be used to implement a normalization operation of two inputs [4].

Stochastic normalization units are widely used in stochastic decoding [4–7] and stochastic signal detection [8,9]. Although there have been many works aiming to improve their convergence speed and accuracy, stochastic normalization units still face a long bit-stream length due to the limitation of the representation ability of stochastic bits [10,11].

Another problem is that non-scaling addition between stochastic bits is difficult to implement, which can lead to calculation overflow and loss of precision, while the latter can lead to fluctuation in the sum of normalized probabilities. The Integral Stochastic Computing (ISC) [12] and amplitude and frequency encoding (AFE) for SC [13] demonstrate a promising way to further improve the representation and calculation accuracy by using multi-bit random variables. ISC and AFE proved that multi-bit random variables can be used in addition, multiplication, and other calculations, which provides the stochastic normalization unit a way to further improve accuracy.

In this study, we improve the accuracy and convergence speed of the stochastic normalization unit by using the idea of multi-bit random variables, and solve the problems of calculation overflow and fluctuation in the sum of normalized probabilities. Simulation results demonstrate that our 3-bit implementation achieves fourfold improvement in convergence speed and doubles the hardware efficiency compared to conventional stochastic normalization units. Furthermore, hardware implementation for stochastic sparse-code multiple-access (SCMA) detection shows a 75% improvement in hardware efficiency, validating the practical benefits of our approach.

The rest of this paper is organized as follows: Section 2 briefly reviews the preliminaries of existing stochastic normalization units and multi-bit random variables. Section 3 describes the proposed stochastic normalization unit and the further improvements. The computing and hardware performance will be presented in Section 4. Finally, the paper concludes in Section 5.

2. Preliminaries

2.1. Existing Stochastic Normalization Units

In stochastic computing (SC), a numerical value x ($0 \leq x \leq 1$) is represented by the proportion of '1's in a stochastic bit stream, with each bit generated by a binary random variable X . Such streams are commonly produced using a comparator and a random number source (RNS), as shown in Figure 1a. A major benefit of SC lies in its ability to perform complex arithmetic operations using simple logic circuits. For example, multiplying two independent stochastic bit streams can be achieved efficiently with a single AND gate, while more intricate multiply–accumulate (MAC) operations can be implemented using a multiplexer (MUX) architecture, as illustrated in Figure 1b and Figure 1c, respectively.

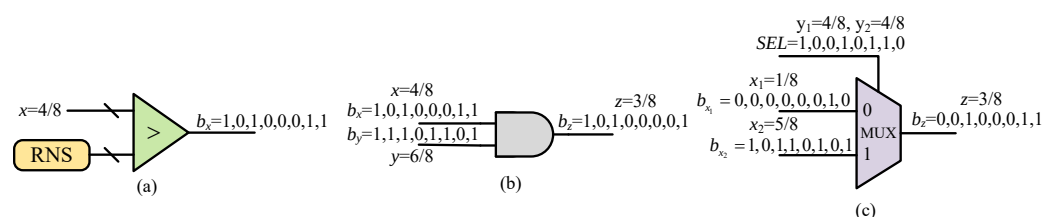


Figure 1. (a) Stochastic Number Generator. (b) The AND gate-based multiplying function. (c) The MUX-based MAC function.

To implement the normalization function in stochastic computing (SC), we first require a division operation. SC dividers are typically implemented as sequential circuits using Markov processes, as illustrated in Figure 2. A classic approach employs a JK flip-flop, which inherently performs normalization on two input streams [14]. Alternatively, counter-based dividers [15] have been proposed to compute binary division results, where multi-bit tracking enables precise division before reconversion to stochastic bit streams.

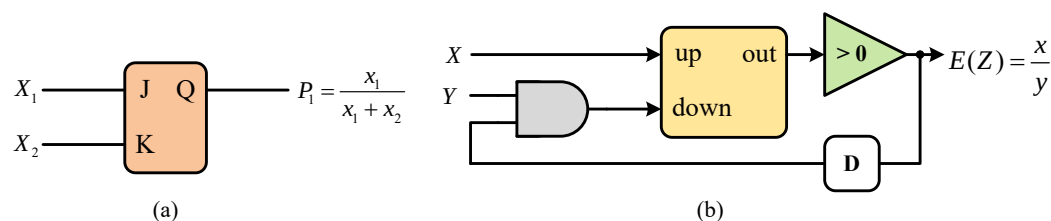


Figure 2. (a) Divider based on the JK flip-flop [14]. (b) Divider based on the counter [15].

The normalization unit for M inputs can be constructed using the aforementioned divider combined with linear operations. A straightforward approach extends the JK flip-flop design (Figure 3a), though this cannot directly output binary-normalized probabilities. Alternatively, counter-based normalization (Figure 3b) tracks probabilities explicitly [16]. While both methods employ MUX-based scaled addition to prevent overflow, they sacrifice computational accuracy.

The joint probability tracking (JPT) method [7,9] improves precision through non-scaled addition and tracking forecast memory (TFM). To address overflow, JPT records overflow counts and compensates by skipping updates (e.g., canceling four updates as shown in Figure 4). However, this effectively shortens the bit-stream length, degrading representation and computational accuracy.

