Minimizing Error of Stochastic Computation through Linear Transformation

Yi Wu, Chen Wang, and Weikang Qian
University of Michigan-Shanghai Jiao Tong University Joint Institute
Shanghai Jiao Tong University, Shanghai, China
Email: {eejessie, wangchen_2011, qianwk}@sjtu.edu.cn

ABSTRACT

Stochastic computation is an unconventional computational paradigm that uses ordinary digital circuits to operate on stochastic bit streams, where signal value is encoded as the probability of ones in a stream. It is highly tolerant of soft errors and enables complex arithmetic operations to be implemented with simple circuitry. Prior research has proposed a method to synthesize stochastic computing circuits to implement arbitrary arithmetic functions by approximating them via Bernstein polynomials. However, for some functions, the method cannot find Bernstein polynomials that approximate them closely enough, thus causing a large computation error. In this work, we explore linear transformation on a target function to reduce the approximation error. We propose a method to find the optimal linear transformation parameters to minimize the overall error of the stochastic implementation. Experimental results demonstrated the effectiveness of our method in reducing the computation error and the circuit area.

Categories and Subject Descriptors

B.6.1 [**Logic Design**]: Design Styles

General Terms

Design, Performance

Keywords

stochastic computing; stochastic circuit; approximation error; stochastic variation

1. INTRODUCTION

Stochastic computation is a method to explicitly use randomness to perform computation [1]. Like conventional digital computation, stochastic computation uses digital circuits to process information encoded by zeros and ones. However, stochastic circuits take random bit streams as their inputs and outputs. Each bit stream encodes a value equal to the ratio of ones in that stream. For example, a stream of 8 random bits $1, 0, 1, 0, 0, 1, 0, 0$ encodes the value $3/8$.

GLSVLSI'15, May 20–22, 2015, Pittsburgh, PA, USA.

http://dx.doi.org/10.1145/2742060.2743761.

One advantage of stochastic computation is that many complex arithmetic operations can be realized using very simple circuitry. As shown in Fig. 1, an AND gate can implement multiplication: given that the two input stochastic bit streams are independent, the probability of ones in the output bit stream equals the product of the probabilities of ones in the input streams. Previous works have also introduced various simple circuits to implement other functions such as addition, division, and square root [2, 3].

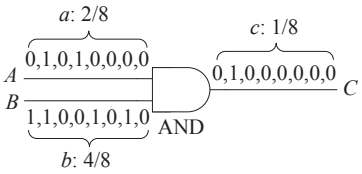

Figure 1: Multiplication on stochastic bit streams with an AND gate. Here the inputs are $2/8$ and $4/8$. The output is $2/8 \times 4/8 = 1/8$, as expected.

Besides area efficiency, stochastic computation is also highly tolerant of soft errors, because a single bit flip occurring anywhere in the stream changes the value only slightly. The strong fault tolerance of stochastic computation was visually demonstrated by several image processing applications [4, 5].

To apply stochastic computation to a wide variety of applications, an automatic way to synthesize stochastic computing circuits is required. Recently, several synthesis approaches have been proposed [4, 6, 7]. One method proposed by Qian et al. is to first approximate an arbitrary function via a Bernstein polynomial [8] and then implement that Bernstein polynomial through a specific stochastic computing circuit [4].

However, we observe that for some functions, we cannot find Bernstein polynomials of a low degree that approximate them closely enough, hence causing a large approximation error. In this case, Bernstein polynomial of a higher degree is needed to reach a close approximation, which increases area cost.

In this work, we propose a linear transformation technique to reduce the approximation error. We perform a proper linear transformation on the original function to obtain a function that can be approximated very closely by a Bernstein polynomial of a low degree. We then synthesize a circuit to compute the transformed function. The value of the original function can be easily obtained through an inverse linear transformation, which is achieved by a simple modification to the original design.

Although the linear transformation technique could reduce the approximation error, it will amplify the error due to random fluctuation, which is another major error component of stochastic computation. In order to minimize the overall er-

ror of stochastic computation, we further propose a method to find the optimal linear transformation parameters. The proposed technique dramatically reduces the overall error.

