Towards Theoretical Cost Limit of Stochastic Number Generators for Stochastic Computing

Meng Yang*, Bingzhe Li†, David J. Lilja†, Bo Yuan‡, Weikang Qian*
*University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, China
†Department of Electrical and Computer Engineering, University of Minnesota, U.S.A.
‡Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
Email: *{yangm.meng, qianwk}@sjtu.edu.cn, †{lixx1743, lilja}@umn.edu, ‡boyuan@sjtu.edu.cn

***Abstract*—Stochastic number generator (SNG) is one important component of stochastic computing (SC). An SNG usually consists of a random number source (RNS) and a probability conversion circuit (PCC). The SNGs occupy a large portion of the total area and power of a stochastic circuit. Thus, it is critical to lower the area and power of the SNGs. The existing methods only focused on simplifying the RNSs inside the SNGs, such as sharing the RNSs and using emerging devices. However, how to reduce the area and power of PCCs is never studied. In this work, we explore this problem and propose a solution that can effectively reduce the area and power of PCCs. We also study the theoretical limit on the cost of SNG and find that our proposed design approaches the limit. The experimental results show that our design can gain up to $2\times$ improvement in power-delay product over the traditional SNGs.**

I. Introduction

Stochastic computing (SC), an unconventional computing paradigm proposed long time ago [1], has gained more and more attention recently [2]. Different from binary computing, it operates on stochastic bit streams, which encode values through the probabilities of 1s in the streams. For example, as shown in Fig. 1(a), the bit stream 11010111 encodes the value 6/8. Based on this encoding mechanism, SC is highly tolerant of bit flip-flop errors. Moreover, its probabilistic nature allows very simple circuits to realize complex arithmetic operations. For example, an AND gate can implement multiplication, as shown in Fig. 1(a). Due to its low hardware cost, it is found to be suitable in applications such as image processing [3], low-density parity-check (LDPC) decoding [4], and artificial neural networks [5].

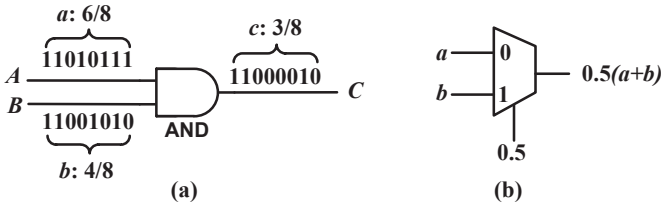


Figure 1: Examples of stochastic computing elements: (a) An AND gate for multiplying two numbers encoded by stochastic bit streams; (b) A 2-to-1 multiplexer (MUX) implementing the scaled addition.

To interface with conventional binary computing, SC requires a component that converts a binary number into a stochastic bit stream. This component is called *stochastic number generator (SNG)*, as shown in the shaded block in Fig. 2. An SNG consists of a random number source (RNS) and a probability conversion circuit (PCC). Typically, the RNS generates a random binary number uniformly distributed in the range $[0, 2^k-1]$, where k is the bit width of the random binary number. Equivalently, it can be viewed as a set of k unbiased random bits, which are of probability of 0.5 to be a 1. In Fig. 2, the RNS is a linear feedback shift register (LFSR). The PCC is a combinational circuit that takes k unbiased random bits and k target bits $c_{k-1}, \ldots, c_0$ as inputs. It produces an output bit stream with probability of $C/2^k$, where C is the binary number encoded by the target bits as $C = (c_{k-1} \ldots c_0)_2$. Since the parameter k determines the precision of the probability, in what follows, we call k the *probability precision*. In Fig. 2, the PCC is a comparator. The circuit taking stochastic bit streams as inputs and outputs is referred to as an *SC core*. For example, the AND gate in Fig. 2 is the SC core. Generally, in a stochastic circuit, the cost of the SNGs is much larger than that of the SC core, which can be easily seen from Fig. 2.

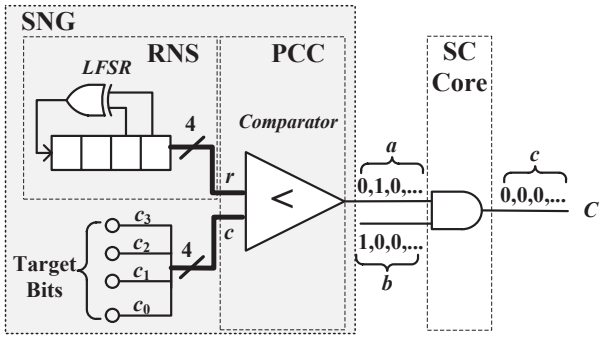


Figure 2: A complete implementation of multiplication in SC.

