IEICE TRANS. FUNDAMENTALS, VOL.E107–A, NO.7 JULY 2024

PAPER

Efficient Realization of an SC Circuit with Feedback and Its Applications

Yuto ARIMURA[†a)], *Nonmember and* **Shigeru YAMASHITA**[†b)], *Senior Member*

SUMMARY Stochastic Computing (SC) allows additions and multiplications to be realized with lower power than the conventional binary operations if we admit some errors. However, for many complex functions which cannot be realized by only additions and multiplications, we do not know a generic efficient method to calculate a function by using an SC circuit; it is necessary to realize an SC circuit by using a generic method such as polynomial approximation methods for such a function, which may lose the advantage of SC. Thus, there have been many researches to consider efficient SC realization for specific functions; an efficient SC square root circuit with a feedback circuit was proposed by D. Wu et al. recently. This paper generalizes the SC square root circuit with a feedback circuit; we identify a situation when we can implement a function efficiently by an SC circuit with a feedback circuit. As examples of our generalization, we propose SC circuits to calculate the n-th root calculation and division. We also show our analysis on the accuracy of our SC circuits and the hardware costs; our results show the effectiveness of our method compared to the conventional SC designs; our framework may be able to implement a SC circuit that is better than the existing methods in terms of the hardware cost or the calculation error.

key words: *Stochastic Computing, feedback circuit, square root, n-th root, division*

1. Introduction

Stochastic Computing (hereafter, SC) was proposed in 1960 [1] as a low power computing paradigm for applications which can allow some calculation errors such as machine learning and pattern recognition [2]–[9].

SC uses a Stochastic Number (hereafter, SN) which represents the ratio of 1's in a bit string; it can perform the multiplication of two SNs by using only a single AND gate. Also we can perform an addition of two SNs by using only a single multiplexer (MUX). Thus, complex arithmetic polynomial functions consisting of only additions and multiplications can be realized with a very small hardware overhead which leads to low power computations.

However, if we want to realize a complex function which cannot be realized by only multiplications and additions (e.g., division and square root operations), we need to approximate the target function by a Bernstein polynomial [10]. We can realize any function by this approximation method [2], but this method may need large hardware overhead and increases calculation errors for some cases. There have been proposed another polynomial approximation method called the Horner's method [11] which uses the Taylor expansions. The polynomial approximation method based on the Horner's method may be better than the one based on Bernstein polynomials for many cases, but still suffers from some hardware overhead and calculation errors.

Therefore, a specific efficient realization of a specific SC calculation have been studied intensively in the research community of SC. Among such researches, an efficient realization of SC calculation for square root has been proposed [12] recently. The method proposes an SC square root circuit called *Bit-Inserting Square Root (hereafter, BISQRT)* [12]; the idea of BISQRT is to increase the number of 1's in the input bit-stream by using a feedback circuit.

Our contribution.
In this paper we generalize the idea of BISQRT so that it can be applied to many functions other than a square root function. More precisely, the contributions of this paper can be summarized as follows.

- We identify a situation when we can implement a function efficiently by an SC circuit with a feedback circuit.
- We show two such situations, and show how to realize SC circuits to calculate the n-th root calculation and the division.
- We show the comparison of the calculation errors and the hardware costs between our proposed SC cubic root circuit and the ones by the conventional polynomial approximation method based on the Horner's method [11]; our proposed circuit can be realized with smaller hardware cost that the conventional method.
- We also show the comparison of the calculation errors and the hardware costs between our proposed SC division circuit and the conventional Correlated Division (CORDIV) [13] circuit to show the effectiveness of our method.

This paper is organized as follows. After providing necessary information for SC in Sect. 2, we propose our idea to generalize BISQRT in Sect. 3; we identify a situation when a function can be realized efficiently by an SC circuit with a feedback circuit. Then, Sect. 4 shows our analysis on the accuracy and the hardware cost of SC cubic root circuits by our proposed framework and by the conventional polynomial approximation method based on the Horner's method. We also show the same analysis for SC division circuits where we

Manuscript received February 18, 2023.
Manuscript revised July 2, 2023.
Manuscript publicized October 26, 2023.

[†]The authors are with College of Information Science and Engineering, Ritsumeikan University, Kusatsu-shi, 525-8577 Japan.
a) E-mail: alain@ngc.is.ritsumei.ac.jp
b) E-mail: ger@cs.ritsumei.ac.jp

DOI: 10.1587/transfun.2023EAP1019

compare our proposed circuit with the conventional SC division circuit called Correlated Division [13] by Te-Hsuan et al. and In-Stream Correlation-based Division [12] by D. Wu et al. Finally, Sect. 5 concludes the paper with our future work.