The remainder of the paper is organized as follows. Section 2 introduces the background on the Bernstein polynomial based approach to synthesize stochastic computing circuit. Section 3 describes the linear transformation technique that decreases the approximation error. Section 4 proposes a method to determine the optimal linear transformation parameters to minimize the overall error of stochastic computation. Section 5 shows the experimental results. Conclusions are drawn in Section 6.

2. BACKGROUND

2.1 Bernstein Polynomial Based Synthesis Method

The method proposed in [4] first approximates the target function by a Bernstein polynomial [8] and then implements the Bernstein polynomial through a specific stochastic computing circuit. A *Bernstein polynomial* of degree n is of the form

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

where each real number b_i is a constant, called *Bernstein coefficient*, and each $B_{i,n}(x)$ $(i = 0, 1, 2, \cdots, n)$ is a *Bernstein basis polynomial* of the form

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

A Bernstein polynomial *with all the coefficients in the unit interval* can be implemented by a stochastic computing circuit shown in Fig. 2. Since it is the core of the stochastic computing system, to be discussed later, we call the circuit *stochastic computing core*. The core consists of an adder and a multiplexer. An analysis in [4] showed that the probability of the output Y to be one is in the form of a Bernstein polynomial of degree n. As shown in the figure, the Bernstein coefficients b_i's are probability values. Therefore, the Bernstein polynomial that can be realized by the stochastic computing core must have all the coefficients in the unit interval, i.e., $0 \le b_i \le 1$ for all $i = 0, \ldots, n$.

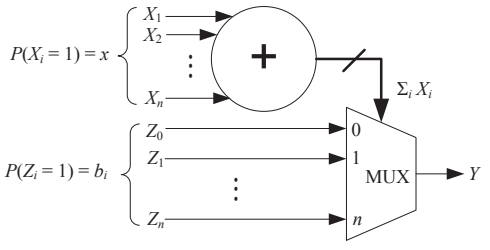

Figure 2: The stochastic computing core that implements a Bernstein polynomial $B_n(x) = \sum_{i=0}^{n} b_i B_{i,n}(x)$ with all the coefficients in the unit interval.

Given an arbitrary target function f, a Bernstein polynomial closest to f with all the coefficients in the unit interval is obtained by solving the following optimization problem on $b_0, \ldots, b_n$:

$$\begin{aligned} &\text{minimize} \int_0^1 \left(f(x) - \sum_{i=0}^{n} b_i B_{i,n}(x)\right)^2 \mathrm{d}x \\ &\text{subject to } 0 \le b_0, \ldots, b_n \le 1. \end{aligned} \tag{1}$$

The computation of the target function f is realized by implementing the approximate Bernstein polynomial using the stochastic computing core.

2.2 Error Components of Stochastic Computation

The result of stochastic computation based on the Bernstein approximation is affected by two major error components — the approximation error and the error due to random fluctuation [4]. Since the stochastic computing core implements a Bernstein polynomial $B_n(x)$ which approximates the target function $f(x)$, we have the approximation error as $e_1 = |B_n(x) - f(x)|$.

The output of the stochastic computing core is a stochastic bit stream $(Y_1, \ldots, Y_N)$ of length N, where each Y_i is a random bit having probability $B_n(x)$ of being one. For this encoding mechanism, the final output of the stochastic computing core is

$$S = \frac{1}{N} \sum_{i=1}^{N} Y_i, \tag{2}$$

which is a binomial random variable taking values from the set $\{0, \frac{1}{N}, \ldots, \frac{N-1}{N}, 1\}$. Therefore, although the output probability of the stochastic computing core is $B_n(x)$, the actual value encoded by the output bit stream is random and may not be equal to $B_n(x)$. The difference is $e_2 = |S - B_n(x)|$, which is an error due to random fluctuation.