In literature, researchers have proposed several methods to reduce the cost of the SNGs. For some situations, accurate computation requires different input bit streams to be independent. One example is multiplication using an AND gate. A few works proposed methods to reduce the SNG cost to provide independent input stochastic bit streams. In [6], the authors proposed a method that shuffles the outputs of a single RNS to produce multiple target bit streams with low correlation. In [7], the authors proposed a method that inserts the delay elements, such as D flip-flops (DFFs), to obtain approximately independent inputs. For these two methods, only one RNS is required to generate multiple independent bit streams. Another

TABLE I: Area breakdown for two stochastic implementations of a 32-tap finite impulse response (FIR) filter [12].

area	w/o RNS sharing		with RNS sharing	
	area	%	area	%
RNS	3385	70	54	3.6
PCC	1274	26.4	1274	84.6
Other	178	3.6	178	11.8
Total	4837	100	1506	100

way is using emerging low-power nanoscale devices [8], such as memristor [9] and spintronic devices [10], as the RNSs.

There exist some other situations where accurate computation does not require different input bit streams to be independent. For example, in SC, a multiplexer (MUX) can realize a scaled addition $(a+b)/2$, as shown in Fig. 1(b). For this design, we do not require the input stochastic bit streams a and b to be independent. This is known as *correlation insensitivity* [11]. Another example is parallel applications such as image processing. The inputs to different output pixels can share a single RNS, which reduces the cost of the SNGs.

Although progress has been made to reduce the SNG cost, the existing methods only focused on simplifying the RNSs inside the SNGs. However, besides an RNS, an SNG also includes a PCC. As the RNS cost has been reduced through various techniques, the cost of the PCCs could now dominate the cost of the SNGs. For example, in [12], the authors have exploited the RNS sharing to reduce the SNG cost used in digital filter design. A summary of their results is shown in Table I. From the table, we can see that after RNS sharing is applied, the area of the RNSs in the entire stochastic circuit is significantly reduced. Consequently, the entire stochastic circuit area reduces. However, the PCC area does not change. It now dominates the area of the circuit, accounting for 84.6% of the entire area. Thus, to further reduce the cost of SNG, it is imperative to study how to reduce the cost of PCC. However, to the best of our knowledge, this problem is never studied in literature. In this work, we try to propose a solution to this problem. We have made the following contributions:

1) We propose a method to reduce the cost of a type of PCC called weighted binary generator (WBG). With this method, we can reduce the cost of SNG built with WBG.
2) For the first time, we analyze the theoretical cost limit of SNGs. We make a conjecture on the exact minimal cost of SNGs and demonstrate that the proposed design achieves this minimal cost.
3) We empirically study the accuracy, area, power, and delay of the proposed design. The results show that our design achieves significant area and power reduction with small delay increase and no accuracy loss, compared to the traditional SNG design. In terms of power-delay product, the proposed design achieves $2\times$ improvement.
4) We apply the proposed technique to stochastic implementations of a deep neural network (DNN) application and demonstrate that it can further reduce the area and power consumption of the stochastic implementations. Furthermore, the proposed stochastic implementation has much smaller area, power, and energy than the traditional binary implementation with a slight accuracy loss.

The rest of this paper is organized as follows. In Section II, we introduce the wegithed binary generator (WBG). In Section III, we present the proposed method to reduce the cost of the SNG built with the WBG. We also analyze the theoretical cost limit and demonstrate that our proposed design achieves this limit. In Section IV, we show the experimental results. Finally, in Section V, we conclude the paper.