2. Stochastic Computing

In Stochastic Computing (SC), the probability of “1” in the bit string is treated as a value. For example, a bit string “11101011” represents 3/4 because there are six “1” in the 8 bit-length. Thus, the probability of the presence of 1’s in a bit string is called “Stochastic Number (SN).” This means that the values of two SNs can be equal even though the two bit strings are different. For example, both “11001001” and “10101001” represent 1/2. In addition, the accuracy of the values increases with the length of the bit string.

SN can be generated by comparing a binary constant with a random number generated by the Linear Feedback Shift Register (hereafter, LFSR), as shown in Fig. 1. The SN generator as shown in Fig. 1 is called a Stochastic Number Generator (hereafter, SNG). Since SN takes as its value the probability of the presence of 1’s in a bit string, the range of representable values is $[0, 1]$. Therefore, appropriate scaling of the outputs is necessary when dealing with values outside this range. For example, if the maximum input number is 100, we generate all the input SNs scaled by a factor of 1/100, and we scale the result of the SC calculation by a factor of 100 to get the correct result.

2.1 SC Calculation

In SC, some operations can be performed with very simple logic gates. For example, multiplication can be realized with a single AND gate as shown in Fig. 2. Addition can be realized with a single MUX, as shown in Fig. 3. However, the output is scaled by 1/2. Also, if the two inputs has negative correlation, saturation addition without scaling can be realized with a single OR gate as shown in Fig. 4.

2.2 The Effect of Correlation Between Input SNs

In SC calculation, the calculated result may have errors if there are correlations between the input SNs. For example, if the same bit string is given as the two inputs for the multiplication with a single AND gate, the output will be the same bit string as the input as shown in Fig. 5. Thus, multiplying with a single AND gate may not produce correct output if there is a strong correlation between the inputs. Such arithmetic errors due to correlation also exist in other SC calculations [14], [15].

On the other hand, there are operations that use correlations [14]. For example, when the correlation between the inputs is maximal, the minimum value of the two inputs can be found by a single AND gate. Therefore, in SC calculation, in order to perform the intended calculation, it may be necessary to manipulate the correlations [16], [17].

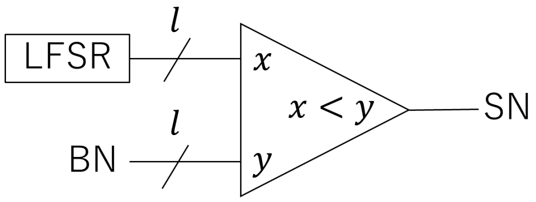

Fig. 1 A Stochastic Number Generator.

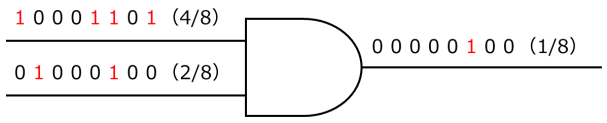

Fig. 2 Multiplication with a single AND gate.

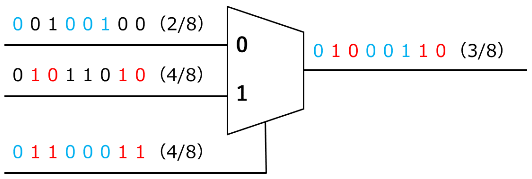

Fig. 3 Addition with a single MUX.

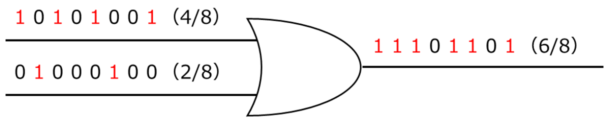

Fig. 4 Addition with a single OR gate.

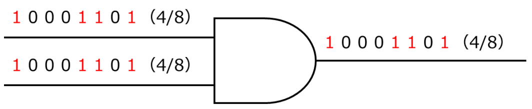

Fig. 5 Example when the same bit string as inputs give to a single AND gate.

3. Realization of SC Calculation with Feedback

In this paper we propose an efficient method to realize SC complex functions by using feedback circuits. Our method is a generalization of the method proposed by D. Wu et al. for realizing SC square root circuits using so called *Bit-Inserting Square Root (BISQRT)* [12]; we generalize the idea of BISQRT so that it can be applied to many functions other than a square root function. More precisely, we identify a situation when we can implement a function efficiently by an SC circuit with a feedback circuit. Then, we show two such situation.