The expectation of S, denoted as $E[S]$, and the variance of S , denoted as $Var(S)$, can be calculated as

$$E[S] = B_n(x), \quad Var[S] = \frac{B_n(x)(1 - B_n(x))}{N}. \tag{3}$$

Since $Var[S] = E[(S - E[S])^2] = E[(S - B_n(x))^2]$, we have

$$E[e_2^2] = \frac{B_n(x)(1 - B_n(x))}{N}.$$

The overall error of stochastic computation is $e = |S - f(x)|$. It is bounded by the sum of the approximation error and the error due to random fluctuation:

$$e \le |S - B_n(x)| + |B_n(x) - f(x)| = e_1 + e_2.$$

2.3 Stochastic Computing System

Fig. 3 shows a stochastic computing system [4], which consists of the stochastic computing core and the input/output interface circuits. The circuit shown in the figure implements a Bernstein polynomial of degree 3. The input interface consists of uniform random number generators (URNGs) and comparators. The URNGs produce independent random numbers, which are fed into different comparators. As shown in the figure, the inputs to the adder are independent stochastic bit streams with the same probability to have a one. The multiplexer selects one of the numbers $b_0, \ldots, b_n$ represented in binary radix form. The output of the multiplexer is then compared with a random number to generate the output stochastic bit stream Y. Finally, a counter converts a stochastic bit stream into a binary radix number.

3. REDUCING THE APPROXIMATION ERROR THROUGH LINEAR TRANSFORMATION

The approach proposed in [4] requires the Bernstein polynomial to have all of its coefficients in the unit interval. Given this constraint, it is impossible to find a low degree Bernstein polynomial to approximate some target functions closely. For example, consider a target function $f(x) = 0.5\sin(8x) + 0.5$. If we set the degree to 6, by solving the

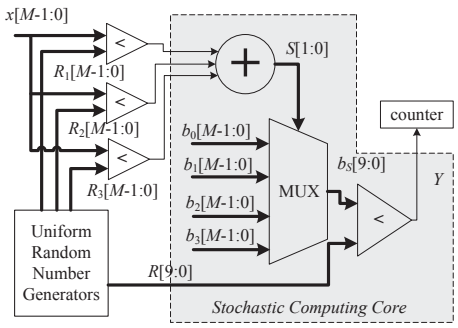

Figure 3: The stochastic computing system.

optimization problem in (1), we obtain a Bernstein polynomial with coefficients as $[b_0, b_1, b_2, b_3, b_4, b_5, b_6] = [0.735, 1, 0.832, 0, 0, 0.455, 1]$.

Fig. 4 shows the approximation effect, from which we can see that the approximation error is quite large. In the previous approach, in order to reduce the approximation error, a Bernstein polynomial with a higher degree is required. This causes a larger area for the stochastic computing system, since the area of the system increases with the degree.

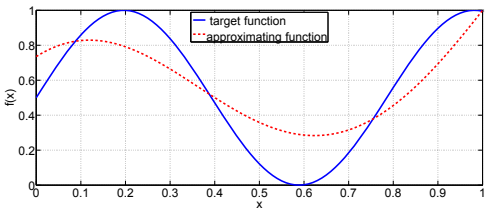

Figure 4: Approximating the target function $f(x)= 0.5\sin(8x)+0.5$ by the optimal Bernstein polynomial with all the coefficients in the unit interval.

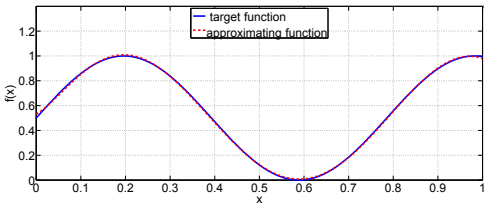

Figure 5: Approximating the target function $f(x)= 0.5\sin(8x)+0.5$ by the optimal unconstrained Bernstein polynomial.

For the same target function $f(x)= 0.5\sin(8x)+0.5$, if we remove the constraint on the Bernstein coefficients, we can obtain the coefficients of the optimal Bernstein polynomial approximation as $[b'_0, b'_1, b'_2, b'_3, b'_4, b'_5, b'_6] = [0.530,\ 0.905,\ 2.898,\ -2.415,\ -0.088,\ 1.193,\ 0.979]$. Its approximation effect is illustrated in Fig. 5. We can see that without the constraint on the Bernstein coefficients, we are able to find a very close Bernstein polynomial approximation to the target function. However, such a Bernstein polynomial cannot be implemented by the stochastic computing core. To reduce the approximation error while making the Bernstein polynomial suitable for stochastic implementation, we propose to change the target function through a linear transformation.