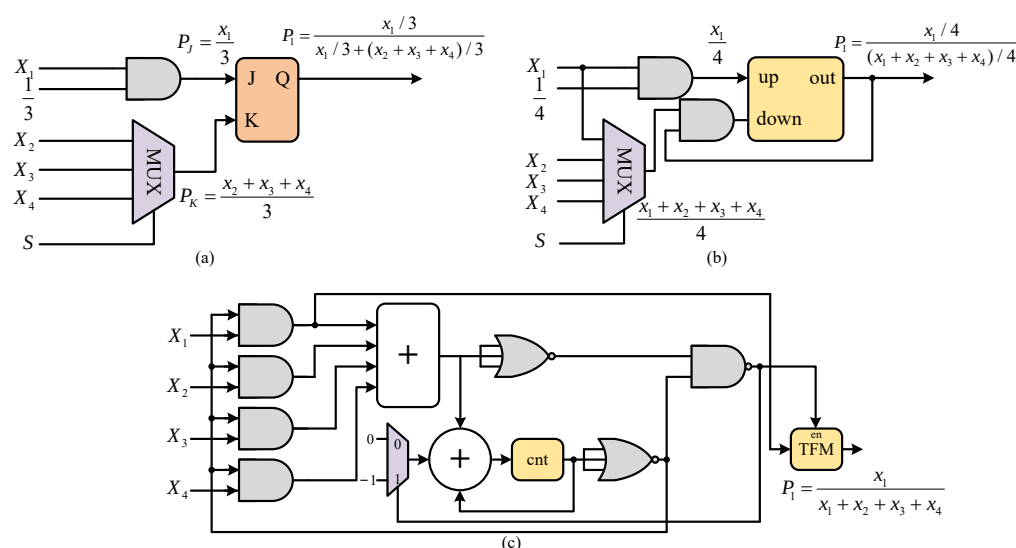


Figure 3. (a) Normalization unit based on MUX and JK flip-flop [17]. (b) Normalization unit based on counter [16]. (c) Normalization unit based on IPT method [79].

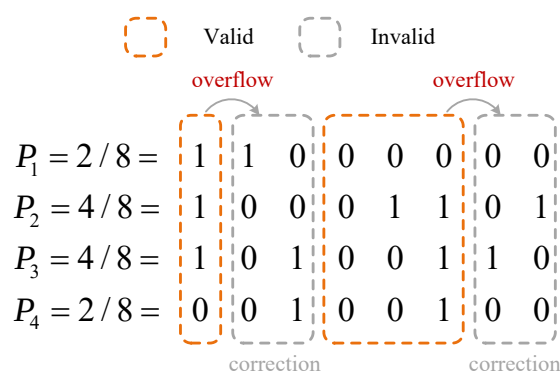


Figure 4. The method of solving calculation overflow in the JPT method [7,9].

In summary, due to the nature of stochastic bits, both scaling and non-scaling addition will result in a loss of precision. Moreover, the reduction in precision will also result in fluctuations in the sum of normalized probabilities. To solve the above problems, a new computing and representation system is required.

2.2. Multi-Bit Random Variables

Conventional SC is represented and calculated using 1-bit random variables. The Integral Stochastic Computing (ISC) [12] and amplitude and frequency encoding (AFE) for SC [13] demonstrate a promising way to further improve the representation and calculation accuracy by using multi-bit random variables, as shown in Figure 5a and Figure 5b, respectively. In multi-bit random variables, for a number x , its multi-bit random variable X should satisfy $E(X) = x$ as well. ISC and AFE proved that multi-bit random variables can be used in addition, multiplication, and other calculations, as shown in Figure 6, which provides the stochastic normalization unit a way to further improve accuracy.

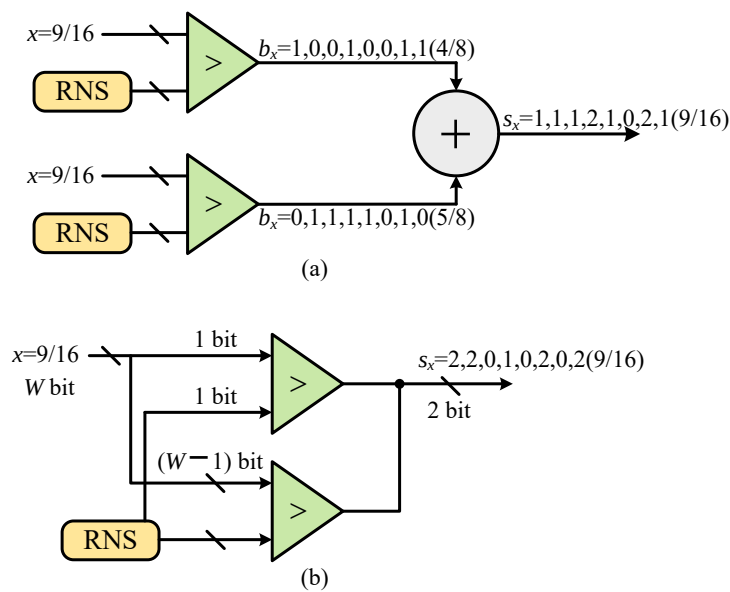


Figure 5. (a) The generator of ISC [12]. (b) The generator of AFE [13].