3.1 Realization of SC Square Root Circuits Based on BISQRT

The output value of the square root calculation should exceed the input value. Thus BISQRT tries to increase the number

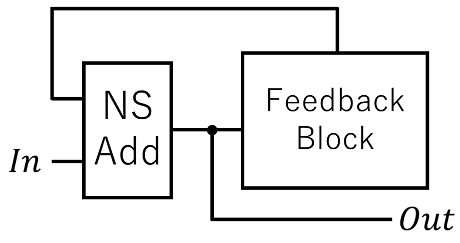

Fig. 6 The circuit of BISQRT.

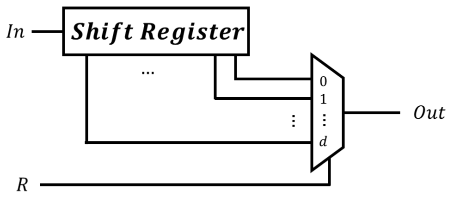

Fig. 7 The circuit that produces less correlated SNs by changing the order of 1's in a SN.

of 1's in the input SN so that the output value becomes the square root function. The circuit of BISQRT proposed in [12] can be shown in Fig. 6. This circuit uses Non-Scale Addition (hereafter, NSAdd) to insert 1's into the input only when the input is 0, depending on the output of Feedback Block.

We adjust the output of this Feedback Block such that the output of the entire circuit becomes the square root of the input value. Let P_{In}, P_{Out} and $P_{Feedback}$ be the probabilities that the input In, the output Out and the output of the Feedback Block in Fig. 6 become 1, respectively. Then P_{Out} can be expressed as Eq. (1). Here we want P_{Out} to be the square root of P_{In}, which means that $P_{In} = P^2_{Out}$. Thus, we replace P_{In} with P^2_{Out} to get Eq. (2). By solving Eq. (2), we obtain $P_{Feedback}$ as Eq. (3). Thus, we can realize an SC square root circuit by replacing the Feedback Block in Fig. 6 with a circuit that produces $P_{Feedback}$ as Eq. (3). Indeed, it is verified in [12] that the circuit as shown in Fig. 6 can realize the SC square root function.

$$P_{Out} = P_{In} + P_{Feedback} \tag{1}$$
$$= P^2_{Out} + P_{Feedback} \tag{2}$$
$$P_{Feedback} = P_{Out} \times (1 - P_{Out}) \tag{3}$$

3.2 Generalization of BISQRT

In this paper, we generalize the framework used in the above-mentioned BISQRT; we consider what kinds of target functions can be realized easily by this framework in the following.

Let P_x, P_{Out} and $P_{Feedback}$ be the probabilities that the input In, the output Out and the output of the Feedback Block in Fig. 6 become 1, respectively. Then P_{Out} can be expressed as Eq. (4). By transforming Eq. (4), we obtain $P_{Feedback}$ as Eq. (5).

Similar to the case of BISQRT, we have Eq. (1) which can be transformed into Eq. (4).

$$P_{Feedback} = P_{Out} - P_x \tag{4}$$
$$P_{Feedback} = P_{Out} \times (1 - \frac{P_x}{P_{Out}}) \tag{5}$$

Then, by rewriting Eq. (4) to Eq. (5), we can observe the following:

Our Observation.

We can easily realize a multiplication of two SNs by a single AND gate, and also $(1 - X)$ can be realized by a single NOT gate for an SN X. Thus, from Eq. (5), our conclusion is as follows: If $\frac{P_x}{P_{Out}}$ can be easily realized by an SC circuit, we can design a low-cost SC circuit to calculate P_{Out}.

Thus, our new proposal is to use an SC circuit with feedback if $\frac{P_x}{P_{Out}}$ can be easily realized by an SC circuit. We will explain how this framework is used in the following section.

3.3 How to Use Our Proposed Framework

In the following, we explain how to use our proposed framework by using two examples; one is n-th root calculation ($n = 3, 4, 5 \ldots$), and the other is division.

3.3.1 SC n-th Root Circuit Based on the Proposed Framework

When the target function $F(x) = x^{\frac{1}{n}}$, we obtain Eq. (6) from Eq. (5). By transforming Eq. (6) to Eq. (7), we can observe that $(x^{1/n})^{n-1}$ can be easily realized from $P_{Out}(= x^{1/n})$ because multiplication operations are easy in SC.