In our method, we use a linear transformation of the original function $g(x) = cf(x)+d$ as the target. The transformation parameters c and d are determined as follows: Suppose that the optimal *unconstrained* Bernstein polynomial that approximates $f(x)$ is $B_o(x) = \sum_{i=0}^{n} b_i B_{i,n}(x)$. We then select c and d so that for all $i = 0, \ldots, n$, $0 \le cb_i + d \le 1$.

Given $B_o(x)$ and the transformation parameters c and d, we use $B_g(x) = cB_o(x) + d$ to approximate the new target function $g(x)$. Based on the property that $\sum_{i=0}^{n} B_{i,n}(x) = 1$ [9], we have

$$B_g(x) = cB_o(x) + d = c\sum_{i=0}^{n} b_i B_{i,n}(x) + d\sum_{i=0}^{n} B_{i,n}(x)$$
$$= \sum_{i=0}^{n} (cb_i + d) B_{i,n}(x).$$

Thus, $B_g(x)$ is a Bernstein polynomial with coefficients $(cb_0 + d), \ldots, (cb_n + d)$. Based on our choice of c and d, all the coefficients of $B_g(x)$ are in the unit interval. Thus, we can implement $B_g(x)$ using the stochastic computing core. In the ideal situation where there is no error due to random fluctuation, the output S of the core is

$$S = B_g(x) \approx g(x) = cf(x) + d.$$

Thus, we can approximate the original function $f(x)$ from the output S using an inverse linear transformation $(S-d)/c$.

The above inverse linear transformation can be realized by replacing the final counter used in the stochastic computing system with an accumulator shown in Fig. 6. The signal Y shown in Fig. 3 is connected to the EN input of the accumulator.

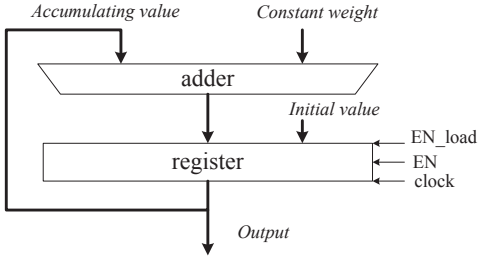

Figure 6: An accumulator for implementing the inverse linear transformation.

Assume that the length of the stochastic bit streams is N. Before the stochastic computing system starts, the signal EN_load is set to one and an initial value $-Nd/c$ is loaded into the register. For each clock cycle, if the signal Y shown in Fig. 3 is a one, then the signal EN is also a one. Then, the register will be updated with the sum of a constant weight $1/c$ and the previous value in the register. After N clock cycles, the final value in the register is

$$\frac{1}{c}\sum_{i=1}^{N} Y_i - N\frac{d}{c} \qquad (4)$$

where Y_i is the i-th bit in the output bit stream Y. From Eq. (2) and (4), the final value is $N(S-d)/c$. If we choose length N to be a power of 2, then we can obtain the stochastic computing result $(S-d)/c$ by shifting the value in the register.

Using the linear transformation method, the *approximation error* is reduced. The approximation error is defined as the difference between $f(x)$ and the ideal output of the system (i.e., output result without considering the error due to random fluctuation), which is $(B_g(x) - d)/c$. Thus, the approximation error is

$$|f(x) - (B_g(x) - d)/c| = |f(x) - B_o(x)|.$$

Since $B_o(x)$ is the optimal unconstrained Bernstein polynomial to approximate $f(x)$, it gives the smallest approximation error $|f(x) - B_n(x)|$ among all Bernstein polynomials, including the one with all the coefficients in the unit interval that best approximates $f(x)$. Thus, using the proposed linear transformation technique, we can reduce the approximation error compared to the previous approach.