II. Weighted Binary Generator

Weighted binary generator (WBG) is a PCC introduced by Gupta and Kumaresan [13]. A WBG with the probability precision $k = 4$ is shown in Fig. 3. The AND gates in the first level take unbiased random bits $r_3, \ldots, r_0$ as inputs and produce the intermediate signals

$$\begin{aligned} &w_3 = r_3, w_2 = \overline{r_3} \wedge r_2, w_1 = \overline{r_3} \wedge \overline{r_2} \wedge r_1, \\ &w_0 = \overline{r_3} \wedge \overline{r_2} \wedge \overline{r_1} \wedge r_0, \end{aligned} \tag{1}$$

where $\wedge$ represents the logical AND. The AND gates in the second level take $w_3, \ldots, w_0$ and the target bits $c_3, \ldots, c_0$ as inputs and produce the intermediate signals $u_i = w_i \wedge c_i$, for $i = 0, \ldots, 3$. Finally, a tree of 3 two-input OR gates takes $u_3, \ldots, u_0$ as inputs and produces the final output signal $x = u_3 \vee \cdots \vee u_0$, where $\vee$ represents the logical OR.

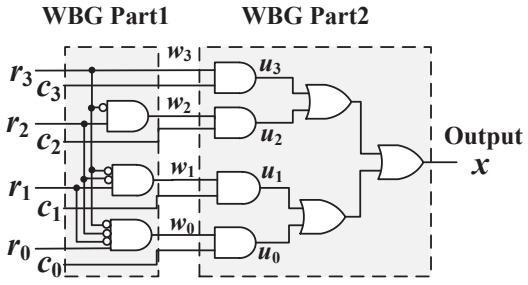


Figure 3: A weighted binary generator with the probability precision $k = 4$.

Given Eq. (1), the probabilities of w_i's are

$$P(w_3) = \frac{1}{2}, P(w_2) = \frac{1}{2^2}, P(w_1) = \frac{1}{2^3}, P(w_0) = \frac{1}{2^4}.$$

By Eq. (1), there cannot exist two w_i's that are both 1. Given this property, the probability of the output x is

$$\begin{aligned} P(x) &= P(w_3)\, c_3 + P(w_2)\, c_2 + P(w_1)\, c_1 + P(w_0)\, c_0 \\ &= \frac{1}{2}c_3 + \frac{1}{2^2}c_2 + \frac{1}{2^3}c_1 + \frac{1}{2^4}c_0 = \frac{C}{2^4}, \end{aligned}$$

where $C = (c_3 \ldots c_0)_2$. Thus, the WBG functions as a PCC.

III. SNG Cost Reduction and SNG Cost Limit

In this section, we present our proposed method of reducing the cost of the SNGs built with WBGs. Then, we analyze the theoretical cost limit of SNG and demonstrate that the proposed SNG cost reduction method achieves this limit. Finally, we summarize when our proposed SNG cost reduction method can be applied.

A. SNG Cost Reduction

In this section, we will demonstrate a method that reduces the cost of the SNGs built with WBGs. The method works when the accurate computation does not require different input bit streams to be independent.

We can decompose the WBG into two parts, the set of AND gates in the first level that produces the signals w_i and the remaining set of gates that produces the final output x. For simplicity, we call the former *WBG part 1* and the latter *WBG part 2*. They are circled out in Fig. 3.

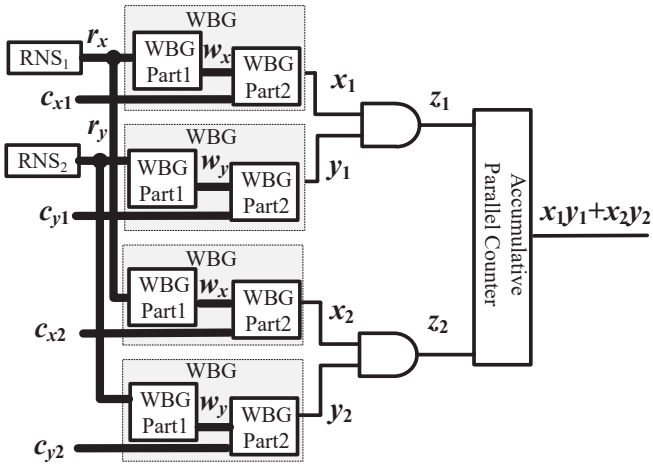


Figure 4: A stochastic implementation of the function $x_1y_1 + x_2y_2$, which uses the WBG as the PCC and applies the RNS sharing.