When we perform a multiplication operation between two SNs, the error becomes large if the two SNs are highly correlated. Therefore, we use a depth-d shift register (hereafter, SR), MUX and random numbers (R) to change the order of 1's in a SN to produce less correlated SNs as shown in Fig. 7. In the following figures, this circuit is represented by a box labeled with "SR." Note that the input R is omitted. Indeed we can improve the accuracy of the SC multiplication by using this circuit [12]. Thus, the circuit can be used to generate P^{n-1}_{Out} as shown in Fig. 8. In the following figures, this circuit is represented by a box labeled with "P^{n-1}_{Out}." In conclusion, we can realize $P_{Out}(= x^{\frac{1}{n}})$ by replacing the Feedback Block of Fig. 6 with the circuit as shown in Fig. 9.

$$P_{Feedback} = x^{\frac{1}{n}}(1 - \frac{x}{x^{\frac{1}{n}}}) \tag{6}$$
$$= x^{\frac{1}{n}}(1 - (x^{\frac{1}{n}})^{n-1}) \tag{7}$$

3.3.2 SC Division Circuit Based on the Proposed Framework

When the target function $F(x) = x/k$, we obtain Eq. (8) from Eq. (5). Let n be the precision level of the input SN, where k is a value in the range $(max(\frac{1}{n}, x) \leq k \leq 1)$. This is because the value of an SN should be between 0 and 1.

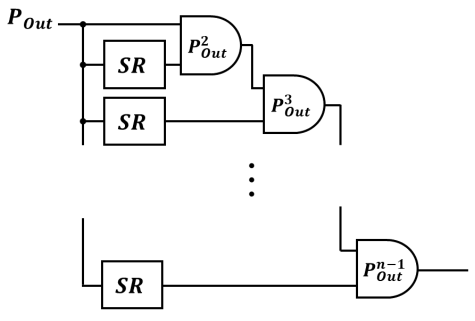

Fig. 8 The circuit that produces P^{n-1}_{Out}.

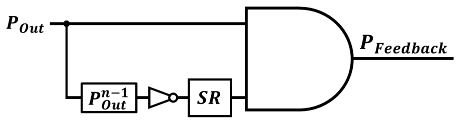

Fig. 9 Feedback Block for SC n-th root.

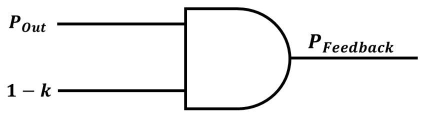

Fig. 10 The Feedback Block for SC division.

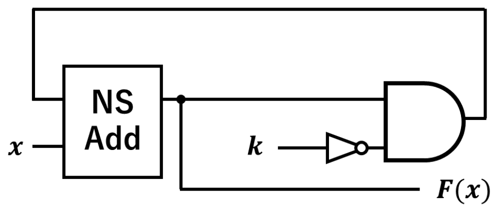

Fig. 11 The proposed SC division circuit.

Then, by transforming Eq. (8) to Eq. (9), we can observe that $P_{Out}(= x/k)$ is realized by replacing the Feedback Block with the circuit as shown in Fig. 10. Therefore, the division circuit based on the proposed framework can be shown in Fig. 11.

$$P_{Feedback} = \frac{x}{k}\left(1 - \frac{x}{\frac{x}{k}}\right) \tag{8}$$

$$= \frac{x}{k}(1 - k) \tag{9}$$

4. Experimental Results

We have performed an analysis on the accuracy of the SC cubic root circuit by our proposed framework and that by the conventional polynomial approximation method based on the Horner's method [11]. Also, we have performed the same analysis for divisor circuits; we compared our SC division circuit with *Correlated Division* (hereafter, CORDIV) [13] circuit by Te-Hsuan et al. and *In-Stream Correlation-based Division* (hereafter, ISCBDIV) [12] by D. Wu et al. The results are shown in the following.

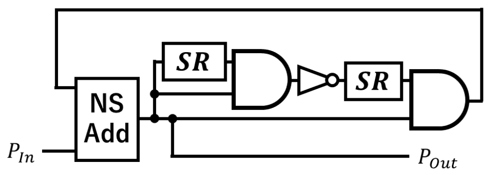

Fig. 12 An SC cubic root circuit by our proposed framework.

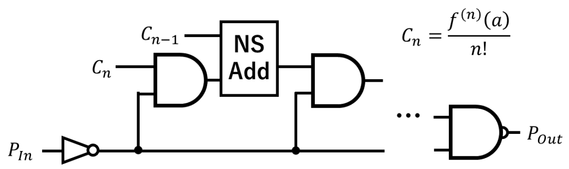

Fig. 13 An SC cubic root circuit by the Taylor expansion at $a = 1$.

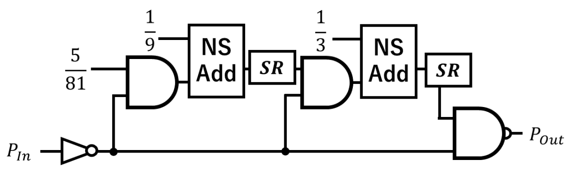

Fig. 14 An SC cubic root circuit by the Taylor expansion when $n = 3$.

4.1 SC Cubic Root Circuits by Our Proposed Framework

We show the SC cubic root circuit generated by our proposed framework in Fig. 12. The circuit is the power-root circuits proposed in Sect. 3 when $n = 3$.