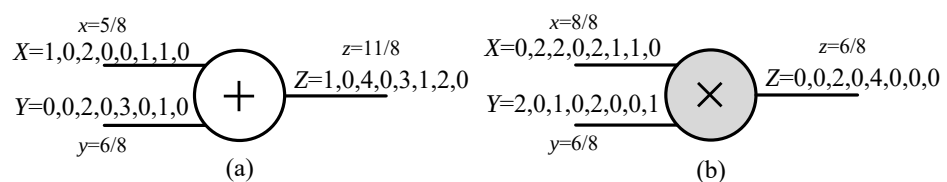


Figure 6. (a) Addition of multi-bit random variables [12]. (b) Multiplication of multi-bit random variables [12].

3. Joint Normalization Unit for Multi-Bit Random Variable

3.1. High-Precision Multi-Bit Random Variable Generation

For higher representation accuracy, we propose applying a W -bit binary number to an n -bit random variable generator as shown in Figure 7a. We retain the highest $n - 1$ bits, generate the probability bit of the lower bit, and add it to the higher bit to get the n -bit random variable. Figure 7c shows the mean square error (MSE) when different methods are used to represent different values of x . Results show that the MSE of the proposed method is much smaller than that of stochastic bit and other multi-bit random variables, which makes it possible to improve the accuracy of the normalization unit.

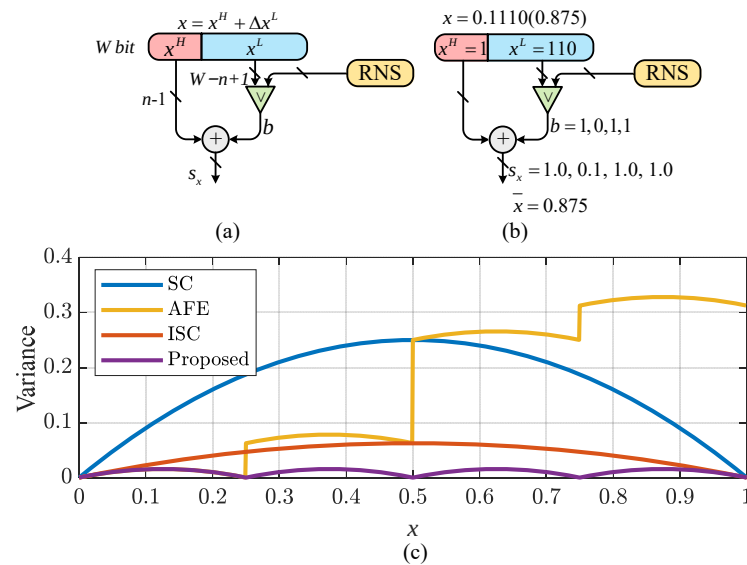


Figure 7. (a) The proposed generator of a multi-bit random variable, where x^H and x^L represent the highest $n - 1$ bits and the lower bit, respectively. (b) An example of the generation of a 2-bit multi-bit random variable. (c) The MSE of the conventional SC [1], AFE [13], ISC [12] and the proposed multi-bit random variable representations with 3 bit width.

3.2. Normalization Unit for Multi-Bit Random Variable

It has been shown that multi-bit random variables can be added and multiplied [12], so they can also be divided according to Markov processes. Figure 8a shows the divider for a multi-bit random variable. Assuming that a steady state has been reached, the result of the next cycle z^{t+1} can be calculated by the current input X^t , Y^t , and the current result z^t as

$$z^{t+1} = z^t + \eta(X^t - Y^t \cdot z^t), \quad (1)$$

where η is the step length. Obviously, this is a Markov process. When this Markov process converges, we find the expectation of both sides of the equation at the same time, and we have

$$\begin{aligned} E(z^{t+1}) &= E(z^t) + \eta(E(X^t) - E(Y^t) \cdot E(z^t)), \\ E(z^{t+1}) &= E(z^t) = E(z), \end{aligned} \quad (2)$$

namely,

$$E(z) = \frac{E(x)}{E(y)} = \frac{x}{y}. \quad (3)$$

To implement the normalization unit, replace Y with the sum of the inputs, as shown in Figure 8b. The addition of multi-bit random variables can naturally expand the bit width, so the addition here will not cause calculation overflow. Meanwhile, the result of the addition can be shared by different dividers.

