Area-Efficient Parallel Stochastic Computing with Shared Weighted Binary Generator

Lun Zhang[1], Weikang Qian[2,3], Hai-Bao Chen[1]
[1]Department of Micro and Nano Electronics, Shanghai Jiao Tong University, Shanghai, China
[2]University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, Shanghai, China
[3]MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University, Shanghai, China

***Abstract*—Stochastic computing (SC), a computing paradigm with strong fault tolerance, draws more attention recently. However, a drawback of SC is its long latency. One straightforward way to reduce the latency is to use parallel bit streams to encode data and apply multiple copies of the circuit to process the bit streams. However, this method multiplies the circuit area. In this work, we propose a method to share the weighted binary generator-based stochastic number generator to reduce the area of the parallel implementation of the stochastic circuit. Our experimental results showed that the accuracy of our design is slightly better than the traditional comparator-based design. Besides, our design can reduce the area-latency product by 58.5% compared to the serial implementation.**

I. Introduction

Stochastic computing (SC) [1], first proposed in 1960s, is an unconventional computing paradigm different from the binary computing. It implements arithmetic computation through applying logical operations on stochastic bit streams, which represent values through the probability of 1s in the bit streams [2]. For example, 11000010 represents the value $3/8$. The special encoding mechanism renders strong fault tolerance to SC. With SC, some complex arithmetic functions can be implemented by simple circuits As shown in Fig. 1, multiplication of two values in the range $[0, 1]$ can be realized by a simple AND gate. Because of strong fault tolerance and low hardware cost, SC has been applied in various fields, such as image processing [3], digital filters [4], and artificial neural network [5].

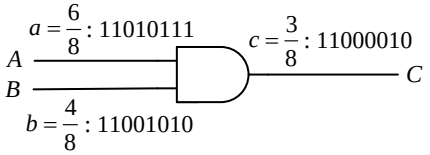

Fig. 1: SC multiplication using a single AND gate.

The circuit operating on stochastic bit streams is called *SC core*. Besides SC core, an essential component of SC is *stochastic numbers generator (SNG)*, which converts input binary numbers into stochastic bit streams. Compared to the SC core, the SNG has a much larger area.

This work is supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61604095 and 61472243, and in part by a 985 research fund from Shanghai Jiao Tong University. Corresponding author: Weikang Qian (Email: qianwk@sjtu.edu.cn) and Hai-Bao Chen (Email: haibaochen@sjtu.edu.cn).

One challenge SC faces is long latency. To reach a precision m of a binary number, the length of a stochastic bit stream should at least be 2^m. In a serial implementation, since the circuit handles one bit in the bit stream per clock cycle, the latency equals the circuit delay times the bit stream length. Thus, serial SC implementation has a large latency that grows exponentially with the precision.

One straightforward way to reduce the latency is to adopt parallel implementation [6]. In this method, a bit stream of length L is replaced by P bit streams of length L/P. Instead of using one SC circuit, P copies of the circuit are used to handle these bit streams in parallel. However, this method achieves latency reduction at the cost of multiplying the circuit area. Particularly, this method multiplies the area of the SNG.

Previous studies showed that an SNG can be decomposed into two modules, a random number source (RNS) and a probability conversion circuit (PCC), where the RNS provides random bits and the PCC converts these bits into a bit stream of a specified probability [7]. For serial SC implementation, techniques to reduce the SNG cost by sharing RNS and PCC were proposed. In [8] and [9], two approaches were proposed to get bit streams with low correlation by shuffling the output of a single RNS. In [10], a solution to reduce the cost of PCCs after RNSs being shared was proposed. It uses a type of PCC called weighted binary generator (WBG) [11]. For parallel implementation, the work [6] proposed a method to share the SNGs. It also uses WBG as the PCC. However, the common module shared among different SNGs is a large memory array, which causes a high area cost for the design. In this work, we propose an area-efficient WBG sharing mechanism for parallel stochastic computing.

