An Energy-Efficient Binary-Interfaced Stochastic Multiplier Using Parallel Datapaths

Yongqiang Zhang, *Member, IEEE*, Siting Liu, Jie Han, *Senior Member, IEEE*, Zhendong Lin, Shaowei Wang, Xin Cheng, and Guangjun Xie

***Abstract*—Stochastic computing (SC) typically requires a low design complexity compared with weighted binary computing, so it has been successfully applied in neural networks (NNs). Usually, SC utilizes random bitstreams as its medium, which makes it suffer from a long delay that offsets its advantages. This drawback can be alleviated by utilizing parallel datapaths, which, however, will significantly increase the hardware cost due to the requirement of multiple parallel computing units. In this paper, a hybrid bit splitting generator (HBSG) is proposed to efficiently produce parallel bitstreams in a single clock cycle to reduce delay. The HBSG uniformly splits binary numbers into *R* segments, each of which is encoded in parallel by using hardwired connections according to the weight of each bit. A binary-interfaced parallel stochastic multiplier (BipSMul) using the HBSG is then proposed to accelerate the multiplication in SC. Experimental results show that the BipSMul is more energy efficient than the state-of-the-art parallel and serial stochastic designs, as well as their binary and Booth counterparts, in delay, power-delay product (PDP), and area-delay product (ADP).**

***Index Terms*—Stochastic computing, parallel datapath, multiplier, energy-efficiency, neural network.**

I. Introduction

STOCHASTIC computing (SC) encodes binary numbers into random bitstreams for fault-tolerant applications [1]. While they suffer from random fluctuations, the bitstreams can be serial or parallel, with all bits of equal weight to counter bit flip induced errors and, thus, achieve a high fault tolerance [2]. Several other errors, such as rounding, constant-induced, and cross-correlation errors, also exist, mainly due to the representation of bitstreams in SC [2]. The main advantages of SC also include simple and compact digital units. For example, multiplication in SC can be realized by an AND gate [1]. Thus, SC is applicable when a slight computing inaccuracy is acceptable in applications, such as neural networks (NNs) consisting of multiply accumulate (MAC) units [3].

However, SC suffers from a serious delay issue because it is usually performed on long serial bitstreams. For example, 2^n clock cycles are required for multiplying two bitstreams with n-bit precision. This delay will be 2^{2n} clock cycles if completely accurate multiplication is expected by using the deterministic approach [4, 5]. The delay can be significantly reduced by using parallel datapaths to reduce it from 2^n to 1. However, few works on this aspect have been reported because the hardware cost required by the computing units also exponentially increases. For example, 256 AND gates are needed for multiplying two numbers with 8-bit precision for parallel processing, while only one is needed for serial processing. The thermometer coding-based parallel bitstream generator has been designed for computing in one clock cycle, thereby saving power [6]. However, its hardware cost is still very high.

To reduce delay and save energy, a hybrid bit splitting generator (HBSG) is proposed in this paper for multipliers. Binary numbers are split into *R* segments with equal bit width before converting them into bitstreams. Then, the delay is reduced to one clock cycle by encoding each segment in parallel. The coding scheme uses hardwired connections according to the weight of each bit in each segment. A binary-interfaced parallel stochastic multiplier (BipSMul) using the HBSG is then proposed to speed up the multiplication in SC. The deterministic approach is applied to the BipSMul for improving the computing accuracy. Although the deterministic approach will require an increased number of AND gates, its hardware cost is kept low because of the utilization of the bit splitting and hardwire connection methods.

The main contributions of this work are summarized as follows. 1) An HBSG is proposed for converting binary numbers into parallel bitstreams in only one clock cycle. 2) A BipSMul using the HBSG is developed to reduce delay and energy. 3) An NN using the BipSMul is quantized to illustrate the proposed designs.

This paper proceeds as follows. Section II reviews the concepts of SC. Section III presents the designs of HBSG and

This work was supported by the Fundamental Research Funds for the Central Universities of China (Grant No. JZ2020HGQA0162, and PA2021KCPY0043), by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Number: RES0048688), and by Natural Science Foundation of Anhui Province (Grant No. 2108085MF226).

(*Corresponding author: Guangjun Xie*)

Y. Zhang, Z. Lin, S. Wang, X. Cheng and G. Xie are with the School of Microelectronics, Hefei University of Technology, Hefei 230009, China (e-mail: ahzhangyq@hfut.edu.cn; zdlin@mail.hfut.edu.cn; 2019010121@mail.hfut.edu.cn; xcheng@hfut.edu.cn; gjxie8005@hfut.edu.cn)

S. Liu is with the School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, and Shanghai Engineering Research Center of Energy Efficient and Custom AI IC, Shanghai 201210, China (e-mail: liust@shanghaitech.edu.cn)