3.3. Constant Sum of Normalized Probabilities

In the above probability normalization unit, the loss of addition calculation accuracy may cause fluctuations in the sum of normalized probabilities. However, this problem does not exist in the proposed structure. Suppose the input multi-bit random variables $\{X_1^t, X_2^t, \dots, X_M^t\}$ satisfy $E[X_i^t] = x_i$; then, the i -th normalized probability updates as

$$P_i^{t+1} = P_i^t + \eta(X_i^t - \sum_{k=1}^M X_k^t \cdot P_i^t). \quad (4)$$

We sum all the tracking probabilities with the assumption that the sum of the probabilities in the previous iteration is 1, namely, $\sum_{i=1}^M P_i^t = 1$; then,

$$\begin{aligned} \sum_{i=1}^M P_i^{t+1} &= \sum_{i=1}^M \left[P_i^t + \eta \left(X_i^t - \sum_{k=1}^M X_k^t \cdot P_i^t \right) \right] \\ &= \sum_{i=1}^M P_i^t + \eta \sum_{i=1}^M X_i^t - \eta \sum_{k=1}^M X_k^t \cdot \sum_{i=1}^M P_i^t = 1. \end{aligned} \quad (5)$$

Equation (5) shows that as long as we set $P_i^0 = \frac{1}{M}$ to satisfy $\sum_{i=1}^M P_i^0 = 1$ during initialization, $\sum_{i=1}^M P_i^t = 1$ can always be maintained in subsequent iterations. Therefore, in order to reduce the complexity and the impact of limited bit width on accuracy, we can cancel the last probability tracking and set the probability $P_M^t = 1 - \sum_{i=1}^{M-1} P_i^t$.

To verify the above conclusion, we assume that $M = 4$ and the inputs $\{x_1, x_2, x_3, x_4\} = \{0.2, 0.4, 0.6, 0.8\}$, and the normalized probabilities are $\{P_1, P_2, P_3, P_4\} = \{0.1, 0.2, 0.3, 0.4\}$. Suppose $\eta = 2^{-4}$; Figure 9 shows that the proposed method can ensure that the sum of the normalized probabilities is 1, while other methods cannot, completely solving the calculation overflow and fluctuation problems.

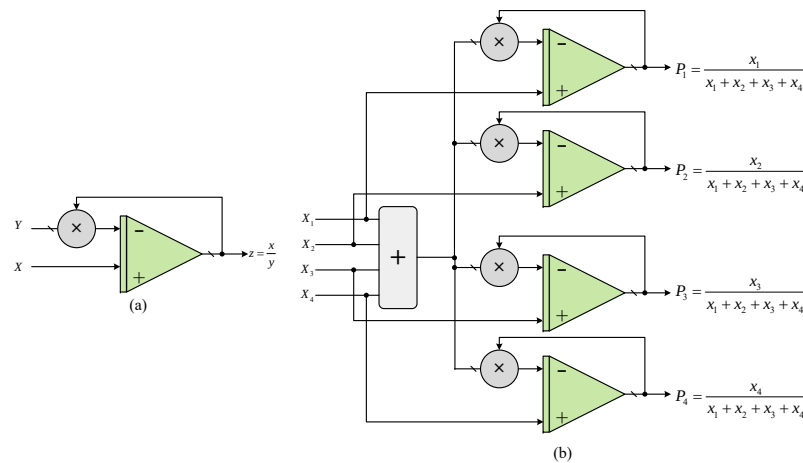


Figure 8. (a) Divider for multi-bit random variables. (b) Normalization unit based on multi-bit random variables.

3.4. Re-Randomize According to Normalized Probabilities

The output of the probability normalization unit is often the tracked probability mass function (PMF). But in stochastic LDPC decoding and SCMA detection, a multi-bit random variable Z according to the tracked PMF should be generated. This is usually achieved through cumulative distribution function (CDF) sampling, which means that some additional adders are required to calculate the CDF, as mentioned in [9].

To reduce hardware overhead, the proposed structure can also track the CDF instead of the PMF, as shown in Figure 10. In the circuit that tracks the CDF, not only the sum of all inputs is needed, but also their cumulative sum one by one, which does not incur any additional hardware overhead. Similarly, only $M - 1$ probabilities need to be tracked, and the probabilities should be initialized as $P_i^0 = \frac{i}{M}$.