In summary, the main contributions of this work are:

1) We propose an area-efficient WBG-based parallel SC implementation by properly sharing the RNSs and the WBGs. Our method can be applied to SC circuit with an arbitrary number of inputs and an arbitrary number of parallel copies.
2) We analyze the theoretical area-latency product of the proposed parallel implementation and compare it with the serial counterpart.
3) We synthesize the circuit and obtain its area and latency. The results show that our design achieves 58.5% reduction over the serial counterpart in terms of area-latency product.

II. Weighted Binary Generator

Weighted binary generator (WBG) is a PCC proposed in [11]. The circuit structure for a target probability of 4-bit precision is depicted in Fig. 2. It can be divided into two parts, WBG part 1 and part 2 [10]. The WBG part 1 takes random bits $r_3, \ldots, r_0$, each with probability of 0.5 to be a 1, as inputs and applies AND gates to them to get the signals w_i as

$$w_3 = r_3, w_2 = \overline{r_3} \wedge r_2, w_1 = \overline{r_3} \wedge \overline{r_2} \wedge r_1,$$
$$w_0 = \overline{r_3} \wedge \overline{r_2} \wedge \overline{r_1} \wedge r_0.$$

where $\wedge$ represents the logical AND. The WBG part 2 is composed of AND gates and a tree of OR gates. The AND gates take $w_3, \ldots, w_0$ and the bits $c_3, \ldots, c_0$ for specifying the target probability as inputs and get $u_i = w_i \wedge c_i$ $(i = 0, \ldots, 3)$. The OR gate tree finally produces the output $x = u_3 \vee \ldots \vee u_0$, where $\vee$ represents the logical OR.

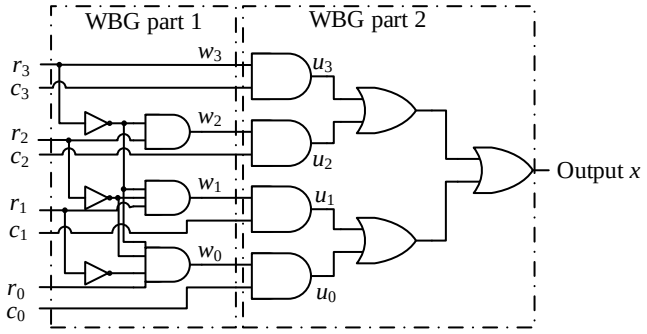

Fig. 2: A weighted binary generator for a target probability of 4-bit precision.

It can be shown that the output probability of the WBG is

$$P(x = 1) = \frac{1}{2}c_3 + \frac{1}{2^2}c_2 + \frac{1}{2^3}c_1 + \frac{1}{2^4}c_0 = \frac{C}{2^4},$$

where $C = (c_3, \ldots, c_0)_2$, the binary number encoded by $c_3, \ldots, c_0$.

The previous work [10] shows that for an SC circuit with SNGs based on WBGs, if some inputs could be correlated, then we could share both the RNS and WBG part 1 to generate these inputs.

III. The Proposed Design

A straightforward parallel SC implementation is shown in Fig. 3 using the multiplication $z = xy$ for illustration. WBG is used as the PCC. As it shows, the input x (resp. y) is represented by two parallel bit streams $x^{(1)}$ and $x^{(2)}$ (resp. $y^{(1)}$ and $y^{(2)}$). The length of each parallel bit stream is half of the original serial bit stream.