Suppose that we want to produce two input bit streams that could correlate but are of different probabilities. For example, suppose that we want to realize the sum of two products $x_1y_1 + x_2y_2$, where x_1, x_2, y_1, y_2 are specified by the users. A realization of this is shown in Fig. 4 It is a mixture of stochastic computing and binary computing. It uses two AND gates to multiply stochastic bit streams and generates two bit streams for the products x_1y_1 and x_2y_2. To get the final sum as a binary number, an accumulative parallel adder (APC) is applied [14]. Note that with the APC, the two output stochastic bit streams z_1 and z_2 could be correlated. Consequently, the inputs x_1 and x_2 could be correlated and the same for the inputs y_1 and y_2. Exploiting this fact, we can share an RNS for generating x_1 and x_2 and also share an RNS for generating y_1 and y_2, as shown in Fig. 4.

However, carefully examining the design reveals that the WBG part 1 for x_1 and that for x_2 can also be shared. The same holds for the WBG part 1 for y_1 and that for y_2. This extra sharing leads to a design with reduced SNG area, as shown in Fig 5. Such a sharing can be generalized for an arbitrary number of input bit streams that could be correlated. Note that this sharing method is essentially based on reducing the cost of PCC, which is different from all the existing SNG simplification methods.

B. Theoretical SNG Cost Limit

As discussed in the introduction, each input bit stream requires an SNG. Thus, in order to reduce the total SNG cost,

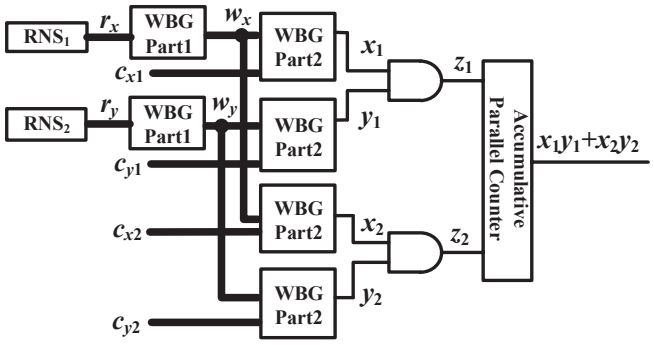


Figure 5: A stochastic implementation of the function $x_1y_1 + x_2y_2$ with the proposed sharing of WBG part 1.

we should reduce the SNG cost per input. As several methods were discovered to reduce the SNG cost, a natural question that raises to us is: what is the lower bound on SNG cost per input? We believe this question is important to the study of SC. In this section, we analyze this bound. In what follows, we will consider the situation where the accurate computation does not require different input bit streams to be independent.

Consider a stochastic circuit with N inputs. The very basic SNG design requires N SNGs, each consisting of an RNS and a PCC. Denote the areas of the RNS and the PCC as A_R and A_P, respectively. Then, the SNG area per input is simply

$$A_R + A_P.$$

Since we assume that different input bit streams could correlate, N input bit streams can share one RNS. However, for all the existing solutions, each input requires a separate PCC. In this case, the SNG area per input is

$$\frac{A_R + NA_P}{N} = \frac{A_R}{N} + A_P.$$

Obviously, the RNS sharing method reduces the SNG cost per input compared to the basic SNG design method.

As we demonstrated in Section III-A, the traditional view that each input requires a separate PCC may not be true. For PCC like WBG, we can identify some part in the PCC that can also be shared among different inputs. For such cases, we can further reduce the SNG area per input. We denote the area of the sharable part in the PCC as $A_{P,S}$ and that of the remaining non-sharable part of the PCC as $A_{P,NS}$. We have $A_P = A_{P,S} + A_{P,NS}$. Given that only one sharable part of the PCC is required for N inputs, the SNG area per input is

$$\frac{A_R + A_{P,S} + NA_{P,NS}}{N} = \frac{A_R + A_{P,S}}{N} + A_{P,NS}. \quad (2)$$

The first term in Eq. (2) depends on the number of inputs N. For a large N, which is possible for those massively parallel applications, the first term in Eq. (2) approaches 0. Thus, for a large N, the SNG area per input approaches $A_{P,NS}$, the area of the non-sharable part of the PCC.

In what follows, we define the *cost* of a circuit as the number of 2-input gates in the circuit when it is implemented with only 2-input gates and inverters. We define the *asymptotic cost*

of SNG per input as the cost of the SNG per input when the number of inputs N approaches infinity. By Eq. (2), the asymptotic cost of SNG per input is equal to the cost of the non-sharable part of the PCC.