In our implementation, we need to calculate P^2_{Out}. In SC, we can multiply two SNs by only a single AND gate. However, if the two SNs have some correlation, the result of the multiplication may have large errors. So when the two SNs are correlated partially, we use an SR as mentioned in Sect. 3. Thus we use an SR before we calculate P^2_{Out} as shown in Fig. 12, where a box labeled with "SR" before an AND gate is an SR. In addition, we adopt the design proposed in [18] for non-scaled additions to achieve high accuracy.

4.2 SC Cubic Root Circuits by a Polynomial Approximation Method

Equation (10) is the Taylor series of a general function. By using this equation, we obtain the Taylor expansion of $f(x) = \sqrt[3]{x}$ as follows. We cannot perform the Taylor expansion of the function at $a = 0$, because when $n \geq 1$, the denominators in Eq. (11) become 0. Thus, we use Eq. (12) to realize SC cubic root circuits. When $a = 1$, to perform the Taylor expansion, we obtain Eq. (12). Furthermore, by deforming this equation to Eq. (13), we obtain Eq. (14). This formula can be realized very efficiently in SC by using only AND, NOT gates and NSAdds.

From the above Taylor expansion, we generate an SC cubic root circuit as shown in Fig. 13. This circuit needs $n-1$ NSAdds whose hardware cost is relatively large [18]. Thus,

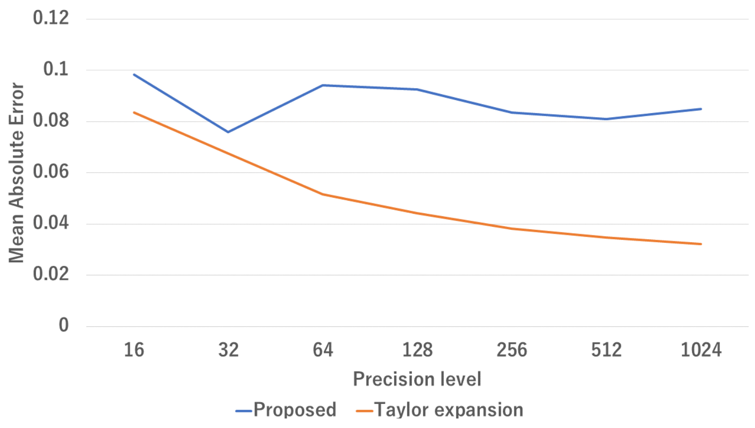

Fig. 15 MAE for each input precision about SC cubic root circuits.

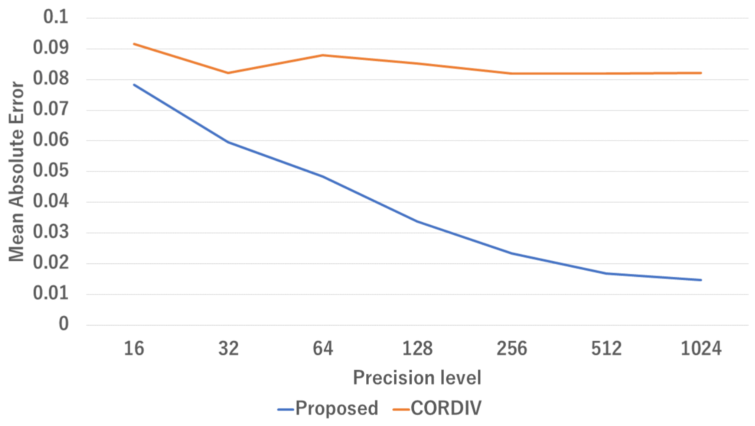

Fig. 16 MAE for each input precision about SC division circuits.

if n increases, the hardware cost of this circuit becomes very large. Therefore, for a fair comparison in terms of the hardware overhead, we consider the case when $n = 3$; we realize the circuit by using Eq. (14) which is the Taylor expansion of $\sqrt[3]{x}$ when $a = 1$ and $n = 3$. Accordingly, Fig. 14 shows the SC cubic root circuit by the conventional polynomial approximation method for our comparison.

