A General Design of Stochastic Circuit and Its Synthesis

Zheng Zhao and Weikang Qian
University of Michigan-Shanghai Jiao Tong University Joint Institute
Shanghai Jiao Tong University, Shanghai, China
Email: {zzhao.ee, qianwk}@sjtu.edu.cn

***Abstract*—Stochastic computing (SC) is an unconventional paradigm to realize arithmetic computation, where real values are encoded as stochastic bit streams. Compared with conventional computation on binary radix encoding, SC can perform arithmetic computation with very simple circuits. It also has strong tolerance to soft errors. In this paper, we introduce a general design of combinational circuit for stochastic computing, together with its analysis. We further show a synthesis method that can implement arbitrary arithmetic functions with the proposed design. The experimental results demonstrated that compared with the previous methods, our approach produces a circuit with much smaller area and delay.**

I. Introduction

With the continued scaling of CMOS technology, reliability has become a paramount concern in designing VLSI circuits as they are inevitably subject to greater process, voltage, and thermal variations [1]. A potential remedy to this problem is to implement computation using the stochastic computing (SC) paradigm [2], which is known to have strong tolerance to bit flip errors [3].

SC is an unconventional way to realize arithmetic computation by digital circuits. The major difference between SC and traditional binary computing is that SC represents numbers by the probability of 1s in stochastic bit streams. For example, the stream A in Fig. 1 contains four 1s out of the total 8 bits, so the value encoded by the stream A is $\frac{4}{8}$. The change of encoding leads to changes in the circuit design. Under the stochastic computing paradigm, many arithmetic functions can be realized with very simple circuits. As an example, multiplication can be realized by a single AND gate, as shown in Fig. 1, since the probability of obtaining a one at the output of an AND gate equals the product of the probabilities of obtaining a one at its inputs.

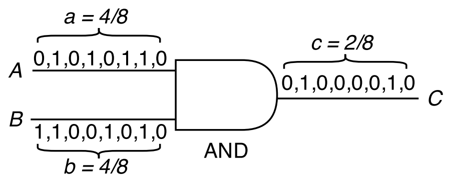

Fig. 1: An AND gate performs multiplication on real values encoded by stochastic bit streams.

SC is highly tolerant to bit flip errors. Since it uses a uniform-weight encoding, a single bit flip happened anywhere in the stream does not affect the encoded value significantly. SC has been successfully applied to a number of applications, such as image processing [4], [5], error-correcting coding [6], control [7], and artificial neural networks [8]. For a comprehensive survey of SC, the readers are referred to [9].

In the earlier works on SC, a number of stochastic circuits are designed manually, but they are restricted to basic computation units such as adder and multiplier [10]. In order to implement more general computation with SC, a few methods for synthesizing stochastic circuits were proposed recently. These methods can be classified into two categories, based on whether the circuit is combinational or sequential.

In [11], the authors proposed a special combinational circuit to implement polynomial computation. Their approach transforms a target univariate function into a so-called Bernstein polynomial. Then it uses an adder to implement each Bernstein term, and a multiplexer to generate the coefficient associated with each term. In [12], the authors introduced a method based on spectral transform to synthesize combinational circuits for stochastic computing. They applied the inverse Fourier transform to obtain the Boolean function for the given stochastic computation. Their methods significantly saves the area of the stochastic circuit.

A few other works considered synthesizing sequential circuits for stochastic computing [13]–[15]. The main idea of these works is to realize the target function as a linear combination of the steady state distribution functions of a finite state machine. Although realizing SC with sequential circuits can save the cost of stochastic number generation, their output bit streams are not a perfect random bit stream, which degrades the accuracy of successive computation.

In this paper, we consider implementing SC with a combinational circuit. We first introduce a general design. Then, we present a method to realize arbitrary arithmetic computation with the proposed design. Given a target function, it turns out that there are many combinational circuits with different Boolean functions that can realize the target in the stochastic domain. We propose effective heuristics to synthesize a low-cost combinational circuit. The experimental results demonstrated that compared with the previous methods, our approach produces a circuit with much smaller area and delay.

The main contributions of this paper are as follows:

- We introduce a general combinational design for SC, which only makes an assumption on the necessary input probabilities without any additional assumption on the underlying combinational circuit. We analyze the general behavior of the design in the stochastic domain.
- We propose a comprehensive methodology to synthesize a stochastic circuit for realizing an arbitrary target arithmetic function. We start by converting the target function into a special form that can be realized by the proposed design. However, there are many choices of Boolean functions that can realize the converted function in the stochastic domain. We demonstrate the characteristics of these candidate Boolean functions and present effective heuristics to determine a good choice that can lead to a combinational circuit with small area and delay.

The remainder of this paper is organized as follows. In Section II, we introduce the general design and analyze the computation it can implement. In Section III, we present the methodology to synthesize a special type of polynomial, namely, the multi-linear polynomial. In Section IV, we present our solution to synthesize a general polynomial. In Section V, we demonstrate the effectiveness of our method through the experimental results. Finally, we conclude the paper in Section VI.

II. A General Design of Stochastic Circuit

We aim at designing a stochastic circuit that can compute an arithmetic function $f(x_1, \dots, x_n)$, i.e., the output probability of the circuit is $f(x_1, \dots, x_n)$. We consider combinational circuits in this work. Clearly, there must be some inputs that take the variable probabilities x_i's to be a one. We assume that these input stochastic bit streams are generated using some standard modules in stochasitc computing, e.g., the randomizer unit shown in [3], and hence, are not the concern of this work. To add more freedom to the function we can realize, we also need some constant input probabilities. The constant probability we can obtain most easily is $\frac{1}{2}$. For example, each output bit of a linear feedback shift register can be viewed as a random Boolean variable with probability of $\frac{1}{2}$ [16]. Thus, our proposed design is a combinational circuit with $m + n$ inputs, $X_1, X_2, \dots, X_n$ and $Y_1, Y_2, \dots, Y_m$. Each X_i takes a variable probability $0 \le x_i \le 1$, and each Y_i takes a probability of $1/2$. The proposed design is shown in Fig. 2. We assume that all the input probabilities are independent.

The first question is what kind of function can be realized by the above design. The following theorem gives the answer.