To get a lower bound on the SNG area per input, we are interested in the *minimal* asymptotic cost of SNG per input. For simplicity, we call it the *minimal asymptotic SNG cost*. From the above discussion, it is equal to the minimal cost of the non-sharable part of any PCC. Next, we analyze this minimal cost.

We assume that the N input probabilities are different. Since each PCC is responsible for one input probability and all the input probabilities could be different, the non-sharable part of a PCC should include the final output of the PCC and the k target bits that determine the probability of a specific input. Furthermore, due to the existence of the randomness in the input bit stream produced by the PCC, the non-sharable part should take at least one random bit stream as its inputs. Thus, for the non-sharable part, it has at least $(k+1)$ inputs. For example, consider the WBG. Its non-sharable part is the WBG part 2. It includes the final output and k target bits. Furthermore, it has k random bit streams as its inputs.

For a circuit with n inputs, if it is only built with 2-input gates and inverters, at least $(n-1)$ 2-input gates are needed. By the definition of cost, its cost is at least $(n-1)$. Given that the non-sharable part of any PCC has at least $(k+1)$ inputs, the cost of the non-sharable part of any PCC is at least k. This gives a lower bound on the minimal cost of the non-sharable part of any PCC. Now consider the WBG. Its non-sharable part is the WBG part 2, which consists of k AND gates and $(k-1)$ OR gates. Thus, the cost of the non-sharable part of the WBG is $(2k-1)$. This gives an upper bound on the *minimal* cost of the non-sharable part of any PCC. As we mentioned above, since the minimal asymptotic SNG cost is equal to the minimal cost of the non-sharable part of any PCC, we conclude the following claim.

Theorem 1
Suppose that the probability precision of the SNG is k. A lower bound on the minimal asymptotic SNG cost is k, while an upper bound on it is $(2k-1)$.

It seems that there is still a gap between the lower bound and the upper bound on the minimal asymptotic SNG cost. Thus, the exact minimal asymptotic SNG cost is still unknown. However, we have the following conjecture on the exact value.

Conjecture 1
Suppose that the probability precision of the SNG is k. The minimal asymptotic SNG cost is $(2k-1)$.

To prove the conjecture, we need to show that the lower bound on the minimal asymptotic SNG cost is $(2k-1)$. However, we are not able to prove this strictly. In what follows, we just give an informal explanation to this. For the given target bits $c_{k-1}, \ldots, c_0$, the output probability is

$$P = \sum_{i=1}^{k} \frac{1}{2^i} c_{k-i}.$$

The above probability expression involves k multiplications and $(k-1)$ additions. To realize this weighted sum, there must exist k 2-input AND gates in the circuit. The i-th $(1 \le i \le k)$ one takes c_{k-i} as one input and a random bit stream as the other input. Furthermore, some additional 2-input gates are needed to merge the outputs of these k 2-input AND gates into the final single output. The number of these gates should at least be $(k-1)$. Thus, such a circuit should include at least $(2k-1)$ 2-input gates. This explains why we conjecture that the lower bound is also $(2k-1)$.

If the conjecture holds, then the WBG sharing method proposed in Section III-A realizes the minimal asymptotic SNG cost. This indicates the optimality of the proposed WBG sharing method.

C. Suitable Situations

In this section, we describe the suitable situations where the proposed WBG sharing method can be used. One thing to note is that the if multiple input probabilities are produced by the proposed sharing method, these inputs are correlated. Therefore, in order to apply the proposed method, we require that the computation accuracy will not be affected when the inputs are correlated. Given this requirement, there are the following two situations where the proposed method can be applied.

1) The inputs belong to the same function and their correlation does not affect the correctness of the final computation. Using the architecture shown in Fig. 5 to realize the function $x_1y_1 + x_2y_2$ is such an example.
2) The inputs belong to different functions. One example is the parallel applications. For example, consider the gamma correction [15] application used in image processing. It applies the computation $y = x^\gamma$ to each pixel, where x and y are the normalized pixel value in the input and the output image, respectively, and γ is a parameter. For this application, multiple pixels can be processed in parallel. For example, we can compute $y_1 = x_1^\gamma$ and $y_2 = x_2^\gamma$ in parallel, where x_1 and x_2 are the values of two different input pixels. In this case, the input probabilities x_1 and x_2 can be correlated.