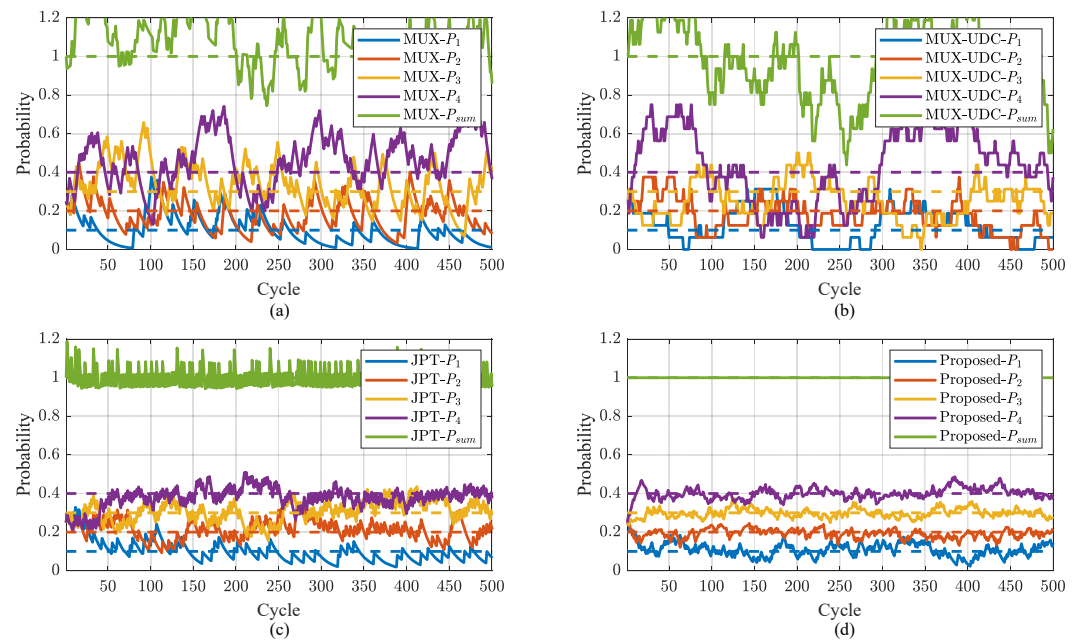


Figure 9. The normalized probabilities and their sum. (a) MUX [17]. (b) MUX-UDC [16]. (c) JPT [7,9]. (d) Proposed unit when $n = 2$.

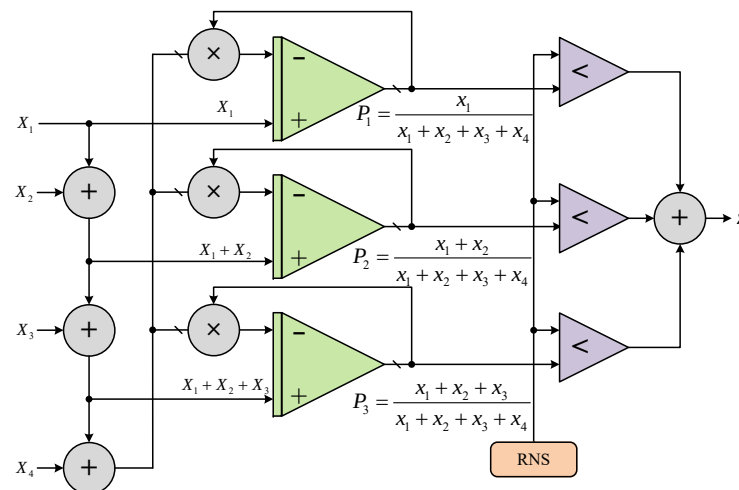


Figure 10. Normalization unit based on multi-bit random variables with CDF sampling.

However, it should be noted that the CDF sampling step is a randomization process. It may introduce large representation noise, significantly reducing the potential gain brought by the multi-bit random variable. To reduce the impact of this noise, we use low-discrepancy (LD) sequences [18,19], such as Sobol sequences, as the random numbers of the CDF sampling.

4. Performance Simulation and Complexity Analysis

4.1. Performance Simulation

To illustrate the computational accuracy of the proposed method, we randomly generate normalized inputs $\{x_1, x_2, x_3, x_4\}$ with $M = 4$ and $\eta = 2^{-4}$. We compare the performance differences between different methods by calculating the MSE of the probability distribution of Y . For a fair comparison, the MUX method [17] is augmented with TFM modules for probability tracking and generation of Y . As mentioned in Section 3, the process of generating Y introduces additional noise, which limits the improvement in tracking probability accuracy brought about by the increase in bit width. Figure 11 shows the

performance of X_i using uniformly distributed random numbers and the Sobol sequence for different bit widths. When using uniformly distributed random numbers, the MSE can hardly be reduced after the bit width exceeds 2 bits. When using the Sobol sequence, the performance can be significantly improved when the stochastic stream length is long.

The comparison of the proposed method and other methods is shown in Figure 12. It can be seen that the performance of the proposed method is significantly higher than that of the state-of-the-art methods, and the convergence speed to achieve the same MSE is more than twice as fast.

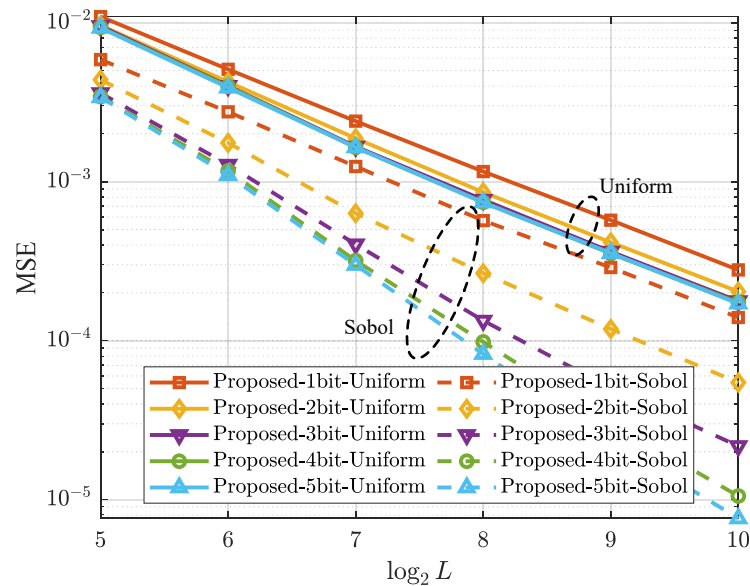


Figure 11. The MSE of the proposed method using uniformly distributed random numbers and the Sobol sequence.