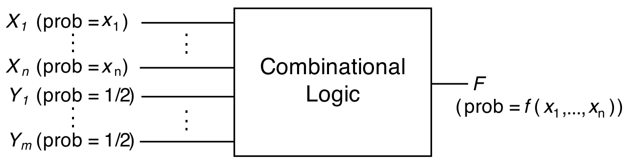

Fig. 2: Proposed general design.

Theorem 1
The proposed design shown in Fig. 2 can implement a function of the form

$$f(x_1,\ldots,x_n) = \sum_{\substack{(a_1,\ldots,a_n)\\ \in\{0,1\}^n}} \frac{g(a_1,\ldots,a_n)}{2^m} \prod_{j=1}^{n} {x_j}^{a_j}(1-x_j)^{1-a_j}. \quad (1)$$

where $0 \le g(a_1,\ldots,a_n) \le 2^m$ *is an integer dependent on the the combination* $(a_1,\ldots,a_n) \in \{0,1\}^n$. □

Note that the function f in Eq. (1) is a sum of 2^n terms. For example, if $n = 2$ and $m = 3$, the proposed design can implement a function of the form

$$f(x_1,x_2) = \frac{g(0,0)}{8}(1-x_1)(1-x_2) + \frac{g(0,1)}{8}(1-x_1)x_2 + \frac{g(1,0)}{8}x_1(1-x_2) + \frac{g(1,1)}{8}x_1x_2,$$

where $g(0,0), g(0,1), g(1,0), g(1,1)$ are integers in the range $[0,8]$.

Theorem 1 can be proved as follows. Suppose that the Boolean function realized by the circuit is $F(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$. We define the *on set* of a Boolean function to be the set of input combinations that let the function evaluate to 1. The output probability of the combinational circuit is equal to the sum of the probabilities that $(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$ takes the input combinations in the on set of the function F, i.e.,

$$\begin{aligned} P(F=1) &= \sum_{\substack{(a_1,\ldots,a_n,b_1,\ldots,b_m)\in\{0,1\}^{m+n}:\\ F(a_1,\ldots,a_n,b_1,\ldots,b_m)=1}} P(X_1{=}a_1,\ldots,X_n{=}a_n, Y_1{=}b_1,\ldots,Y_m{=}b_m) \\ &= \sum_{(a_1,\ldots,a_n)\in\{0,1\}^n} \sum_{\substack{(b_1,\ldots,b_m)\in\{0,1\}^m:\\ F(a_1,\ldots,a_n,b_1,\ldots,b_m)=1}} P(X_1{=}a_1,\ldots,X_n{=}a_n, Y_1{=}b_1,\ldots,Y_m{=}b_m) \end{aligned} \quad (2)$$

With X_i $(1 \le i \le n)$ as a random Boolean variable that has probability x_i to be 1, the probability that $X_i = 0$ is $(1-x_i)$ and the probability that $X_i = 1$ is x_i. Thus, the probability that $X_i = a_i \in \{0,1\}$ can be written as $x_i^{a_i}(1-x_i)^{(1-a_i)}$.

Since Y_i $(1 \le i \le m)$ is a random Boolean variable that has probability 0.5 to be 1, the probability of $X_1 = a_1,\ldots,X_n = a_n$, $Y_1 = b_1,\ldots,Y_m = b_m$ is

$$\frac{1}{2^m}\prod_{j=1}^{n} {x_j}^{a_j}(1-x_j)^{1-a_j}. \quad (3)$$

Note that the above probability does not depend on the values of $b_1,\ldots,b_m$.

Given any $(a_1,\ldots,a_n) \in \{0,1\}^n$, define $g(a_1,\ldots,a_n)$ to be the number of combinations $(b_1,\ldots,b_m) \in \{0,1\}^m$ that let $F(a_1,\ldots,a_n,b_1,\ldots,b_m) = 1$. Clearly, $0 \le g(a_1,\ldots,a_n) \le 2^m$. Then, for any given $(a_1,\ldots,a_n) \in \{0,1\}^n$, we have

$$\begin{aligned} &\sum_{\substack{(b_1,\ldots,b_m)\in\{0,1\}^m:\\ F(a_1,\ldots,a_n,b_1,\ldots,b_m)=1}} P(X_1{=}a_1,\ldots,X_n{=}a_n, Y_1{=}b_1,\ldots,Y_n{=}b_m) \\ &= \frac{g(a_1,\ldots,a_n)}{2^m}\prod_{j=1}^{n} {x_j}^{a_j}(1-x_j)^{1-a_j}. \end{aligned} \quad (4)$$

Putting Eq. (4) into Eq. (2), we obtain

$$P(F=1) = \sum_{(a_1,\ldots,a_n)\in\{0,1\}^n} \frac{g(a_1,\ldots,a_n)}{2^m}\prod_{j=1}^{n} {x_j}^{a_j}(1-x_j)^{1-a_j}.$$

This concludes the proof.

From the above proof, we can see that the value $g(a_1,\ldots,a_n)$ in Eq. (1) equals the number of combinations $(b_1,\ldots,b_m) \in \{0,1\}^m$ that let $F(a_1,\ldots,a_n,b_1,\ldots,b_m) = 1$. Thus, we can implement any function of the form shown in Eq. (2) by properly setting the Boolean function F.

Eq. (1) can be generalized into the form

$$f(x_1,\ldots,x_n) = \sum_{\substack{(a_1,\ldots,a_n)\\ \in\{0,1\}^n}} h(a_1,\ldots,a_n)\prod_{j=1}^{n} {x_j}^{a_j}(1-x_j)^{1-a_j}, \quad (5)$$

where each $h(a_1,\ldots,a_n)$ is a real number. Since each product term $\prod_{j=1}^n {x_j}^{a_j}(1-x_j)^{1-a_j}$ corresponds to a binary combination $(a_1,\ldots,a_n)$, we call the above form a *binary combination polynomial (BCP)*. We call $h(a_1,\ldots,a_n)$ the coefficient of the BCP.

III. Synthesis of Multi-linear Polynomial

We can see that the proposed stochastic circuit implements a special type of BCP, of which the coefficients are of the form $\frac{g}{2^m}$, where $0 \le g \le 2^m$ is an integer.