4. DETERMINING THE LINEAR TRANSFORMATION PARAMETERS TO MINIMIZE THE ERROR OF STOCHASTIC COMPUTATION

As we stated in Section 2.2, the result of the stochastic computation is also subject to error due to random fluctuation, which makes the output value of the stochastic computing core, S, different from the Bernstein polynomial after linear transformation, $B_g(x)$. As a result, the actual output of the system $(S-d)/c$ is different from its ideal output $(B_g(x)-d)/c$. The error caused by random fluctuation is $|S - B_g(x)|/c$. To make this error small, c should be large. However, a large c will not help reduce the approximation error. Thus, to minimize the overall error, we need to choose a proper c. In this section, we propose a mathematical formulation to find the optimal choice of the linear transformation parameters c and d to minimize the overall error of stochastic computation.

4.1 Formulation of the Optimization Problem

To indicate that the actual output S of the stochastic computing core is related to the input x, we will rewrite S as $S(x)$. The final output of the system is $(S(x)-d)/c$. For a given x, the overall computation error is

$$e(x) = \left| \frac{S(x)-d}{c} - f(x) \right|. \tag{5}$$

To measure the average error of stochastic computation for all $x \in [0,1]$, we use the square of the L^2-norm of the function $e(x)$:

$$T = \int_0^1 e(x)^2 \, \mathrm{d}x. \tag{6}$$

Since T is a random variable, we use the expectation on T, represented as $E[T]$, as a measure of the error of the stochastic computation. Our target is to find c and d to minimize $E[T]$. From Eq. (5) and (6), we have

$$E[T] = E\left[\int_0^1 \left(\frac{S(x)-d}{c} - f(x) \right)^2 \mathrm{d}x \right].$$

Define

$$U = \frac{S(x) - B_g(x)}{c}, \; V = \frac{B_g(x)-d}{c} - f(x),$$

where $B_g(x)$ is the Bernstein polynomial implemented by the stochastic computing core. Note that U is a random variable and V is a deterministic value. Thus, we have

$$\begin{aligned} E[T] &= E\left[\int_0^1 (U+V)^2 \, \mathrm{d}x \right] \\ &= \int_0^1 \left(E[U^2] + 2VE[U] + V^2 \right) \mathrm{d}x. \end{aligned} \tag{7}$$

From Eq. (3), we have

$$E[S(x)] = B_g(x), \; E[(S(x) - B_g(x))^2] = \frac{B_g(x)(1-B_g(x))}{N}.$$

Given the above two equations, we could further simplify Eq. (7) as

$$\begin{aligned} E[T] = &\int_0^1 \frac{B_g(x)(1-B_g(x))}{c^2 N} \, \mathrm{d}x \\ &+ \int_0^1 \frac{(B_g(x) - cf(x) - d)^2}{c^2} \, \mathrm{d}x. \end{aligned} \tag{8}$$

Eq. (8) shows how the parameters c and d affect our error measurement $E[T]$. From that equation, we can see that the value of $E[T]$ also depends on the Bernstein polynomial $B_g(x)$, or more specifically, its Bernstein coefficients. However, with c and d not known yet, we do not know $B_g(x)$ neither. Thus, we will also treat the coefficients of $B_g(x)$ as unknowns. Suppose that the Bernstein polynomial $B_g(x)$ of degree n is

$$B_g(x) = \sum_{i=0}^{n} b_i B_{i,n}(x).$$

Our target is to find a set of values $b_0, b_1, \ldots, b_n, d, c$ to minimize the objective function (8). Note that $b_0, \ldots, b_n$ should satisfy the constraint that $0 \le b_i \le 1$ for all $i = 0, \ldots, n$.

4.2 Solution of the Optimization Problem

In this section, we will show how to transform the optimization problem into a standard quadratic programming problem. For this purpose, we first define the following new variables

$$b_i^* = \frac{1}{c} b_i, \text{ for } i = 0, \ldots, n \tag{9}$$

$$c^* = \frac{1}{c}, d^* = \frac{d}{c}. \tag{10}$$

Define

$$B_g^*(x) = \sum_{i=0}^{n} b_i^* B_{i,n}(x). \tag{11}$$

Then, $B_g^*(x) = \frac{1}{c} B_g(x)$. We can rewrite objective function (8) as

$$\begin{aligned} E[T] = &\int_0^1 \frac{B_g^*(x)(c^* - B_g^*(x))}{N} \, \mathrm{d}x \\ &+ \int_0^1 (B_g^*(x) - f(x) - d^*)^2 \, \mathrm{d}x. \end{aligned} \tag{12}$$

Now our target is to find a set of values $b_0^*, \ldots, b_n^*, d^*, c^*$ to minimize the objective function (12).