IV. Experimental results

In this section, we present the experimental results to validate the benefits of the proposed WBG sharing method. In some of the following experiments, we compared three different SNG designs. The first design uses the comparator as the PCC. This is the widely used SNG design. We call it *CMP*. The second design uses the WBG as the PCC, but without applying the WBG sharing method. We call it *non-shared WBG*. The third design uses the proposed WBG sharing method to realize the PCC. We call it *shared WBG*.

A. Accuracy Study

In this section, we study the accuracy of the proposed shared WBG design. We compared two stochastic implementations for the function $x_1y_1 + x_2y_2$. The first one uses the proposed shared WBG design, as shown in Fig. 5. The RNS used in this design is the LFSR. The second one uses the CMP design. Furthermore, each SNG has its own RNS. The second

implementation is shown in Fig. 6. Table II compares the mean absolute error (MAE) of the two designs for different bit stream lengths. From the results, we can see that the MAE for the proposed shared WBG design is even smaller than the traditional CMP design in most cases. It demonstrates the high accuracy of the proposed sharing method.

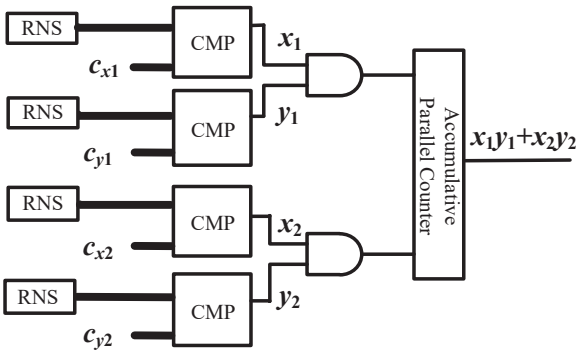


Figure 6: The stochastic implementation of $x_1y_1 + x_2y_2$ using the traditional SNGs.

TABLE II: Mean absolute error of the two stochastic implementations of $x_1y_1 + x_2y_2$ shown in Figs. 5 and 6.

Bit stream length	16	32	64	128	256	512	1024
Shared WBG	0.0462	0.0312	0.02	0.0164	0.0135	0.0081	0.0054
CMP	0.0773	0.0442	0.0269	0.0171	0.0118	0.0086	0.0058

B. Hardware Cost Study

In this section, we study the hardware cost of the proposed shared WBG design. To study the effectiveness of the proposed design in reducing the SNG cost, we just focus on the SNG part for generating multiple correlated inputs.

Fig. 7 shows the area, power, and delay of three SNG designs, CMP, non-shared WBG, and shared WBG, for different numbers of inputs. The probability precision k is 8. The red, blue, and green curves in each figure correspond to the CMP, the non-shared WBG, and the shared WBG designs, respectively. Note that for all the three SNG designs, a single RNS is shared among multiple target probabilities. In each figure, we chose the number of inputs as $2, 4, \ldots, 10$. Figs. 7a, 7b, and 7c show the area, power, and delay comparison, respectively.

As we can see, for all three SNG designs, both the area and power increase linearly with the number of inputs, which is expected. The proposed shared WBG design has much smaller area and power than the other two designs. Its delay, however, is larger than the other two designs. The reason is due to factors like logic fanouts. When the number of inputs is 10, the proposed design saves 43.4% area and 30.8% power compared to the non-shared WBG design. Its delay increases slightly by 1.02%. Compared to the CMP design, the proposed design saves 44.3% area and 55.7% power, with a delay increase of 13.5%. In terms of power-delay product, the proposed design has a reduction of 30.1% and 49.7% over the non-shared WBG design and the CMP design, respectively.

C. Case Study: Neural Network Classifier

In this section, we apply the proposed shared WBG design to a real application, neural network classifier. Neural network (NN) is widely used in computer vision, decision making, etc., nowadays. For modern deep neural networks (DNNs), they need numerous multiplications and additions. Thus, their power consumption could be huge. Meanwhile, since NN is usually used for classification tasks, which can tolerate a small amount of error, the computation in NNs does not require high accuracy. SC happens to target at computation that does not need high accuracy and can reduce the system cost. Thus, SC is very suitable for NN.