A further question is that given an arbitrary arithmetic function, how we can synthesize the combinational circuit in Fig. 2 so that the stochastic circuit implements the target function.

In what follows, we will only consider how to realize *polynomial* computation with the proposed stochastic circuit. There are two reasons for this. First, as shown in Eq. (1), the proposed stochastic circuit implements polynomial computation. Second, we can approximate any function closely as a multivariate polynomial, for example, by Taylor expansion. In this section, we will show how we can synthesize a stochastic circuit to realize a special type of polynomial, i.e., multi-linear polynomial. We will show our method to synthesize general polynomials in the next section.

Definition 1
A multi-linear polynomial (MLP) of n *variables is of the form*

$$f(x_1,\ldots,x_n) = \sum_{(a_1,\ldots,a_n)\in\{0,1\}^n} c(a_1,\ldots,a_n)\prod_{j=1}^{n} {x_j}^{a_j},$$

where each $c(a_1,\ldots,a_n)$ *is a real value. We call* $c(a_1,\ldots,a_n)$ *the coefficient of the MLP.* □

For example, a bivariate MLP is of the form

$$f(x_1,x_2) = c(0,0) + c(0,1)x_2 + c(1,0)x_1 + c(1,1)x_1x_2.$$

Since the inputs to the stochastic circuit represent probability values, in what follows, we further assume that the function arguments $x_1,\ldots,x_n$ are in the unit interval. Otherwise, we will first perform a linear transform on the input arguments to scale them into the unit interval.

Our method for synthesizing MLP includes two steps: determining the coefficients of BCP and synthesizing the circuit.

A. Determine the Coefficients of BCP

Given an MLP, the first step is to transform it into a BCP, since our proposed design computes a BCP.

The first question is: given an arbitrary MLP, can it always be transformed into a BCP? The answer is yes according to the following theorem.

Theorem 2
Given any MLP, it can be uniquely mapped to a BCP. □

A sketch of the proof is shown as follows. A BCP shown in Eq. (5) is a linear combination of 2^n polynomials $\prod_{j=1}^n {x_j}^{a_j}(1-x_j)^{1-a_j}$, $(a_1,\ldots,a_n) \in \{0,1\}^n$. To show that any multi-linear polynomial can be uniquely mapped to a BCP, we only need to prove that the above 2^n polynomials form a basis of the vector space of n-variable multi-linear polynomials. Since the dimension of the vector space of n-variable multi-linear polynomials is 2^n, (of which a well-known basis is $\{\prod_{j=1}^n {x_j}^{a_j} : (a_1,\ldots,a_n) \in \{0,1\}^n\}$), we only need to prove that the basis of the 2^n terms are linearly independent. This can be further proved by mathematical induction over n. The details are omitted due to the space limit.

To determine the coefficient $h(a_1,\ldots,a_n)$ of a BCP mapped from an MLP for any combination $(a_1,\ldots,a_n)\in\{0,1\}^n$, we can simply set $x_1=a_1,\cdots,x_n=a_n$ in Eq. (5). This gives us

$$h(a_1,\ldots,a_n)=f(a_1,\ldots,a_n). \tag{6}$$

For example, suppose that a bivariate MLP is

$$f(x_1,x_2)=c_1x_1x_2+c_2x_1+c_3x_2+c_4$$

Its equivalent BCP has the form shown in Eq. (5), i.e.,

$$\begin{aligned}f(x_1,x_2)=h(0,0)(1-x_1)(1-x_2)+h(0,1)(1-x_1)x_2\\+h(1,0)x_1(1-x_2)+h(1,1)x_1x_2,\end{aligned}$$

Then we can obtain coefficients h's as follows

$$\begin{aligned}&h(0,0)=f(0,0)=c_4,\\&h(0,1)=f(0,1)=c_3+c_4,\\&h(1,0)=f(1,0)=c_2+c_4,\\&h(1,1)=f(1,1)=c_1+c_2+c_3+c_4.\end{aligned}$$

Not all BCPs can be implemented by the proposed stochastic circuit. One requirement from Eq. (1) is that all the coefficients of the BCP must be in the unit interval. If some coefficients of the BCP are outside the unit interval, we need to further perform a linear transform on them.

Specifically, suppose that r and s are the minimum and the maximum of all the 2^n coefficients of a BCP. We transform each coefficient $h(a_1,\ldots,a_n)$ into $h'(a_1,\ldots,a_n)$ as

$$h'(a_1,\ldots,a_n)=\frac{h(a_1,\ldots,a_n)-r}{s-r}.$$

Then, all the $h'(a_1,\ldots,a_n)$'s are in the unit interval.

Now the target polynomial is

$$f'(x_1,\ldots,x_n)=\sum_{\substack{(a_1,\ldots,a_n)\\\in\{0,1\}^n}}h'(a_1,\ldots,a_n)\prod_{j=1}^{n}{x_j}^{a_j}(1-x_j)^{1-a_j},$$

which can be implemented using the proposed stochastic circuit. The original computation can be recovered from the stochastic computing result as $f(x_1,\ldots,x_n)=(s-r)f'(x_1,\ldots,x_n)+r$.

Finally, since the coefficients of a BCP implemented by the proposed stochastic circuit are of the form $\frac{g}{2^m}$, we need to approximate each coefficient $h'(a_1,\ldots,a_n)$ by the closest value of the form $\frac{g}{2^m}$. Technically, we use an iterative method to decide m. Given a precision requirement $prec$, we start m from 1, and check if the closest possible $\frac{g}{2^m}$, where $g\in\{0,1,\ldots,2^m\}$, can satisfy the precision requirement. If this turns out to be true, we stop with the current m; otherwise, we increase m by 1 until the requirement is met. Theoretically, the largest m to reach a precision $prec$ is $\lceil\log_2\frac{1}{prec}\rceil$. Yet, the iterative method can sometimes find a smaller m, and hence, smaller circuit complexity to satisfy the requirement.