By substituting (11) into (12) and expanding (12), we can rewrite the objective function as

$$f_{obj}(\mathbf{z}) = \frac{1}{2} \mathbf{z}^{\mathrm{T}} \mathbf{H} \mathbf{z} + \mathbf{c}^{\mathrm{T}} \mathbf{z} + \int_0^1 f^2(x) \, \mathrm{d}x, \tag{13}$$

where

$$\mathbf{z} = [b_0^*, \ldots, b_n^*, d^*, c^*]^{\mathrm{T}}, \mathbf{c} = [c_1, c_2]^{\mathrm{T}}, \mathbf{H} = \begin{bmatrix} H_1 & H_2 \\ H_2^{\mathrm{T}} & H_3 \end{bmatrix}$$

with

$$c_1 = [-2\int_0^1 f(x)B_{0,n}(x)\,\mathrm{d}x, \ldots, -2\int_0^1 f(x)B_{n,n}(x)\,\mathrm{d}x],$$

$$c_2 = [2\int_0^1 f(x)\,\mathrm{d}x, 0]$$

$$H_1 = \frac{2(N-1)}{N}H_1',$$

$$H_1' = \begin{bmatrix} \int_0^1 B_{0,n}(x)B_{0,n}(x)dx & \ldots & \int_0^1 B_{0,n}(x)B_{n,n}(x)dx \\ \int_0^1 B_{1,n}(x)B_{0,n}(x)dx & \ldots & \int_0^1 B_{1,n}(x)B_{n,n}(x)dx \\ \vdots & \ddots & \vdots \\ \int_0^1 B_{n,n}(x)B_{0,n}(x)dx & \ldots & \int_0^1 B_{n,n}(x)B_{n,n}(x)dx \end{bmatrix},$$

$$H_2 = \begin{bmatrix} -2\int_0^1 B_{0,n}(x)dx & \frac{1}{N}\int_0^1 B_{0,n}(x)dx \\ \vdots & \vdots \\ -2\int_0^1 B_{n,n}(x)dx & \frac{1}{N}\int_0^1 B_{n,n}(x)dx \end{bmatrix},$$

$$H_3 = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}.$$

Note that **c** and **H** in Eq. (13) are all known matrices. The original set of constraints that $0 \le b_i \le 1$, for $i = 0, \ldots, n$, is now transformed into a new set of constraints on b_i^* and c^* as

$$0 \le b_i^* \le \frac{1}{c} = c^*, \text{ for } i = 0, \ldots, n. \quad (14)$$

By now, we have transformed the problem of minimizing the error of stochastic computation to finding a set of values $b_0^*, \ldots, b_n^*, c^*, d^*$ to minimize the objective function (13) subject to the constraint (14). This is a standard quadratic programming problem and can be solved using standard techniques. Once we obtain the optimal solution, we can get the optimal Bernstein coefficients $b_0, \ldots, b_n$ and the optimal linear transformation parameters c and d based on Eq. (9) and (10).

5. EXPERIMENTAL RESULTS

We performed experiments on five test functions shown in Table 1 to study the effect of the proposed linear transformation technique. For each test function, we can find an unconstrained Bernstein polynomial of degree 6 to approximate it very closely. However, for the first four functions, if we approximate them by Bernstein polynomials with coefficients in the unit interval, we need polynomials of degree at least 12 to get a close approximation. For `Test5`, we can still find a degree 6 Bernstein polynomial with all the coefficients in the unit interval to approximate it closely. `Test5` was used here to show that our method takes the previous method as a special case. For all the experiments, we fixed the length of the stochastic bit streams N as 1024.

Table 1: Five test functions used in our experiments.