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{10}$$

$$f(x) = a|_{a=0} + \left.\frac{x}{3a^{\frac{2}{3}}}\right|_{a=0} - \left.\frac{x^2}{9a^{\frac{5}{3}}}\right|_{a=0} \cdots \tag{11}$$

$$f(x) = 1 + \frac{x-1}{3} - \frac{(x-1)^2}{9} + \frac{5(x-1)^3}{81} \cdots \tag{12}$$

$$\begin{aligned} f(x) &= 1 + \frac{x-1}{3} - \frac{(x-1)^2}{9} + \frac{5(x-1)^3}{81} \cdots \\ &= 1 + (x-1)\Big(\frac{1}{3} - (x-1)\Big(\frac{1}{9} - (x-1)\Big(\frac{5}{81} \cdots \end{aligned} \tag{13}$$

$$f(x) = 1 - (1-x)\Big(\frac{1}{3} + (1-x)\Big(\frac{1}{9} + (1-x)\Big(\frac{5}{81} \cdots \tag{14}$$

$$E_{cr} = \frac{1}{N-1} \sum_{n=1}^{N-1} |P_n - T_n| \tag{15}$$

$$E_{div} = \frac{2}{(N-2)(N-1)} \sum_{i=2}^{N-1} \sum_{j=1}^{i-1} \left|P_{i,j} - T_{i,j}\right| \tag{16}$$

4.3 Experimental Results

4.3.1 Accuracy Comparison

We implemented a simulation program to compare the arithmetic errors and the mean absolute error (hereafter, MAE) between the circuits by our proposed framework and that by the conventional Taylor expansion-based method or the existing division circuit. First, in order to examine the change in MAE by the variation of the precision level of SNs, we consider the precision levels of SNs are N bits ($N = 8, 16, 32, \cdots, 1024$) in this order. For each precision level N, we calculated the error values of the function from $x = 1/N$ to $x = (N-1)/N$. For the division circuit, for each x, we calculated the error value for all k satisfying ($x \le k \le 1$).

In our simulation, we consider the behavior of NSAdd based on [18]. The simulation results should depend on

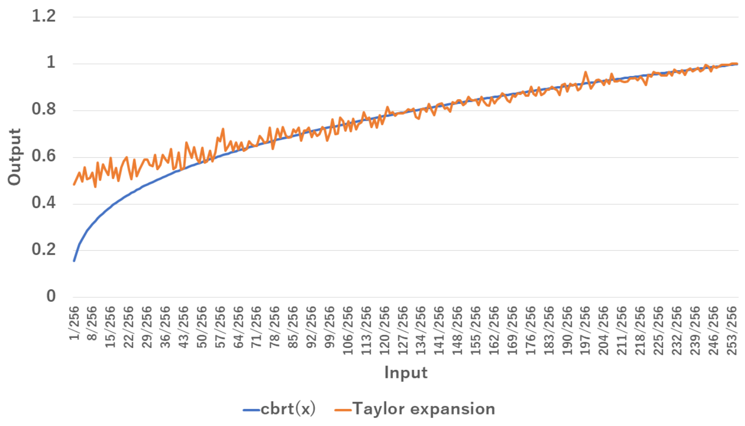

Fig. 17 Comparison between the output of SC cubic root by Taylor expansion and the correct value.

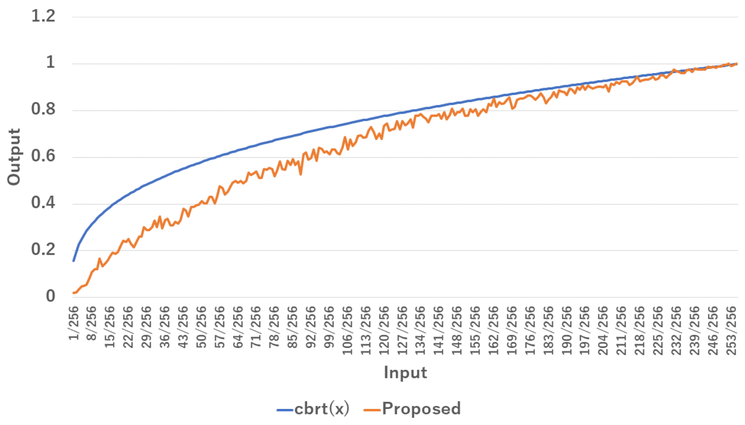

Fig. 18 Comparison between the output of SC cubic root by proposed method and the correct value.

the random numbers used for the LFSRs, thus we calculate MAE at each precision level by using different 100 random numbers for the LFSRs.

When we denote the output value and the correct value as P and T, we calculate the average error values by Eqs. (15) and (16) for the SC cubic root circuits and the division circuits, respectively. Note that the number of inputs for the SC cubic root circuits is $N-1$, and so we calculate the average of $N-1$ combinations of inputs by Eq. (15). For the division circuits, we have two inputs (divisor and dividend) and the number of combinations to be considered is $\frac{(N-2)(N-1)}{2}$ because we consider the cases where the dividend is less than the divisor which is $\frac{1}{N}$ to $\frac{N-1}{N}$. Thus Eq. (16) is obtained by the average of $\frac{(N-2)(N-1)}{2}$ combinations. Then, we try different 100 random numbers for the LFSRs in the above simulations, and we report the average of the 100 trials in the following. We also calculate all the error values when the precision level is 256 bits for SC cubic root circuits. In our comparison, we used depth-5 SRs for shift registers.