B. Synthesize the Circuit

After the previous step, we finally obtain a BCP of the form shown in Eq. (1). In order to realize that BCP, we only need to select a Boolean function F satisfying that for any $(a_1,\ldots,a_n)\in\{0,1\}^n$, the number of combinations $(b_1,\ldots,b_m)\in\{0,1\}^m$ that let $F(a_1,\ldots,a_n,b_1,\ldots,b_m)=1$ is $g(a_1,\ldots,a_n)$. There are many Boolean functions that could satisfy this requirement. However, different functions lead to circuits of different areas.

Example 1

Suppose that we want to implement the function $f(x_1,x_2)=\frac{1}{2}(x_1+x_2)$ using the proposed stochastic circuit. If we choose $n=2$ and $m=1$, we can obtain

$$\begin{aligned}g(0,0)/2=f(0,0)=0,\quad g(0,1)/2=f(0,1)=1/2,\\g(1,0)/2=f(1,0)=1/2,\quad g(1,1)/2=f(1,1)=1\end{aligned}$$

Thus, we have

$$g(0,0)=0,g(0,1)=1,g(1,0)=1,g(1,1)=2.$$

This means that the numbers of $b\in\{0,1\}$ that let $F(0,0,b)=1$, that let $F(0,1,b)=1$, that let $F(1,0,b)=1$, and that let $F(1,1,b)=1$ are $0,1,1,$ *and* 2*, respectively. Thus, we have* $F(0,0,0)=F(0,0,1)=0$ *and* $F(1,1,0)=F(1,1,1)=1$*. However, there is freedom in determining which of $b\in\{0,1\}$ lets* $F(0,1,b)=1$ *and which of $b\in\{0,1\}$ lets* $F(1,0,b)=1$*. If we choose* $F(0,1,1)=1$ *and* $F(1,0,1)=1$*, then the Boolean function represented in a simplified sum-of-product (SOP) form is* $F=x_1y_1+x_2y_1+x_1x_2$*. If we choose* $F(0,1,0)=1$ *and* $F(1,0,1)=1$*, then the F can be represented as* $F=x_1y_1+x_2\bar{y_1}$*. If we measure the circuit area in terms of the number of literals, clearly, the latter choice produces a smaller circuit.* □

Among the many choices of Boolean function, we want to find one that leads to a circuit with minimized area. We formulate the synthesis problem as follows.

> Given 2^n integers $g(a_1,\ldots,a_n)$, where $0\le g(a_1,\ldots,a_n)\le 2^m$ for all $(a_1,\ldots,a_n)\in\{0,1\}^n$, synthesize a circuit with optimized area that implements a Boolean function $F(X_1,\ldots,X_n,Y_1,\ldots,Y_m)$ satisfying that for any $(a_1,\ldots,a_n)\in\{0,1\}^n$, the number of combinations $(b_1,\ldots,b_m)\in\{0,1\}^m$ that let $F(a_1,\ldots,a_n,b_1,\ldots,b_m)=1$ is $g(a_1,\ldots,a_n)$.

Finding an optimal solution to the above problem is a hard problem. In this section, we present a heuristic to construct a Boolean function that can be realized by a small circuit.

Define $M(a_1,\ldots,a_n,b_1,\ldots,b_m)$ to be the minterm corresponding to the combination $(a_1,\ldots,a_n,b_1,\ldots,b_m)$. Define $S(a_1,\ldots,a_n)$ to be the set of minterms

$$\{M(a_1,\ldots,a_n,b_1,\ldots,b_m):(b_1,\ldots,b_m)\in\{0,1\}^m\}.$$

We need to select $g(a_1,\ldots,a_n)$ minterms out of the set $S(a_1,\ldots,a_n)$ to compose the function F. Our basic strategy is to select these minterms so that they can be covered by as few cubes as possible. Here, a cube means a product of literals, e.g., $X_1\overline{X_2}$ is a cube.

Clearly, for each combination $(a_1,\ldots,a_n)$ such that $g(a_1,\ldots,a_n)=2^m$, we have to pick all the minterms in the set $S(a_1,\ldots,a_n)$; for each combination $(a_1,\ldots,a_n)$ such that $g(a_1,\ldots,a_n)=0$, we pick no minterms in the set $S(a_1,\ldots,a_n)$.

Now consider the remaining combinations $(a_1,\ldots,a_n)$'s such that $0<g(a_1,\ldots,a_n)<2^m$. Based on the binary representation of each $g(a_1,\ldots,a_n)$, we can represent it as a sum of powers of two,

$$g(a_1,\ldots,a_n)=\sum_{k=0}^{r-1}2^{i_k},$$

where r is the number of ones in the binary representation of $g(a_1,\ldots,a_n)$ and $0\le i_0\le i_1\le\cdots\le i_{r-1}\le m-1$ are integers. For example, we can represent $g=11$ as $g=2^3+2^1+2^0$.

The idea of our heuristic is to pick r cubes to cover $g(a_1,\ldots,a_n)$ minterms from each set $S(a_1,\ldots,a_n)$, with the k-th $(0\le k\le r-1)$ cube covering 2^{i_k} minterms. To ensure that these r cubes cover $g(a_1,\ldots,a_n)$ minterms, we require these cubes to be pairwise disjoint, i.e., they share no common minterms pairwise. Furthermore, we want to increase the chance that the cubes selected for different $(a_1,\ldots,a_n)$'s can be combined into a larger one. Thus, we require that for those combinations $(a_1,\ldots,a_n)$ that have a cube of size 2^k, their cubes of size 2^k cover the same set of $(b_1,\ldots,b_m)$'s.

To achieve this, for each combination $(a_1,\ldots,a_n)$, if $g(a_1,\ldots,a_n)$ contains a power of two, 2^k, then the corresponding cube of size 2^k picked are composed of 2^k minterms from the set $S(a_1,\ldots,a_n)$ with $b_1=\cdots=b_{m-k-1}=1,b_{m-k}=0$. With this approach, a cube of size 2^{m-1} is composed of the first 2^{m-1} minterms in the set $S(a_1,\ldots,a_n)$ (i.e., they are minterms with $b_1=0$), a cube of size 2^{m-2} is composed of the next 2^{m-2} minterms in the set (i.e., they are minterms with $b_1=1,b_2=0$), and so on.