Test1	$0.5\sin(8x) + 0.5$
Test2	$1 - 4(x - 0.5)^2$
Test3	$4x^2 \log(x + 0.1) + 0.53$
Test4	$59.2x^4 - 118.7x^3 + 74.9x^2 - 15.4x + 1$
Test5	e^{-3x}

5.1 Error of Stochastic Computation

In this section, we studied the computation error of the system designed using our method. We used the first four test functions as the targets. Since an unconstrained Bernstein polynomial of degree 6 can approximate each test function very closely, we chose the degree of $B_g(x)$ as 6. For each test function, we solved the quadratic programming problem proposed in Section 4.2 and obtained the optimal choice of $B_g(x)$ and the linear transformation parameters c and d. The parameters c and d for the four test functions are listed in Table 2.

Table 2: Linear transformation parameters c and d for the first four test functions.

	Test1	Test2	Test3	Test4
c	0.405	0.893	0.961	0.308
d	0.379	0	0.039	0.538

With these design parameters, we simulated our stochastic computing system to obtain the computation error. For each function, we obtained computation errors for 101 inputs $x = 0, 0.01, \ldots, 1$. For each function and each input x, we simulated the stochastic computing system 100 times. We then averaged the errors for these 100 simulations. Finally, for each function, we averaged the errors over all input points.

For comparison purpose, we also simulated the system designed using the previous method [4]. The system was designed based on a Bernstein polynomial of degree 6, but with all the coefficients in the unit interval. We obtained its computation error in the same way as above.

The errors of the two methods for each test function are listed in Table 3. We can see that when the degrees of the Bernstein polynomials implemented by the two systems are the same, the proposed method reduces the computation error significantly in comparison with the previous method.

Table 3: Average error of the proposed method and the previous method [4].

	Test1	Test2	Test3	Test4
proposed method	0.0396	0.0128	0.0103	0.0422
method in [4]	0.1539	0.0354	0.0251	0.1959

5.2 Hardware Cost

In this section, we studied the hardware cost of the proposed stochastic computing system. Still, we used the first four test functions as the targets.

The previous stochastic computing system is shown in Fig. 3. The proposed system is similar, but with the final counter replaced by the accumulator shown in Fig. 6. We used linear feedback shift registers (LFSR) as URNGs. The precision M of the binary radix numbers was chosen as 10. In order to count the number of ones in a stream of 1024 bits, the counter in the previous system has 10 bits. The accumulator in our system works on binary numbers with 10 and 6 bits before and after the binary point, respectively.

We fixed the degree of the Bernstein polynomial used in the proposed system as 6. For the previous system, we manually adjusted the degree n of the Bernstein polynomial until the system produces an average error close to that of the proposed system. The degrees n we obtained are listed in the second row of Table 4.

With the above design parameters, we used the Synopsys Design Compiler [10] to synthesize the two systems and evaluate their areas. The designs were mapped to the Nangate FreePDK45 library [11]. The area of each system for each test function is listed in Table 4. We also showed the

percentage of area saving of the proposed system over the previous system. We can see that for these four test functions, when the amount of computation errors are close, the proposed method produces a system with smaller area than the previous system.

Table 4: The degree of the Bernstein polynomial required by the method [4] in order to achieve close error to our proposed method, the areas for the two systems, and the percentage of area saving of the proposed system.

		Test1	Test2	Test3	Test4
degree of Bernstein polynomial in [4]		17	14	12	17
area (μm^2)	method in [4]	1497	1266	1136	1497
	proposed method	775			
area saving (%)		48.2	38.8	31.8	48.2

The proposed system replaces the counter in the previous system with an accumulator, which has a larger area than the counter. Therefore, when the degrees of the Bernstein polynomials implemented by the two systems are the same, the proposed system has a larger hardware cost. In order to determine when our proposed design will have advantage in area over the previous design, we further used Design Compiler to synthesize the two stochastic computing systems for different degrees of the Bernstein polynomials. Fig. 7 plots the areas of the two systems versus degrees. We performed linear regression on each set of points and obtain a linear relation between the circuit area a and the degree d. The relations for the proposed method and the previous method are shown by Eq. (15) and (16), respectively.

$$a_1 = 75.9d_1 + 319.2. \quad (15)$$

$$a_2 = 75.9d_2 + 204.1. \quad (16)$$

When the area of the proposed system is smaller than that of the previous system, i.e., $a_1 \leq a_2$, we have $d_2 - d_1 \geq 1.52$. Therefore, if the degree of the Bernstein polynomial required by the previous method is larger than the degree required by the proposed method by 2, the proposed method is more area efficient.