By observing most applications of SCs in literature, it would be enough to consider the precision level is 256. Also it is pointed out that if the precision level becomes large, SC would not be energy-efficient [19]. Thus, in the following, we consider that the precision level is 256.

Table 1 Area of each module.

	Area(μm)	Area_ref
LFSR	848.33	37.57
Comparator	1248.31	55.29
NSAdd [19]	2709.5	120
Counter	1161.22	51.43
SS [17]	174.18	7.71
SR (depth = 2)	319.33	14.14

Figures 15 and 16 show how MAE of each method differs for the different precision levels. An SC cubic root circuit based on Taylor expansion has a lower MAE than the one based on our proposed framework. An SC division circuit based on our proposed framework has a lower MAE than CORDIV. Note that we did not compare our method with ISCBDIV because CORDIV has better arithmetic accuracy than ISCBDIV [12].

Figures 17 and 18 show the comparison between the correct value and the output value of each method when the input is from $x = 1/N$ to $x = (N-1)/N$ where the precision level N is 256. In each figure, the correct value of the cubic root is plotted as $cbrt(x)$.

The SC cubic root circuit based on the Taylor expansion outputs the values more than the correct values when the input is between 1/256 to 46/256. The maximum difference

Table 2 Comparison of hardware costs for SC division circuits.

	LFSR	Comparator	NSAdd [19]	Counter	SS [17]	SR (depth = 2)
CORDIV [14]	1	2	0	1	0	0
ISCBDIV [13]	0	0	0	0	1	1
Proposed method	0	0	1	0	0	0

from the correct value is 0.338603869. When the input is more than 46/256, the output value is sometimes more and sometimes less than the correct value. The SC cubic root circuit by our proposed framework does not output a value exceeding the correct value for most input values. The maximum difference from the correct value is 0.202898136.

From these results, we could observe that the error values of the SC cubit root circuit by our method is not so bad for most inputs, and also our method is even better than the Taylor expansion-based method when the input x is very small although MAE of our method is worse than that of the Taylor expansion-based method. Note that the maximum difference between the output of a circuit based on our proposed framework and the correct value is lower than that of the conventional Taylor expansion based method.

As we will show, our method is better in terms of the hardware cost than the conventional Taylor expansion based method. Thus, we would conclude that our framework can provide a new type of SC circuits which may be better than the known SC circuits in some aspects.

4.3.2 Hardware Comparison

We implemented major modules used in the SC cubic root and SC diviosion circuits based on our proposed framework and the conventional methods for the case of 256-bit SNs. Table 1 shows the hardware cost for each module measured by Synopsys Design Compiler with Rohm 0.18μm library. "Area" and "Area_ref" in the table represent the circuit area and the number of NAND2 gates, respectively.

First, by observing Figs. 12 and 14 for the SC cubic root circuit, it can be seen that the design with the Taylor expansion has one more NSAdd and one more NAND gate than our method. Thus, by observing that the cost of NSAdd is relatively large compared to other modules in Table 1, it is obvious that the circuit based on our proposed framework can be realized at a lower cost than Taylor expansion based method.

Next, let us consider the hardware costs for the SC division circuits based on CORDIV, ISCBDIV and our framework. CORDIV requires positive correlation between inputs, therefore it needs SN re-generation, and thus we need an additional LSFR and a comparator. ISCBDIV uses a *Skewed Synchronizer* (hereafter, SS) of depth-1 to ensure positive correlation between inputs. Also, to increase the arithmetic accuracy, one D Flip-Flop used in CORDIV is replaced with an SR of depth-2. In contrast, the circuit based on our proposed framework does not need the correlation between inputs, but it requires an NSAdd. In conclusion, Table 2 shows the number of major modules used in each SC division circuit. Then, considering the hardware cost of each module listed in Table 1, we can conclude that the circuit based on our proposed framework has lower hardware cost than the one based on CORDIV.

5. Conclusion

In this paper, we proposed the SC design framework to use a feedback circuit; we considered what kinds of target functions can be realized easily by using a feedback circuit. Our framework is considered as a generalization of BISQRT. Then, we proposed an SC n-th root circuit and an SC division circuit based on our proposed framework.

An SC cubic root circuit based on our proposed framework has a lower hardware area than an SC cubic root circuit by a Taylor expansion. An SC division circuit based on our proposed framework shows the lower MAE and hardware area than the one based on CORDIV. Our simulation results also show that the maximum difference between the output of a circuit based on our proposed framework and the correct value is lower than that of the conventional Taylor expansion based method. In conclusion, we can expect that our framework provide us a new type of SC circuits which are better than the known SC circuits in some aspects.