It is not hard to see that the cubes picked for each combination $(a_1,\ldots,a_n)$ are pairwise disjoint and the cubes of size 2^k belonging to different combinations $(a_1,\ldots,a_n)$'s cover the same set of $(b_1,\ldots,b_m)$'s, which makes it possible to combine these cubes into larger ones.

IV. Synthesis of General Polynomial

In this section, we present our method of synthesizing the proposed stochastic circuit to implement a general polynomial. The first step of our method transforms the target polynomial into an *initial* BCP. As we will show in Section IV-B, different from the MLP case, there are a number of BCPs that are equivalent to a general polynomial. The next step determines an optimal BCP and then synthesizes the circuit.

A. Transform the Target Polynomial into an Initial BCP

Given a general polynomial $f(x_1, \dots, x_n)$, we first transform it into an initial BCP. To do this, we first transform it into an MLP. A simple way is to replace each ${x_i}^r$ term in the polynomial by a product of r new variables $x_{i,1}, x_{i,2}, \dots, x_{i,r}$, which takes the same value as x_i. For example, suppose that the target polynomial is

$$f = c_1 x_1^2 + c_2 x_2^2 + c_3 x_1 x_2 + c_4 x_1 + c_5 x_2 + c_6.$$

In order to convert it into an MLP, we replace the product x_1^2 by a product of two variables $x_{1,1}$ and $x_{1,2}$, the product x_2^2 by a product of two variables $x_{2,1}$ and $x_{2,2}$, and so on. The resulting polynomial is

$$f = c_1 x_{1,1} x_{1,2} + c_2 x_{2,1} x_{2,2} + c_3 x_{1,1} x_{2,1} + c_4 x_{1,1} + c_5 x_{2,1} + c_6,$$

which is an MLP on four variables $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}$. The original polynomial computation can be obtained by setting $x_{1,1} = x_{1,2} = x_1$ and $x_{2,1} = x_{2,2} = x_2$.

Once the MLP is determined, we can further obtain its equivalent BCP and the g values using the method shown in Section III-A. We denote the initial g values as g_0.

B. Synthesize the Circuit

In the case of general polynomial, we have flexibility in determining the BCP as the synthesis target. The initial BCP is just one of them. For example, consider a target univariate polynomial $f_0(x_1)$ of degree 2. It is transformed into an initial BCP by the procedure shown in the previous section:

$$\begin{aligned} f(x_{1,1}, x_{1,2}) = & \frac{g_0(0,0)}{2^m}(1 - x_{1,1})(1 - x_{1,2}) \\ & + \frac{g_0(0,1)}{2^m}(1 - x_{1,1}) x_{1,2} \\ & + \frac{g_0(1,0)}{2^m} x_{1,1}(1 - x_{1,2}) + \frac{g_0(1,1)}{2^m} x_{1,1} x_{1,2}. \end{aligned}$$

However, since in realizing the original polynomial, we set $x_{1,1} = x_{1,2} = x_1$, thus the product terms $(1 - x_{1,1}) x_{1,2}$ and $x_{1,1}(1 - x_{1,2})$ can be treated as equivalent. Thus, we could choose another BCP

$$\begin{aligned} f^*(x_{1,1}, x_{1,2}) = & \frac{g_0(0,0)}{2^m}(1 - x_{1,1})(1 - x_{1,2}) \\ & + \frac{g^*(0,1)}{2^m}(1 - x_{1,1}) x_{1,2} \\ & + \frac{g^*(1,0)}{2^m} x_{1,1}(1 - x_{1,2}) + \frac{g_0(1,1)}{2^m} x_{1,1} x_{1,2} \end{aligned}$$

as the synthesis target, as long as $g^*(0,1) + g^*(1,0) = g_0(0,1) + g_0(1,0)$.

In the general situation, suppose that the degree of x_i in the target polynomial is $d_i \geq 1$ ($i = 1, \dots, n$). Define $d = \sum_{i=1}^n d_i$. By the procedure of introducing new variables shown in Section IV-A, the BCP has d variables $x_{1,1}, \dots, x_{1,d_1}, \dots, x_{n,1}, \dots, x_{n,d_n}$, where the variables $x_{i,1}, \dots, x_{i,d_i}$ all equal a common variable x_i (for each $i = 1, \dots, n$). Each product term in the BCP is of the form

$$\prod_{j=1}^{n} \prod_{k=1}^{d_j} x_{j,k}^{a_{j,k}} (1 - x_{j,k})^{1 - a_{j,k}},$$

where $(a_{1,1}, \dots, a_{1,d_1}, \dots, a_{n,1}, \dots, a_{n,d_n}) \in \{0,1\}^d$. Since $x_{i,1} = \dots = x_{i,d_i} = x_i$, we can see that two product terms

$$\prod_{j=1}^{n} \prod_{k=1}^{d_j} x_{j,k}^{a_{j,k}} (1 - x_{j,k})^{1 - a_{j,k}}$$

TABLE I: Example of equivalence classes.

$a_{1,1}$	$a_{1,2}$	$a_{2,1}$	$a_{2,2}$	Class	Class ID
0	0	0	0	$I(0,0)$	0
0	0	0	1	$I(0,1)$	1
0	0	1	0	$I(0,1)$	1
0	0	1	1	$I(0,2)$	2
0	1	0	0	$I(1,0)$	3
0	1	0	1	$I(1,1)$	4
0	1	1	0	$I(1,1)$	4
0	1	1	1	$I(1,2)$	5
1	0	0	0	$I(1,0)$	3
1	0	0	1	$I(1,1)$	4
1	0	1	0	$I(1,1)$	4
1	0	1	1	$I(1,2)$	5
1	1	0	0	$I(2,0)$	6
1	1	0	1	$I(2,1)$	7
1	1	1	0	$I(2,1)$	7
1	1	1	1	$I(2,2)$	8

and

$$\prod_{j=1}^{n} \prod_{k=1}^{d_j} x_{j,k}^{b_{j,k}} (1 - x_{j,k})^{1 - b_{j,k}}$$

are equivalent if and only if for each $j = 1, \dots, n$, $\sum_{k=1}^{d_j} a_{j,k} = \sum_{k=1}^{d_j} b_{j,k}$.