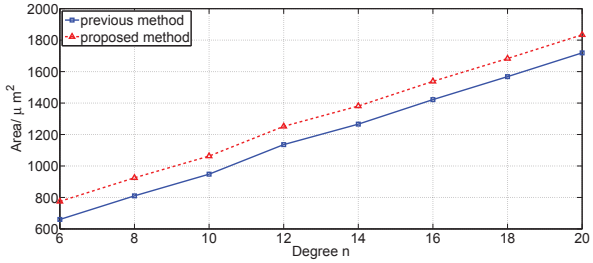

Figure 7: The areas of the proposed system and the previous system for implementing Bernstein polynomials with different degrees.

5.3 The Relation between the Proposed Method and the Previous Method [4]

In this section, we took `Test5` as the target function and implement it using the proposed technique. Note that `Test5` can be approximated closely by a degree 6 Bernstein polynomial with all the coefficients in the unit interval. In the proposed technique, we chose the degree of $B_g(x)$ as 6. After solving the quadratic programming problem shown in Section 4.2, we obtained the optimal linear transformation parameters as $c = 1$ and $d = 0$. This means that there is no linear transformation needed in order to implement `Test5`. Therefore, the system synthesized by the proposed method essentially degenerates to the system synthesized by the previous method [4]. This indicates that our method takes the previous method as a special case.

6. CONCLUSION AND DISCUSSION

In this paper, we proposed a technique using linear transformation to synthesize stochastic computing systems. We formulated an optimization problem to find the best linear transformation parameters to reduce the overall computation error. The design technique is effective to those functions which cannot be approximated closely by a Bernstein polynomial with all the coefficients in the unit interval. Compared with the previous method without linear transformation, our method could reduce the computation error as well as the circuit area.

The proposed technique performs the final inverse linear transformation through an accumulator, which converts the output stochastic bit stream into a binary radix number. Thus, the proposed system is applicable when the final result of the stochastic computation should be encoded in binary radix form.

The proposed linear transformation technique is applied in the context of the Bernstein polynomial-based synthesis approach. However, we believe that the technique can also be applied to the finite state machine (FSM)-based approach introduced in [6]. The details will be studied in our future work.

Acknowledgement

This work is supported by National Natural Science Foundation of China (NSFC) under Grant No. 61204042.

7. REFERENCES

[1] B. Gaines, "Stochastic computing systems," in *Advances in Information Systems Science*. Plenum, 1969, vol. 2, ch. 2, pp. 37–172.

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

[3] S. Toral, J. Quero, and L. Franquelo, "Stochastic pulse coded arithmetic," in *International Symposium on Circuits and Systems*, vol. 1, 2000, pp. 599–602.

[4] W. Qian, X. Li, M. Riedel, K. Bazargan, and D. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Transactions on Computers*, vol. 60, no. 1, pp. 93–105, 2011.

[5] P. Li and D. Lilja, "Using stochastic computing to implement digital image processing algorithms," in *International Conference on Computer Design*, 2011, pp. 154–161.

[6] P. Li, D. Lilja, W. Qian, K. Bazargan, and M. Riedel, "The synthesis of complex arithmetic computation on stochastic bit streams using sequential logic," in *International Conference on Computer-Aided Design*, 2012, pp. 480–487.

[7] A. Alaghi and J. Hayes, "A spectral transform approach to stochastic circuits," in *International Conference on Computer Design*, 2012, pp. 315–321.

[8] G. Lorentz, *Bernstein Polynomials*. University of Toronto Press, 1953.

[9] R. Farouki and V. Rajan, "On the numerical condition of polynomials in Bernstein form," *Computer Aided Geometric Design*, vol. 4, no. 3, pp. 191–216, 1987.

[10] *Design Compiler*, Synopsys Inc.

[11] *Nangate FreePDK45 library*, Silicon Integration Initiative, Inc.