Previous works [16]–[18] proposed novel stochastic implementations of NNs. In this case study, we applied the proposed shared WBG design to one of the classical NNs called *restricted Boltzmann machine (RBM)* [19]. Specifically, we considered an RBM used for handwritten digit recognition [16]. The test input is MNIST handwritten digit image dataset [20]. One input image has 28×28 pixels. Thus, the number of inputs in the neural network is 784. The neural network consists of 2 hidden layers $L1$ and $L2$, with 500 and 1000 neurons, respectively. Its output layer has 10 neurons.

Our study is based on the stochastic implementation of the RBM proposed in [16]. Notably, in that design, it uses a circuit similar to the one shown in Fig. 4 to calculate the weighted sum. Thus, our proposed technique can be applied to reduce the area and power of the SNGs used in that design. Note that this application of the proposed technique corresponds to situation 1 mentioned in Section III-C.

TABLE III: The recognition error rate comparison between two stochastic implementations of RBM with different SNGs.

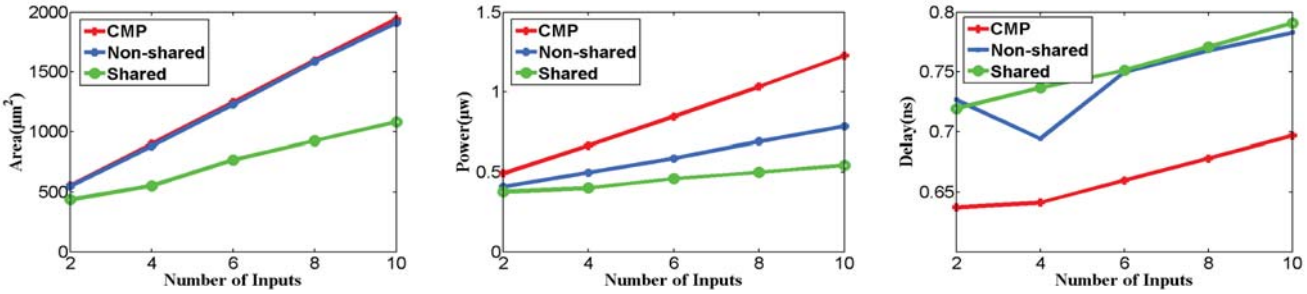
Bit stream length	32	64	128	256	512
Shared WBG	3.14%	1.75%	1.48%	1.52%	1.47%
CMP	2.63%	1.49%	1.53%	1.53%	1.45%

The recognition error rates for different bit stream lengths and PCCs are shown in Table III. The error rate of the traditional binary computing for the same network is 1.23%. From the table, we can see that as the bit stream length grows, the error rates of the stochastic implementations of the RBM reduce. When the bit stream length is 512, the error rate is 1.47% using the shared WBG design and 1.45% using the CMP design, which is very close to that of the binary computing. The accuracy of the shared WBG design is slightly worse than that of the CMP design. However, the difference between them becomes smaller as the bit stream length grows.

TABLE IV: Hardware cost comparison of various implementations of the RBM.

Circuits	CMP	Non-shared	Shared	Conventional Binary
Area(mm^2)	2.24	1.86	1.36	29.15
Power(mW)	759	560	394	14894
Energy(nJ)	36.46	26.89	18.93	29.79

Finally, we compare the hardware cost of different implementations of the RBM, including the binary implementation. The results are listed in Table IV. The 2nd, 3rd, and 4th columns show the results of the stochastic implementations



(a) Area comparison. (b) Power comparison. (c) Delay comparison.

Figure 7: Area, power, and delay comparison among three SNG designs for different numbers of inputs.

using CMP, non-shared WBG, and the proposed shared WBG, respectively. The last column shows the result of the conventional binary implementation. The stochastic bit stream length is 32 and the bit width of the binary computing is 5. As we can see from the table, the area and the power of the binary computing is much larger than the stochastic implementations. Among all the stochastic implementations, the area/power/energy of the proposed shared WBG design is the smallest. Compared to the non-shared WBG design, the proposed design can save 26.9% area ,29.6% power, and 29.6% energy. Compared to the binary implementation, the proposed design can save 36.5% energy with an accuracy loss of only 1.9%.