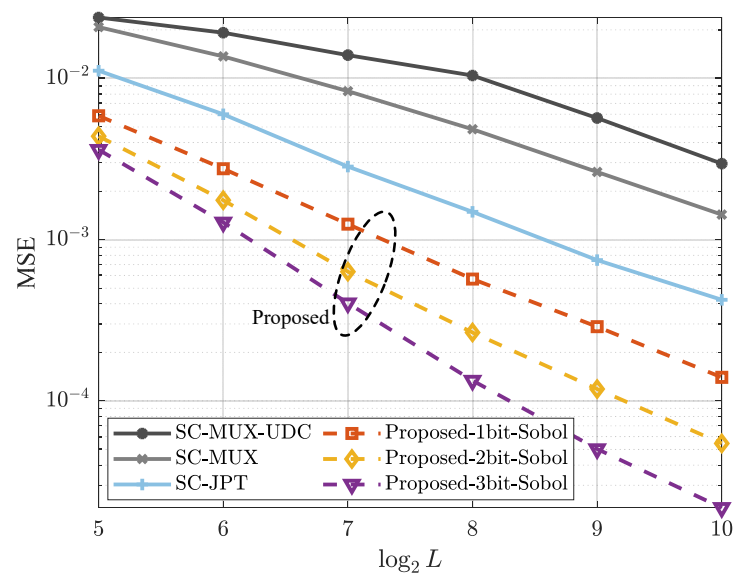


Figure 12. The MSE of the proposed method and conventional SC based MUX-UDC [16], MUX [17], JPT [7] methods.

4.2. Hardware Implementation

We synthesize all the structures under Semiconductor Manufacturing International Corporation (SMIC) 65 nm CMOS technology with the Synopsys Design Compiler (DC). The throughput–area ratio (TAR) is defined by the ratio of throughput (TP) to the area, as

$$\text{TAR} = \frac{\text{TP}}{\text{Area}}, \quad (6)$$

which can be used to survey the hardware efficiency. We implement the proposed structure with bit widths of 1, 2, 3, and 4 bits. We compare the hardware implementation results of the proposed structure and other structures when the MSE is roughly equal, as shown in Tables 1 and 2.

It can be seen that the area of the proposed structure increases linearly with the increase in input bit width. However, increasing the bit width in this MSE interval cannot significantly improve the convergence speed, so the hardware efficiency of the proposed structure reaches its maximum when the input bit width is 3 bits. Although the proposed architecture contains multiplication, one of the multipliers has a small bit width and does not significantly increase the critical path compared to the JPT method. In summary, the proposed scheme achieves $2\times$ the hardware efficiency of the JPT scheme.

Table 1. Hardware implementation of the proposed structure for different bit widths.

Structure	Proposed			
Technology	65 nm			
Bit Width	1	2	3	4
Area (μm^2)	1336.32	1754.64	2156.04	2528.28
Frequency (MHz)	500	500	500	500
MSE	1.2×10^{-3}	1.3×10^{-3}	1.3×10^{-3}	1.2×10^{-3}
MSE Convergence Cycle	128	80	64	64
Throughput (MS/s)	3.9	6.3	7.8	7.8
TAR (MS/(s · mm^2))	2.9×10^3	3.6×10^3	3.6×10^3	3.1×10^3

Table 2. Hardware implementation comparison with different schemes.

Structure	Proposed	JPT [7]	MUX [17]	MUX-UDC [16]
Technology	65 nm	65 nm	65 nm	65 nm
Bit width	3	1	1	1
Area (μm^2)	2156.04	1133.64	1002.24	743.40
Frequency (MHz)	500	500	500	1000
MSE	1.3×10^{-3}	1.5×10^{-3}	1.3×10^{-3}	1.3×10^{-3}
Cycle	64	256	1024	2048
Throughput (MS/s)	7.8	2.0	0.5	0.5
TAR (MS/(s · mm^2))	3.6×10^3	1.7×10^3	0.49×10^3	0.66×10^3
TAR Ratio	2.10	1.00	0.28	0.38

4.3. Application Verification

In algorithms represented by message-passing algorithms (MPAs), normalization calculations are widely used. For example, in MPAs, variable nodes (VNs) need to calculate the normalization of the product of messages passed by adjacent factor nodes (FNs) as

$$P(x = c) = \frac{\prod_i P_i(x = c)}{\sum_{a \in S} \prod_i P_i(x = a)}, \quad (7)$$

where S is the set of all possible symbols. It can be seen that it has a very high complexity, and the complexity of the FNs may be even higher, which is also the main problem of the MPA.