Based on the above equivalence relation, we can partition the set of combinations $\{0,1\}^d$ into a number of equivalence classes $I(s_1, \dots, s_n)$, $0 \leq s_1 \leq d_1, \dots, 0 \leq s_n \leq d_n$, where

$$\begin{aligned} I(s_1, \dots, s_n) = \{ & (a_{1,1}, \dots, a_{n,d_n}) \in \{0,1\}^d : \\ & \sum_{k=1}^{d_j} a_{j,k} = s_j, \text{for each } j = 1, \dots, n \}. \end{aligned}$$

It is clear that two product terms are equivalent if and only if their corresponding combinations $(a_{1,1}, \dots, a_{n,d_n})$ belong to the same equivalence class.

For example, consider the case where $n = 2$, $d_1 = 2$, and $d_2 = 2$. The set of combinations $\{0,1\}^4$ can be partitioned into 9 equivalence classes as shown in Table I. The combinations in the same equivalence class lead to equivalent product terms. For instance, the combinations $(0,1,0,1)$, $(0,1,1,0)$, $(1,0,0,1)$, and $(1,0,1,0)$ all belong to the class $I(1,1)$, they produce the same product term $x_1(1 - x_1) x_2 (1 - x_2)$.

A sufficient condition that a BCP can generate the original polynomial is that its g value should satisfy that

$$\begin{aligned} & \sum_{\substack{(a_{1,1}, \dots, a_{n,d_n}) \\ \in I(s_1, \dots, s_n)}} g(a_{1,1}, \dots, a_{n,d_n}) \\ = & \sum_{\substack{(a_{1,1}, \dots, a_{n,d_n}) \\ \in I(s_1, \dots, s_n)}} g_0(a_{1,1}, \dots, a_{n,d_n}) \triangleq G(s_1, \dots, s_n), \qquad (7) \\ & \text{for all } 0 \leq s_1 \leq d_1, \dots, 0 \leq s_n \leq d_n, \end{aligned}$$

where g_0 is the g value of the initial BCP and $G(s_1, \dots, s_n)$ denotes the sum of g_0's over all the combinations in the set $I(s_1, \dots, s_n)$.

To obtain a circuit with small area, we need to properly choose g values which satisfy Eq. (7). A natural idea is to distribute the value $G(s_1, \dots, s_n)$ evenly among all combinations in the set $I(s_1, \dots, s_n)$.

A better strategy is to unevenly distribute $G(s_1, \dots, s_n)$. Define $k = \lfloor G(s_1, \dots, s_n)/2^m \rfloor$. The strategy sets the g values for the first k combinations in the set $I(s_1, \dots, s_n)$ to 2^m, the g value for the $(k+1)$-th combination to $(G(s_1, \dots, s_n) - 2^m k)$, and the g values for the remaining combinations in the set $I(s_1, \dots, s_n)$ to 0. This strategy essentially chooses the largest feasible cube for each of the first k combinations in the set $I(s_1, \dots, s_n)$. Our experimental results in Section V showed that this approach produces much smaller circuits than the even distribution strategy.

Once the g values for the BCP are determined, we apply the heuristic shown in Section III-B to synthesize the combinational circuit.

V. Experimental Results

We conducted three sets of experiments to demonstrate the effectiveness of the proposed method. In the experiments, we used Espresso [17] to compute the number of literals, and ABC [18] to obtain the area and delay of the combinational circuits. We used the standard script `resyn2` to perform multi-level synthesis, and MCNC generic standard cell library to map the circuits. We only considered polynomials with small degree in our experiments as they are sufficient for the precision requirement of many applications using SC [9], [12]. As our algorithm needs to build the truth table, the runtime of our algorithm is an exponential function on m and n. However, the runtime is tolerable in practice, since for practical applications of SC, n, the degree of polynomial, and m, determined by the computation precision, are usually small.

A. Synthesis of Multi-linear Polynomials

The first set of experiments compares our method with one of the state-of-the-art methods [12] in synthesizing multi-linear polynomials. We experimented with different numbers of variables X_i's and Y_i's, denoted by n and m, respectively. As shown in Fig. 3, the horizontal axis is for n, ranging from 4 to 10. Different curves the in the figure correspond to different m's, ranging from 4 to 11. The points for $m = 10, n = 11$ are not plotted due to the long synthesis time of Espresso.

For each (n, m) pair, we generated 100 random MLPs as the target functions. For each target function, we applied both the prior method and our method to synthesize the corresponding combinational circuit. We calculated the geometric means of the area, delay, and number of literals for both methods. The ratios of the means of our method to the means of the prior one are shown in Fig. 3 as the vertical coordinates. We can see that compared with the previous method [12], ours achieves much smaller circuit area and much fewer number of literals, together with smaller circuit delay. As n or m increases, the savings on the area and the number of literals grows.

B. Optimality Study

To study the effectiveness of our heuristic of constructing Boolean functions shown in Section III-B, we performed the second set of experiments, where a complete search over all the possible Boolean functions was conducted for a target MLP. As the number of possible Boolean functions grows exponentially with m and n (upper bounded by $\binom{2^m}{2^{m-1}}^{2^n}$), we only considered target polynomials of small sizes, of which the number of possible Boolean functions is practical for a complete search.

We randomly chose 6 such polynomials, which are listed in Table II. Column 2 to 5 show the basic statistics of these polynomials: the numbers of variables X_i's (n), the numbers of Y_i's (m), the sets of corresponding g's (ordered by the binary combinations), and the numbers of possible Boolean functions (#choices).

For each test case, we enumerated all the possible Boolean functions. The minimum and geometric mean of the area, delay, and number of literals over all the choices are listed in Column 6 to 8, and Column 9 to 11, respectively. The results of our method are shown in column 12 to 17. The ranks of our solutions among all the solutions are also listed. For example, for `case1`, the area, delay, and number of literals of our solution are ranked top 4.17%, 3.90% and 8.33% among all the solutions, respectively. As can be seen, although our method does not guarantee to provide the optimal solution for these test cases, it is able to achieve a reasonably good one.

C. Synthesis of General Polynomials

The third set of experiments compares the uneven distribution of G heuristic with the even distribution of G heuristic discussed in Section IV-B in synthesizing general polynomials (rather than an MLP). As a representative, we considered general univariate polynomial.