V. Conclusion

In this work, we studied the problem of how to reduce the cost of SNG in SC. Different from the existing works, which focus on simplifying the RNS inside the SNG, we studied how to reduce the cost of the PCC. We proposed a method to reduce the cost of a type of PCC called WBG by decomposing it into the sharable and the non-sharable parts. The propose design is applicable to generating multiple input bit streams that can be correlated, which indeed is a property of several applications such as image processing, digital filter design, and artificial neural networks. Moreover, for the first time, we give the theoretical cost limit of SNGs. We show that the proposed WBG sharing method can achieve this theoretical limit. Finally, we demonstrated through experiments the effectiveness of the proposed design. In our future work, we will study how to formally prove the conjecture on the theoretical cost limit of SNGs.

Acknowledgment

This work is supported in part by National Natural Science Foundation of China (NSFC) under grant no. 61472243 and 61204042 and in part by National Science Foundation (NSF) of U.S.A. under grant no. CCF-1408123 and CCF-1438286. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSFC and NSF.

References

[1] B. R. Gaines, "Stochastic computing," in *AFIPS Spring Joint Computer Conference*, 1967, pp. 149–156.

[2] A. Alaghi, W. Qian, and J. P. Hayes, "The promise and challenge of stochastic computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (in press)*, 2017.

[3] P. Li, D. J. Lilja *et al.*, "Computation on stochastic bit streams digital image processing case studies," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 3, pp. 449–462, 2014.

[4] V. C. Gaudet and A. C. Rapley, "Iterative decoding using stochastic computation," *Electronics Letters*, vol. 39, no. 3, pp. 299–301, 2003.

[5] B. Li, M. H. Najafi, and D. J. Lilja, "Using stochastic computing to reduce the hardware requirements for a restricted boltzmann machine classifier," in *International Symposium on Field-Programmable Gate Arrays*, 2016, pp. 36–41.

[6] B. Yuan, Y. Wang, and Z. Wang, "Area-efficient scaling-free DFT/FFT design using stochastic computing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 12, pp. 1131–1135, 2016.

[7] T.-H. Chen and J. P. Hayes, "Analyzing and controlling accuracy in stochastic circuits," in *International Conference on Computer Design*, 2014, pp. 367–373.

[8] M. Yang, J. P. Hayes *et al.*, "Design of accurate stochastic number generators with noisy emerging devices for stochastic computing," in *International Conference on Computer-Aided Design*, 2017, pp. 638–644.

[9] P. Knag, W. Lu, and Z. Zhang, "A native stochastic computing architecture enabled by memristors," *IEEE Transactions on Nanotechnology*, vol. 13, no. 2, pp. 283–293, 2014.

[10] R. Venkatesan, S. Venkataramani *et al.*, "Spintastic: Spin-based stochastic logic for energy-efficient computing," in *Design, Automation & Test in Europe*, 2015, pp. 1575–1578.

[11] A. Alaghi and J. P. Hayes, "Dimension reduction in statistical simulation of digital circuits," in *Symposium on Theory of Modeling and Simulation*, 2015, pp. 1–8.

[12] H. Ichihara, T. Sugino *et al.*, "Compact and accurate digital filters based on stochastic computing," *IEEE Transactions on Emerging Topics in Computing*, 2016.

[13] P. K. Gupta and R. Kumaresan, "Binary multiplication with PN sequences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 4, pp. 603–606, 1988.

[14] P.-S. Ting and J. P. Hayes, "Stochastic logic realization of matrix operations," in *Euromicro Conference on Digital System Design*, 2014, pp. 356–364.

[15] D.-U. Lee, R. C. Cheung, and J. D. 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.

[16] B. Li, Y. Qin *et al.*, "Neural network classifiers using stochastic computing with a hardware-oriented approximate activation function," in *International Conference on Computer Design*, 2017, pp. 97–104.

[17] K. Kim, J. Kim *et al.*, "Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks," in *Design Automation Conference*, 2016, pp. 124:1–124:6.

[18] Z. Li, A. Ren *et al.*, "DSCNN: Hardware-oriented optimization for stochastic computing based deep convolutional neural networks," in *International Conference on Computer Design*, 2016, pp. 678–681.

[19] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[20] Y. LeCun, "The MNIST database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.