J. Han is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada (e-mail: jhan8@ualberta.ca)

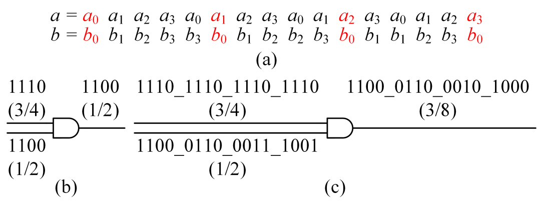
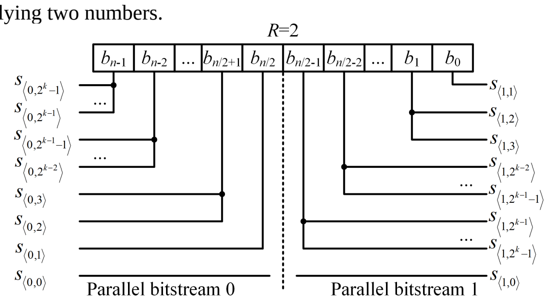


Fig. 1. (a) An illustration of the deterministic approach. (b) A traditional approach for multiplying two numbers. (c) The deterministic approach for multiplying two numbers.



R=2
b_{n-1} b_{n-2} ... $b_{n/2+1}$ $b_{n/2}$ $b_{n/2-1}$ $b_{n/2-2}$ ... b_1 b_0
$S_{\langle 0,2^k-1\rangle}$
$S_{\langle 0,2^{k-1}\rangle}$
$S_{\langle 0,2^{k-1}-1\rangle}$
$S_{\langle 0,2^{k-2}\rangle}$
$S_{\langle 0,3\rangle}$
$S_{\langle 0,2\rangle}$
$S_{\langle 0,1\rangle}$
$S_{\langle 0,0\rangle}$
$S_{\langle 1,1\rangle}$
$S_{\langle 1,2\rangle}$
$S_{\langle 1,3\rangle}$
$S_{\langle 1,2^{k-2}\rangle}$
$S_{\langle 1,2^{k-1}-1\rangle}$
$S_{\langle 1,2^{k-1}\rangle}$
$S_{\langle 1,2^k-1\rangle}$
$S_{\langle 1,0\rangle}$
Parallel bitstream 0
Parallel bitstream 1

Fig. 2. The proposed HBSG.

BipSMul. Section IV discusses the experimental results of the proposed multiplier. An application of an NN is shown in Section V. Section VI concludes this work.

II. Basic Concepts

A. Bitstream Generators

Random bitstreams in SC are generated by stochastic number generators (SNGs). A typical SNG mainly consists of a linear feedback shift register (LFSR) and a comparator for generating serial bitstreams [7]. It produces a 1 (or 0) if the binary number in the LFSR is smaller (or larger) than the input binary number in each clock cycle. If the LFSR and the binary number are n-bit wide, the generated bitstream length (BSL) will be 2^n, or 2^n clock cycles are required for generating bitstreams with an n-bit precision. This incurs a long delay and leads to significant energy consumption.

The thermometer coding based SNG produces parallel bitstreams to reduce the delay from 2^n to one clock cycle [8]. That is, for an n-bit binary number, 2^n parallel outputs are simultaneously generated with this SNG, making use of a large number of logic gates.

Compared with the two methods above, a natural method is to generate bitstreams according to the weight of each bit in a binary number. This method has been employed to build an n-to-2^n-1 SNG by using hardwired connections to generate parallel bitstreams in one clock cycle [9]. However, the bitstreams have a very high correlation, which has been reduced through a four-stage omega-flip network, leading to a large hardware cost and energy dissipation [9].

B. Deterministic Approaches

Typically, SC suffers from inherent random fluctuations, which make it inaccurate. The deterministic approaches have been developed to alleviate this issue [10]. Similar to the convolution in mathematics, each bit in one bitstream interacts with all bits in the other one so that the generated outputs are accurate. For example, when multiplying two bitstreams a and b, b_0 interacts with a_0, a_1, a_2, and a_3, as illustrated in Fig. 1(a).

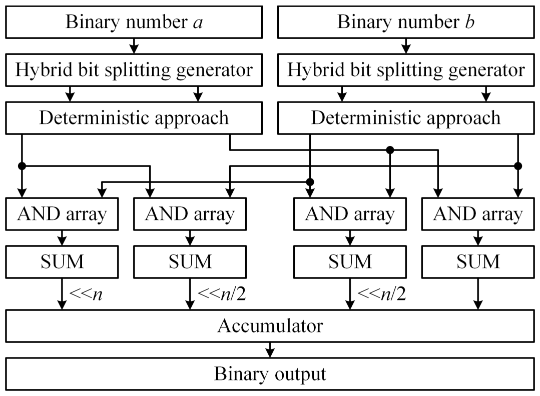


Fig. 3. The proposed BipSMul (R=2).

Fig. 1(b) and (c) illustrate the comparison between the traditional and deterministic approaches when multiplying two 2-bit binary numbers. In Fig. 1(c), 1110 (encoding 3/4) is repeated 4 times and 1100 (encoding 1/2) is rotated 4 times. It is shown that the result generated using the deterministic approach is completely accurate, whereas the traditional one produces inaccurate results. However, the bitstream length in the deterministic approach exponentially increases as the precision increases, resulting in a much longer delay.