Although this parallel method reduces the latency by half, the circuit area doubles. Inspired by the previous work [10], we also exploit the sharing of the RNS and WBG part 1 to reduce the area cost of SNGs. The design is shown in Fig. 4. The SNG part contains two RNSs, two WBG part 1s, and four WBG part 2s. We denote the set of w signals produced by the ith $(i = 1, 2)$ pair of RNS and WBG part 1 as $\overrightarrow{w_i}$. We denote the set of binary inputs to the WBG that specify the target

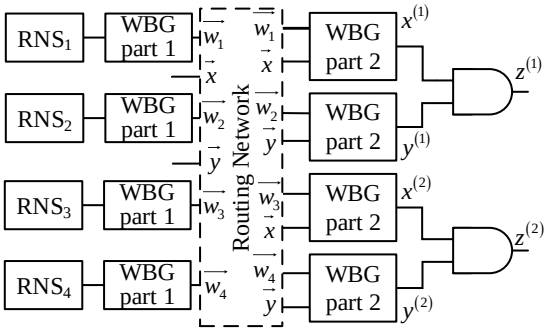

Fig. 3: A straightforward parallel stochastic computing implementation based on WBG.

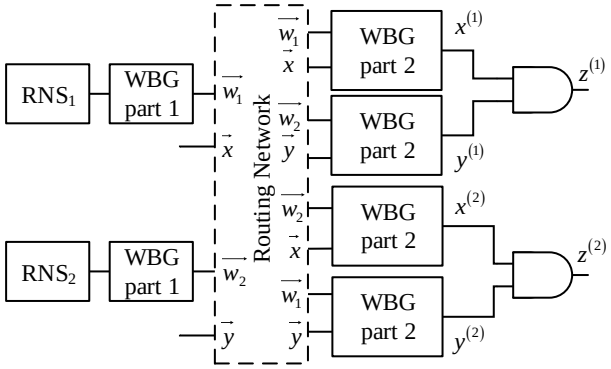

Fig. 4: The proposed parallel SC design with shared WBG for two independent inputs.

probability x (resp. y) as $\overrightarrow{x}$ (resp. $\overrightarrow{y}$). For the top AND gate, its two inputs are produced by two WBG part 2s. The first WBG part 2 has $\overrightarrow{w_1}$ and $\overrightarrow{x}$ as inputs and produces the bit stream $x^{(1)}$, while the second has $\overrightarrow{w_2}$ and $\overrightarrow{y}$ as inputs and produces the bit stream $y^{(1)}$. For the bottom AND gate, its two inputs are produced by the two remaining WBG part 2s. The first WBG part 2 has $\overrightarrow{w_2}$ and $\overrightarrow{x}$ as inputs and produces the bit stream $x^{(2)}$, while the second has $\overrightarrow{w_1}$ and $\overrightarrow{y}$ as inputs and produces the bit stream $y^{(2)}$. This design has the following three important features.

1) The two input bit streams for each AND gates are produced by two different RNSs. Thus, they are independent and the multiplication is correctly realized.
2) The parallel bit streams $x^{(1)}$ and $x^{(2)}$ (resp. $y^{(1)}$ and $y^{(2)}$) are generated by two different RNSs, which ensures that $x^{(1)}$ and $x^{(2)}$ (resp. $y^{(1)}$ and $y^{(2)}$) are not identical. Otherwise, if they are identical, the effective precision encoded by the two parallel bit streams is reduced by half and it is meaningless to have such parallel bit streams.
3) It saves two RNSs and two WBG part 1s compared to the straightforward parallel implementation shown in Fig. 3.

The above design methodology can also be extended to general situation where the SC circuit has N independent inputs and each input is represented by P parallel bit streams. Assume the N inputs are $x_1, \ldots, x_N$ and their corresponding sets of binary inputs are $\overrightarrow{x_1}, \ldots, \overrightarrow{x_N}$. For the general situation, the parallel design has P copies of the SC core. The total