Sparse code multiple access (SCMA) is a promising non-orthogonal multiple-access (NOMA) technology candidate for the next-generation communication system. However, messaging-passing algorithm (MPA)-based SCMA detection can achieve near-maximum-likelihood (ML) performance with extremely high complexity [20]. To address this problem, some other NOMA methods and their detection methods [21,22] have been proposed, which can also achieve performance close to ML detection through linear or macrosymbol detection. Despite their lower complexity, these methods are not directly applicable to SCMA detection. Recently, the stochastic SCMA detector [9] used the idea of stochastic computing, replaced complex FN calculations with MUXs, and used the JPT normalization unit for normalization in VNs. To further improve the hardware efficiency, we use the proposed normalization unit in the VNs. We used a 1/3-code-rate Turbo code from the LTE standard, and the SCMA system has $K = 6$ users spread over $N = 4$ resource elements (REs) with a modulation order of 4. The bit error ratio (BER) performance simulation results and hardware implementation results are shown in Figure 13 and Table 3, respectively.

It can be seen that when the bit width of the proposed scheme is 1, 2, 3, and 4 bits, it takes 100, 60, 45, and 40 decoding cycles (DCs) to achieve similar performance. Further increasing the bit-width accuracy will not significantly speed up the convergence, but the area increases linearly. Therefore, when the input bit width is 3 bits, the hardware efficiency reaches its maximum, which is about 1.75 times that of the JPT method.

Table 3. Hardware implementation comparison for SCMA detector.

Structure		Proposed		JPT [7,9]
Technology		65 nm		65 nm
Bit Width	2	3	4	1
Area (μm^2)	1754.64	2156.04	2528.28	1133.64
Frequency (MHz)	500	500	500	500
BER (SNR = 4dB)	1.0×10^{-5}	2.2×10^{-5}	2.2×10^{-5}	2.0×10^{-5}
Decoding Cycle	60	45	40	150
Throughput (MS/s)	8.3	11.1	12.5	3.3
TAR (MS/(s · mm ²))	4.8×10^3	5.2×10^3	4.9×10^3	2.9×10^3
TAR	1.62	1.75	1.68	1.00

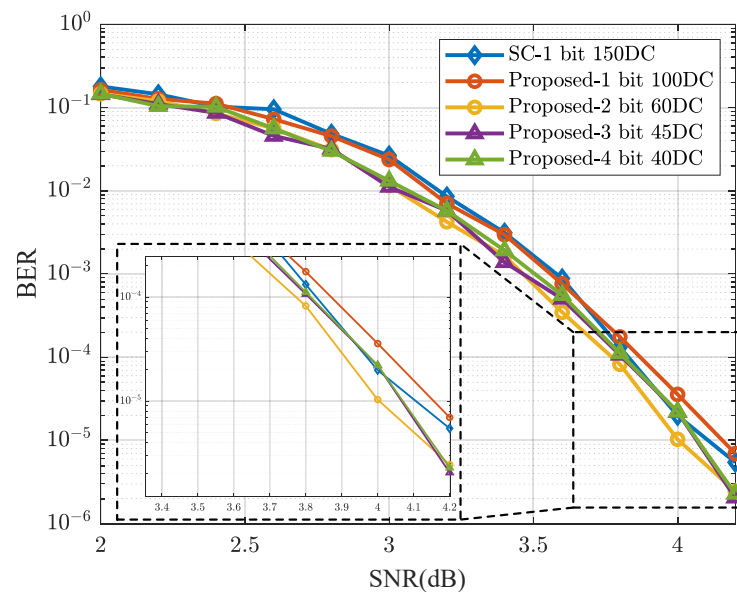


Figure 13. The SCMA BER performance of the JPT method [7,9] and the proposed method.

Similarly, the proposed stochastic normalization unit can also be used in MPA-based MIMO detection. There has been some work using message passing based on conventional stochastic computing [23–25]. These studies also involve normalization units, and we believe that the proposed scheme can also be used in MIMO detection to enable more efficient computation. In addition, the Softmax function, which is widely used in the field of artificial neural networks (ANNs) [26], also contains a normalization function, so the proposed unit is also expected to be applied in the field of neural network accelerators.

5. Conclusions

A multi-bit random variable is a promising way to further improve the representation and calculation accuracy of stochastic computing (SC). Based on it, we propose a high-accuracy normalization unit. It completely solves the problem of the sum of normalized probabilities not being equal to 1, caused by scaling and overflow in stochastic bit computing. Compared to other state-of-the-art stochastic normalization units, it converges 4 times faster and achieves 2.5 times higher hardware efficiency. It provides new possible solutions for the efficient implementation of SCMA detection, MIMO detection, and neural network accelerators.

Author Contributions: Conceptualization, Y.Z., K.H. and J.H.; Methodology, Y.Z.; Investigation, Y.Z. and K.H.; Writing—original draft preparation, Y.Z.; Writing—review and editing, K.H. and J.H.; Funding acquisition, K.H.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 62371099.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gaines, B.R. Stochastic computing. In Proceedings of the Spring Joint Computer Conference, Atlantic City, NJ, USA, 18–20 April 1967; pp. 149–156.