The experiment setup is similar as the first experiment, except that in this study we generated 100 *univariate* polynomials randomly and n refers to the degree of those univariate polynomials. For each target function, we applied both heuristics to synthesize the corresponding combinational circuit, and computed the geometric means of the area, delay and number of literals. The ratios of the means of the uneven distribution to those of the even distribution are plotted in Fig. 4. Besides those randomly generated functions, we also synthesized two benchmark functions: $f(x) = \cos(x)$ and gamma correction function $f(x) = x^{0.45}$, which is commonly used in image processing [19]. We also obtained the ratios of the synthesis results by the uneven distribution to those by the even distribution. The ratios for $f(x) = \cos(x)$ and $f(x) = x^{0.45}$ are plotted in Fig. 5 and 6, respectively. The results shows that the uneven distribution heuristic generates much better circuits than the even distribution heuristic in terms of area, delay, and number of literals.

VI. Conclusion

In this work, we propose a general combinational design for stochastic computing and its associated synthesis method for realizing an arbitrary arithmetic function. We demonstrate the flexibility in choosing the Boolean functions to realize the target function, and present effective heuristics to determine a good choice to reduce the circuit area. The experimental results showed that compared with the previous method, our method generates a stochastic circuit with smaller area and delay.

Acknowledgement

This work is supported by National Natural Science Foundation of China (NSFC) under Grant No. 61204042 and 61472243. The authors would like to thank Mr. Armin Alaghi and Prof. John Hayes from University of Michigan for their help on the experiments.

References

[1] S. Borkar, T. Karnik, and V. De, "Design and reliability challenges in nanometer technologies," in *Design Automation Conference*, 2004, p. 75.

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

[3] 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.

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

[5] A. Alaghi, C. Li, and J. Hayes, "Stochastic circuits for real-time image-processing applications," in *Design Automation Conference*, 2013, pp. 1–6.

[6] S. Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, and W. Gross, "Majority-based tracking forecast memories for stochastic LDPC decoding," *IEEE Transactions on Signal Processing*, vol. 58, no. 9, pp. 4883–4896, 2010.

[7] D. Zhang and H. Li, "A stochastic-based fpga controller for an induction motor drive with integrated neural network algorithms," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 2, pp. 551–561, 2008.

[8] B. Brown and H. Card, "Stochastic neural computation II: Soft competitive learning," *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 906–920, 2001.

[9] A. Alaghi and J. Hayes, "Survey of stochastic computing," *ACM Transactions on Embedded Computing Systems*, vol. 12, no. 2s, pp. 92:1–92:19, 2013.

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

[11] W. Qian and M. Riedel, "The synthesis of robust polynomial arithmetic with stochastic logic," in *Design Automation Conference*, 2008, pp. 648–653.

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

[13] P. Li, W. Qian, M. Riedel, K. Bazargan, and D. Lilja, "The synthesis of linear finite state machine-based stochastic computational elements," in *Asia and South Pacific Design Automation Conference*, 2012, pp. 757–762.

[14] 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.

[15] N. Saraf, K. Bazargan, D. Lilja, and M. Riedel, "Stochastic functions using sequential logic," in *International Conference on Computer Design*, 2013, pp. 507–510.

[16] J. Alspector, J. Gannett, S. Haber, M. Parker, and R. Chu, "A VLSI-efficient technique for generating multiple uncorrelated noise sources and its application to stochastic neural networks," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 1, pp. 109–123, 1991.

[17] R. Rudell, "Multiple-valued logic minimization for pla synthesis," *Technical Report, University of. California, Electronics Research Laboratory, Berkeley*, 1986.

[18] "Berkeley logic synthesis and verification group, ABC: A system for sequential synthesis and verification, release 20140530." http://www.eecs.berkeley.edu/ alanmi/abc/.

[19] D. Lee, R. Cheung, and J. Villasenor, "A flexible architecture for precise gamma correction," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 4, pp. 474–478, 2007.

TABLE II: Comparison of the Boolean function chosen by the heuristic proposed in Section III-B with all the feasible choices.

case no.	statistics				min			geomean			ours					
	n	m	$g(a_1, \ldots, a_n)$	#choices	area	delay	lit.	area	delay	lit.	area	rank(%)	delay	rank(%)	lit.	rank(%)
1	2	2	2 1 1 3	384	10	3.20	8	18.06	4.44	13.18	13	4.17	3.90	3.65	12	8.33
2	2	2	3 2 2 3	576	10	3.10	7	18.30	4.42	12.83	11	0.35	4.00	12.50	7	0.00
3	2	3	1 7 1 7	4096	9	3.00	7	23.26	5.10	17.11	10	0.10	3.50	0.34	7	0.00
4	2	3	7 2 1 5	100352	16	3.20	15	32.19	5.44	24.65	22	1.38	5.10	23.21	16	0.38
5	3	2	3 2 3 3 4 1 1 3	24576	18	4.00	13	37.37	5.82	26.15	24	0.93	5.00	3.77	17	0.49
6	3	2	1 2 2 3 1 2 4 4	13824	8	2.60	8	30.65	5.40	21.60	17	0.90	4.70	8.93	13	0.81

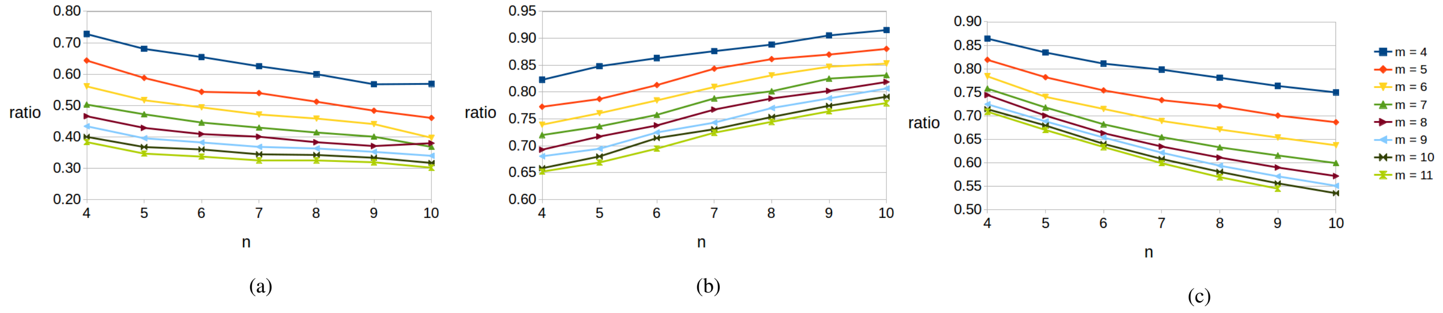

Fig. 3: The ratios of the synthesized results by our method to those by the prior method [12]: (a) the area ratio; (b) the delay ratio; (c) the ratio of the number of literals.