For our future work, we need to consider other target functions. In addition, it would be interesting to consider replacing NSAdd in our framework with other gates such as an AND gate or an OR gate.

References

[1] B.R. Gaines, "Stochastic computing systems," Advances in Information Systems Science, Springer, pp.37–172, 1969.

[2] A. Alaghi and J. Hayes, "Survey of stochastic computing," ACM Trans. Embed. Comput. Syst., vol.12, no.2s, pp.92:1–92:19, May 2013.

[3] A. Alaghi, C. Li, and J.P. Hayes, "Stochastic circuits for real-time image-processing applications," 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC), pp.1–6, 2013.

[4] K. Kim, J. Kim, J. Yu, J. Seo, J. Lee, and K. Choi, "Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks," 2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp.1–6, 2016.

[5] P. Li and D.J. Lilja, "Using stochastic computing to implement digital image processing algorithms," 2011 IEEE 29th International Conference on Computer Design (ICCD), pp.154–161, Oct. 2011.

[6] Z. Wang, N. Saraf, K. Bazargan, and A. Scheel, "Randomness meets feedback: Stochastic implementation of logistic map dynamical system," Proc. 52nd Annual Design Automation Conference, ACM, p.7, 2015.

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

[8] B. Brown and H. Card, "Stochastic neural computation. I. computational elements," IEEE Trans. Comput., vol.50, no.9, pp.891–905, 2001.

[9] V. Gaudet, "Iterative decoding using stochastic computation," Electron. Lett., vol.39, no.3, pp.299–301, Feb. 2003.
[10] G.G. Lorentz, Bernstein Polynomials, 2nd ed., Chelsea Pub. Co., New York, N.Y., 1986.
[11] W.G. Horner, "A new method of solving numerical equations of all orders, by continuous approximation," Philosophical Transactions of the Royal Society, vol.2, pp.117–117, 1819.
[12] D. Wu, R. Yin, and J.S. Miguel, "In-stream correlation-based division and bit-inserting square root in stochastic computing," IEEE Des. Test, vol.38, no.6, pp.53–59, 2021.
[13] T.-H. Chen and J.P. Hayes, "Design of division circuits for stochastic computing," 2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp.116–121, 2016.
[14] A. Alaghi and J.P. Hayes, "Exploiting correlation in stochastic circuit design," 2013 IEEE 31st International Conference on Computer Design (ICCD), pp.39–46, 2013.
[15] M. Parhi, M.D. Riedel, and K.K. Parhi, "Effect of bit-level correlation in stochastic computing," 2015 IEEE International Conference on Digital Signal Processing (DSP), pp.463–467, 2015.
[16] V.T. Lee, A. Alaghi, and L. Ceze, "Correlation manipulating circuits for stochastic computing," 2018 Design, Automation & Test in Europe Conference Exhibition (DATE), pp.1417–1422, 2018.
[17] R. Ishikawa, M. Tawada, M. Yanagisawa, and N. Togawa, "An effective stochastic number duplicator and its evaluations using composite arithmetic circuits," 2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS), pp.53–56, 2018.
[18] D. Wu, J. Li, R. Yin, H. Hsiao, Y. Kim, and J.S. Miguel, "UGEMM: Unary computing architecture for GEMM applications," 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), pp.377–390, 2020.
[19] V.T. Lee, A. Alaghi, R. Pamula, V.S. Sathe, L. Ceze, and M. Oskin, "Architecture considerations for stochastic computing accelerators," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.37, no.11, pp.2277–2289, 2018.

Yuto Arimura the B.E. degree in Information Science and Engineering from Ritsumei University in 2022. He is currently a graduate student of Graduate School of Information Science and Engeneering. Ritsumeikan University, Shiga, Japan. His research interests include how to use stochastic numbers for approximate computing.

Shigeru Yamashita is a professor at College of Information Science and Engineering, Ritsumeikan University. He received his B.E., M.E. and Ph.D. degrees in Information Science from Kyoto University, Kyoto, Japan, in 1993, 1995 and 2001, respectively. His research interests include new types of computation and logic synthesis for them. He received the 2000 IEEE Circuits and Systems Society Transactions on Computer-Aided Design of Integrated Circuits and Systems Best Paper Award, SASIMI 2010 Best Paper Award, 2010 IPSJ Yamashita SIG Research Award, and 2010 Marubun Academic Achievement Award of the Marubun Research Promotion Foundation. He is a senior member of IEEE, and a member of IPSJ.