2. Alaghi, A.; Qian, W.; Hayes, J.P. The promise and challenge of stochastic computing. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2017**, *37*, 1515–1531.
3. Joe, H.; Kim, Y. Novel stochastic computing for energy-efficient image processors. *Electronics* **2019**, *8*, 720.
4. Gross, W.J.; Gaudet, V.C.; Milner, A. Stochastic implementation of LDPC decoders. In Proceedings of the 2005 39th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA 30 October–2 November 2005; pp. 713–717.
5. Tehrani, S.S.; Naderi, A.; Kamendje, G.A.; Mannor, S.; Gross, W.J. Tracking forecast memories in stochastic decoders. In Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing, Taipei, Taiwan, 19–24 April 2009; pp. 561–564.
6. Sarkis, G.; Hemati, S.; Mannor, S.; Gross, W.J. Stochastic decoding of LDPC codes over GF (q). *IEEE Trans. Commun.* **2013**, *61*, 939–950.
7. Han, K.; Hu, J.; Chen, J.; Zhang, Z.; Lu, H. A fast converging normalization unit for stochastic computing. *IEEE Trans. Circuits Syst. II Express Briefs* **2017**, *65*, 501–505.
8. Chen, J.; Zhang, Z.; Lu, H.; Hu, J.; Sobelman, G.E. An intra-iterative interference cancellation detector for large-scale MIMO communications based on convex optimization. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2016**, *63*, 2062–2072.
9. Han, K.; Hu, J.; Chen, J.; Lu, H. A low complexity sparse code multiple access detector based on stochastic computing. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *65*, 769–782.
10. Frasser, C.F.; Roca, M.; Rossello, J.L. Optimal stochastic computing randomization. *Electronics* **2021**, *10*, 2985.
11. Kim, J.; Jeong, W.S.; Jeong, Y.; Lee, S.E. Parallel stochastic computing architecture for computationally intensive applications. *Electronics* **2023**, *12*, 1749.
12. Ardakani, A.; Leduc-Primeau, F.; Onizawa, N.; Hanyu, T.; Gross, W.J. VLSI implementation of deep neural network using integral stochastic computing. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 2688–2699.
13. Chen, Y.; Li, H. Stochastic computing using amplitude and frequency encoding. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2022**, *30*, 656–660.
14. Gross, W.J.; Gaudet, V.C. *Stochastic Computing: Techniques and Applications*; Springer: Cham, Switzerland, 2019.
15. Temenos, N.; Sotiriadis, P.P. Deterministic finite state machines for stochastic division in unipolar format. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Virtual, 12–14 October 2020; pp. 1–5.
16. Canals, V.; Morro, A.; Rosselló, J.L. Stochastic-based pattern-recognition analysis. *Pattern Recognit. Lett.* **2010**, *31*, 2353–2356.
17. Perez-Andrade, I.; Zhong, S.; Maunder, R.G.; Al-Hashimi, B.M.; Hanzo, L. Stochastic computing improves the timing-error tolerance and latency of turbo decoders: Design guidelines and tradeoffs. *IEEE Access* **2016**, *4*, 1008–1038.
18. Zhang, Y.; Zhang, X.; Song, J.; Wang, Y.; Huang, R.; Wang, R. Parallel convolutional neural network (CNN) accelerators based on stochastic computing. In Proceedings of the 2019 IEEE International Workshop on Signal Processing Systems (SiPS), Nanjing, China, 20–23 October 2019; pp. 19–24.
19. Zhu, Y.; Dai, Y.; Han, K.; Wang, J.; Hu, J. An efficient bicubic interpolation implementation for real-time image processing using hybrid computing. *J. Real-Time Image Process.* **2022**, *19*, 1211–1223.
20. Nikopour, H.; Baligh, H. Sparse code multiple access. In Proceedings of the 2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC), London, UK, 8–11 September 2013; pp. 332–336.
21. Tan, C.W.; Calderbank, A.R. Multiuser detection of Alamouti signals. *IEEE Trans. Commun.* **2009**, *57*, 2080–2089.
22. Yang, K. Non-Orthogonal Multiple Access Using Guessing Random Additive Noise Decoding Aided Macrosymbols. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2025.
23. Yang, J.; Zhang, C.; Xu, S.; You, X. Efficient stochastic detector for large-scale MIMO. In Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, 20–25 March 2016; pp. 6550–6554.
24. Chen, J.; Hu, J.; Sobelman, G.E. Stochastic iterative MIMO detection system: Algorithm and hardware design. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2017**, *62*, 1205–1214.
25. Li, M.; Ji, H.; Tan, X.; Zhang, C. Stochastic Belief Propagation-Based Iterative Detection and Decoding for MIMO Systems. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2025**, *33*, 2324–2328. [[CrossRef](#)]
26. Lv, Q.; Geng, L.; Cao, Z.; Cao, M.; Li, S.; Li, W.; Fu, G. Adaptive Sparse Softmax: An Effective and Efficient Softmax Variant. *IEEE Trans. Audio Speech Lang. Process.* **2025**, *33*, 3148–3159. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.