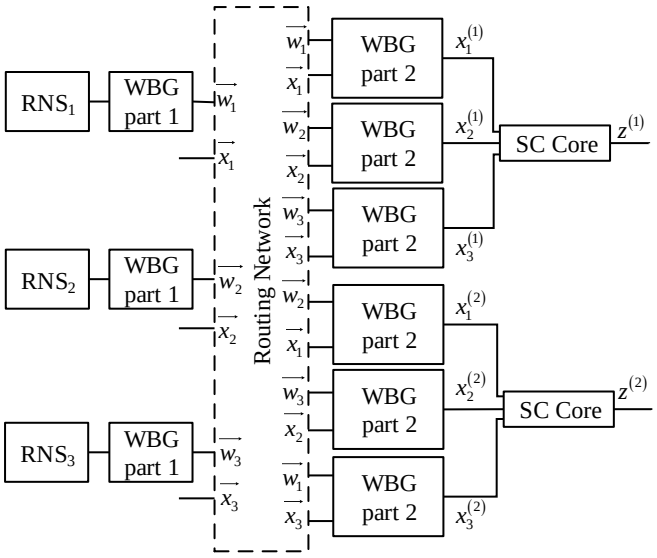

Fig. 5: The proposed parallel SC design with shared WBG for three independent inputs and two parallel copies.

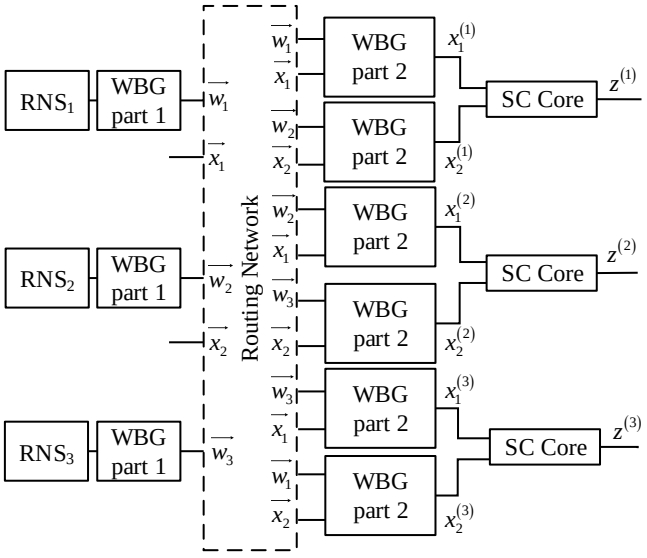

Fig. 6: The proposed parallel SC design with shared WBG for two independent inputs and three parallel copies.

number of inputs to all the SC cores is NP. Thus, it needs NP WBG part 2s. The number of pairs of RNS and WBG part 1 needed depends on which one of N and P is larger.

1) When $N \geq P$, the design needs N pairs of RNS and WBG part 1. This is because we need N pairs of RNS and WBG part 1 to provide N different input streams to each copy of the SC core. Denote the N groups of w signals provided by these N pairs as $\overrightarrow{w_1}, \ldots, \overrightarrow{w_N}$. For the ith ($1 \leq i \leq P$) copy of the SC core, we let the N pairs of inputs to N WBG part 2s for that copy be $(\overrightarrow{w_{f(i)}}, \overrightarrow{x_1}), (\overrightarrow{w_{f(i+1)}}, \overrightarrow{x_2}), \ldots, (\overrightarrow{w_{f(i+N-1)}}, \overrightarrow{x_N})$, where $f(k) = ((k-1) \mod N) + 1$. An example for three independent inputs and two parallel copies is shown in Fig. 5.
2) When $N < P$, the design needs P pairs of RNS and WBG part 1. For the ith ($1 \leq i \leq P$) copy of the SC core, we let the N pairs of inputs to N WBG part 2s for that copy be $(\overrightarrow{w_{f(i)}}, \overrightarrow{x_1}), (\overrightarrow{w_{f(i+1)}}, \overrightarrow{x_2}), \ldots, (\overrightarrow{w_{f(i+N-1)}}, \overrightarrow{x_N})$, where $f(k) = ((k-1) \mod P)+1$. An example for two independent inputs and three parallel copies is shown in Fig. 6.