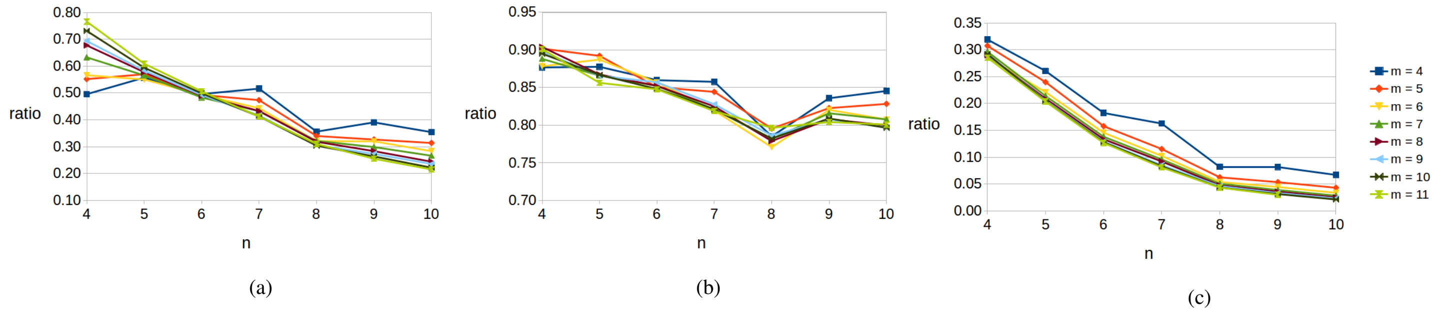

Fig. 4: The ratios of the synthesized results by the uneven distribution heuristic to those by the even distribution heuristic The synthesis targets are 100 randomly generated polynomials. The data plotted are the geometric means. (a) the area ratio; (b) the delay ratio; (c) the ratio of the number of literals.

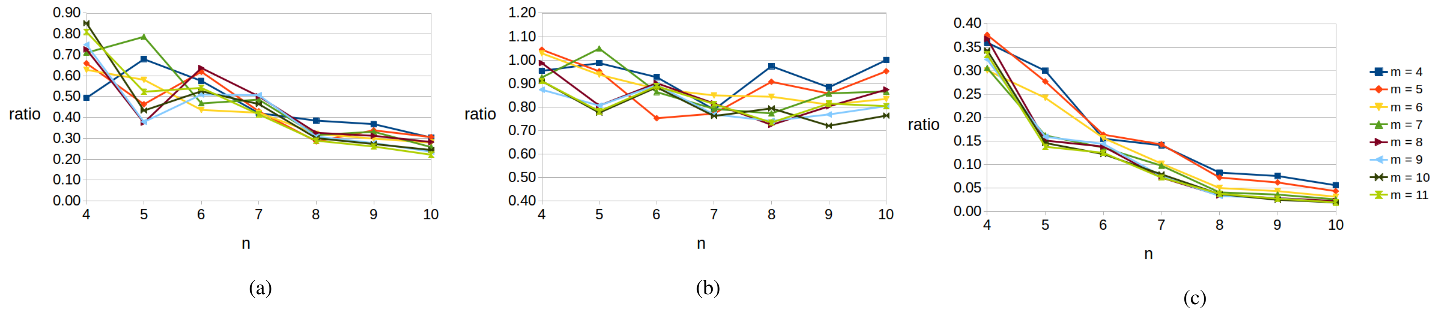

Fig. 5: The ratios of the synthesized results for the target function $f(x) = \cos(x)$ by the uneven distribution heuristic to those by the even distribution heuristic: (a) the area ratio; (b) the delay ratio; (c) the ratio of the number of literals.

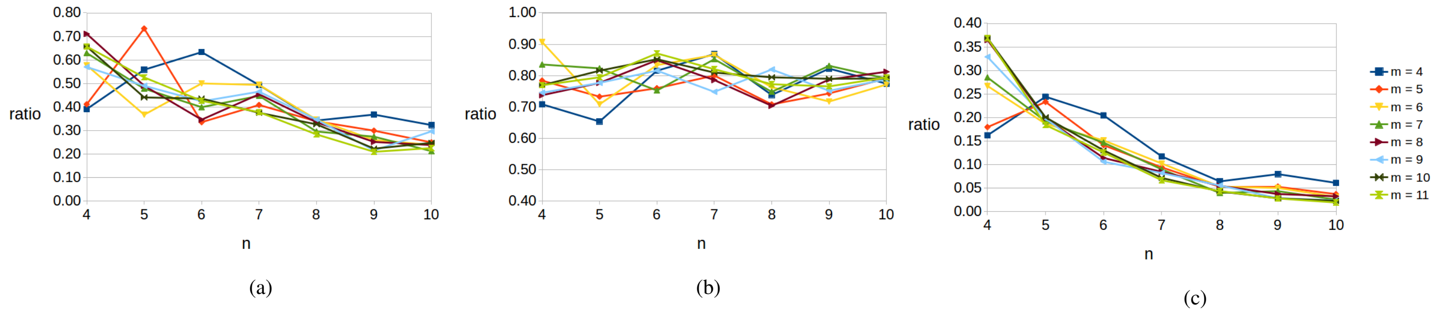

Fig. 6: The ratios of the synthesized results for the target function $f(x) = x^{0.45}$ (i.e., the gamma correction function) by the uneven distribution heuristic to those by the even distribution heuristic: (a) the area ratio; (b) the delay ratio; (c) the ratio of the number of literals.