IV. Analysis of the Area-Latency Product

In this section, we will analyze the area-latency product to show the advantages of our design. We denote the areas of the RNS, the WBG part 1, the WBG part 2, and the SC core as A_R, A_{W1}, A_{W2}, and A_C, respectively. We denote the circuit delay as d. We first take an SC circuit with 2 independent inputs as an example. For the serial implementation based on WBG, assume the bit stream length is L. It needs 2 RNSs, 2 WBG part 1s, 2 WBG part 2s, and 1 SC core. Thus, its area-latency product is

$$(2A_R + 2A_{W1} + 2A_{W2} + A_C)Ld.$$

For our proposed design with 2 parallel copies, as shown in Fig. 4, it needs 2 RNSs, 2 WBG part 1s, 4 WBG part 2s, and 2 SC cores. However, the bit stream length is reduced to $L/2$. Thus, the area-latency product of our design is

$$\begin{aligned}&(2A_R + 2A_{W1} + 4A_{W2} + 2A_C)\frac{L}{2}d\\ =&(A_R + A_{W1} + 2A_{W2} + A_C)Ld.\end{aligned}$$

Thus, our design reduces the area-latency product over the serial design by $(A_R + A_{W1})Ld$.

For a general stochastic circuit with N independent inputs, its serial implementation needs N SNGs. Its area-latency product is

$$(NA_R + NA_{W1} + NA_{W2} + A_C)Ld. \quad (1)$$

For the proposed implementation with P parallel copies, the area-latency product can be obtained as

$$\begin{aligned}&(MA_R + MA_{W1} + NPA_{W2} + PA_C)\frac{L}{P}d\\ =&\left(\frac{M}{P}A_R + \frac{M}{P}A_{W1} + NA_{W2} + A_C\right)Ld,\end{aligned}$$

where $M = \max\{N, P\}$. Compared with Eq. (1), the reduction of the area-latency product is $(N - M/P)(A_R + A_{W1})Ld$.

V. Experimental Results

In this section, we experimentally study the accuracy and the hardware cost of the proposed design. Three different SC designs are considered for comparison. The first one is a serial implementation using the widely-used comparators as the PCCs. The second one is also a serial implementation, but it uses the WBGs as the PCCs. The last one is our proposed parallel implementation with WBG sharing. For simplicity, we refer to these three designs as the *serial CMP*, the *serial WBG*, and the *parallel WBG*, respectively. 8-bit LFSRs were used as the RNSs. In the experiments, we first evaluated the accuracy of the SC circuits. Then, we synthesized the circuits and obtained their hardware costs.

A. Accuracy Comparison

In this section, we compare the accuracy of the three designs. We used the multiplication $z = xy$ as the target function. For the parallel WBG, we chose the number of parallel copies as 2. We tested on 5 different bit stream lengths $16, 32, 64, 128, 256$. For each length, we performed the simulation 100 times and for each simulation, we randomly chose the LFSR seed and the input probabilities. We calculated the mean absolute error (MAE) over the 100 simulations for each length. Fig. 7 plots how MAE changes with bit stream length for the three designs. From the figure, we can see that our design has the smallest MAE among the three designs.

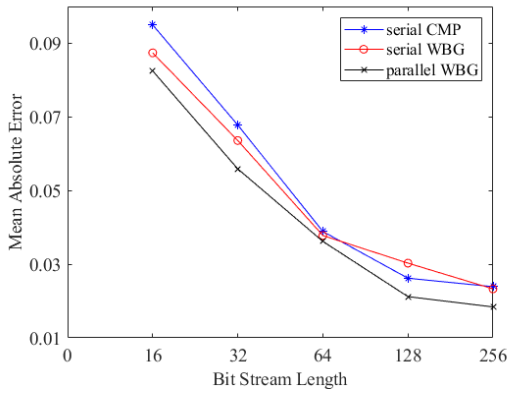

Fig. 7: Mean absolute error comparison for three different SC designs implementing $z = xy$ over different bit stream lengths.

B. Hardware Cost Comparison

In this section, we compare the area and the latency of the three designs. The latency equals the circuit delay times the bit stream length. We set the bit stream length for the serial implementations as 256. We focused on the SNG part and ignored the SC core and the *derandomizer*, which further converts the bit stream representation back into the binary representation. The technology library that we used is SMIC 40nm library [12].

We first considered implementing a function with 2 independent inputs. We assumed that the parallel WBG has 2 parallel copies. The experimental results for the three designs are listed in Table I. Both the area and the latency of the serial CMP are the largest. The latency of the parallel WBG reduces by 38.3% over the serial WBG although its area increases by 8.3%. In terms of the area-latency product, the parallel WBG achieves 51.9% reduction over the serial CMP and 33.1% reduction over the serial WBG. It should be noted that the area of the serial CMP design is even larger than that of the parallel WBG. This is because the area of one comparator is larger than the total area of one WBG part 1 and two WBG part 2s.

We also compared the hardware cost of the three designs for a function with 3 independent inputs. For the parallel WBG, we set its number of parallel copies as 3. The results are listed in Table II. From the table, we can see that the area-latency product of our design achieves 68.2% and 58.5% reduction over those of the serial CMP and the serial WBG, respectively. Compared to the 2-input case, the reduction ratio increases.

TABLE I: Hardware cost comparison of three SC designs for a function with 2 independent inputs.

Design	serial CMP	sreial WBG	parallel WBG
Area (μm^2)	264	240	260
Latency (ns)	182.24	144.16	**88.96**
Area-latency product ($\mu m^2 \cdot ns$)	48111.4	34598.4	**23129.6**

TABLE II: Hardware cost comparison of three SC designs for a function with 3 independent inputs.

Design	serial CMP	serial WBG	parallel WBG
Area (μm^2)	438	423	453
Latency (ns)	177.6	140.96	**54.56**
Area-latency product ($\mu m^2 \cdot ns$)	77788.8	59626.1	**24715.7**

VI. Conclusion

In this paper, we considered parallel implementation of SC circuit. It could reduce the latency of an SC circuit but at the cost of a large area. We proposed a method to reduce the area by using WBGs as the PCCs and sharing the RNSs and the WBGs. We showed the design methodology for SC circuit with an arbitrary number of inputs and an arbitrary number of parallel copies. Our experimental results showed that our proposed parallel implementation could reduce the area-latency product by 58.5% over the serial implementation. In our future work, we will study how to efficiently implement the derandomizer for the parallel implementation.

References

[1] B. R. Gaines, "Stochastic computing systems," in *Advances in information systems science*. Springer, 1969, pp. 37–172.

[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*, vol. 37, no. 8, pp. 1515–1531, 2018.

[3] A. Alaghi, C. Li, and J. P. Hayes, "Stochastic circuits for real-time image-processing applications," in *Design Automation Conference*, 2013, pp. 136:1–136:6.

[4] Y.-N. Chang and K. K. Parhi, "Architectures for digital filters using stochastic computing," in *International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 2697–2701.

[5] 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.

[6] L. Miao and C. Chakrabarti, "A parallel stochastic computing system with improved accuracy," in *International Workshop on Signal Processing Systems*, 2013, pp. 195–200.

[7] 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.

[8] H. Ichihara, T. Sugino *et al.*, "Compact and accurate digital filters based on stochastic computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 7, no. 1, pp. 31–43, 2019.

[9] 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.

[10] M. Yang, B. Li *et al.*, "Towards theoretical cost limit of stochastic number generators for stochastic computing," in *IEEE Computer Society Annual Symposium on VLSI*, 2018, pp. 154–159.

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

[12] Semiconductor Manufacturing International Corporation (SMIC), "SMIC 40nm technology," http://www.smics.com/en/site/technology_advanced_Ft, accessed June 20, 2019.