III. The Proposed Designs

A. A Hybrid Bit Splitting Generator (HBSG)

The advantages of parallel designs will be completely offset if a higher precision is required because of the huge hardware resources. Hence, simplifying the complexity of parallel designs becomes imperative.

Bit splitting is a common method to lower the hardware costs of digital arithmetic circuits, such as approximate adders [11]. This method splits binary numbers into several segments and then processes each segment according to certain rules. It has also been employed for accelerating serial SC [12]. However, a high delay remains, which leads to a low energy efficiency. It is noteworthy that the bit splitting method can be exploited much further for reducing the complexity of parallel designs to lower delay and energy.

With this in mind, an HBSG is proposed for accelerating parallel designs by means of bit splitting in this work. A given n-bit binary number $b=b_{n-1}b_{n-2}\ldots b_1b_0$ is split into R segments with equal bit width. Each segment is then respectively converted into a parallel bitstream by using hardwired connections according to the weight of each bit in the segment, as shown in Fig. 2, where R is set to 2 for simplicity and k equals n/R as the bit width of each segment. In addition, a zero is added to each generated bitstream to make the probability of 1s in the bitstream proportional to the value of the segment divided by 2^k, as the insertion of an all-zero state to a linear feedback shift register [13]. Therefore, the bitstream lengths of two parallel bitstreams $S_{<1,>}$ and $S_{<0,>}$ are both 2^k in Fig. 2, so the subscripts of the bitstreams range from 0 to 2^k-1. For example, the number of connections of the i-th bit (i=1, 2, 3, …, n/2) in $b_{n-1}b_{n-2}\ldots b_{n/2+1}b_{n/2}$ is 2^{i-1}, so the subscripts of the bitstreams range from 2^{i-1} to 2^i-1. Note that just one clock cycle is required for generating multiple parallel bitstreams.

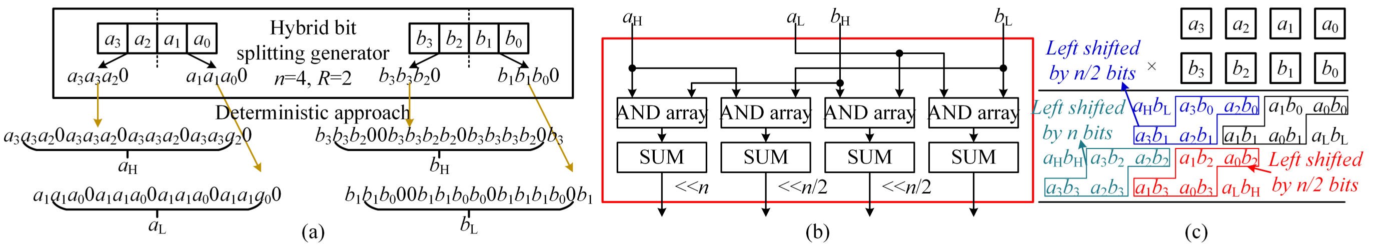


Fig. 4. An example of multiplying two 4-bit binary numbers by using the proposed BpSMul (R=2). (a) Bit splitting and bitstream generation. (b) Bitstream multiplication and accumulation. (c) Illustration of left-shifting.

B. A Binary-Interfaced Parallel Stochastic Multiplier (BipSMul)

A BipSMul is proposed by using the HBSG, as illustrated in Fig. 3, where R is set to 2 for convenience in description, "$<<n$" means a left-shift by n bits. Two n-bit binary numbers a and b are processed through the HBSG to generate 4 parallel bitstreams with a length of $2^{n/2}$ bits. These bitstreams are then handled by using the deterministic approach, as illustrated in Fig. 1(a), to generate 4 deterministic bitstreams with a length of 2^n bits. They are then paired into four AND arrays to perform multiplication. Four parallel counters indicated as "SUM" convert the bitstreams generated by AND arrays into binary numbers. Finally, the converted numbers are left-shifted by different numbers of bits and fed into an accumulator to obtain the final binary result.

Unipolar and bipolar formats are two encoding methods in SC, in which numbers range in [0,1] and [-1,+1], respectively. For example, the bitstream 1100 encodes 1/2 in the unipolar format, whereas it encodes 0 in the bipolar one. The unipolar format is used in the proposed BipSMul. As for signed multiplication, one more sign bit is added to the bitstreams. The product is determined by the sign bit, which can be obtained through an XOR gate.

C. An Example of Multiplying Two Binary Numbers

Fig. 4 shows an example of multiplying two 4-bit binary numbers (a=$a_3a_2a_1a_0$ and b=$b_3b_2b_1b_0$) to illustrate the principle of the proposed BipSMul.

At first, each number is split into two segments, i.e., a_3a_2 and a_1a_0, b_3b_2 and b_1b_0. These segments are respectively converted to 4 parallel bitstreams, including $a_3a_3a_2$0 and $a_1a_1a_0$0, $b_3b_3b_2$0 and $b_1b_1b_0$0, according to the weight of each bit in each segment. For example, the weight of a_3 in a_3a_2 is 2, so a_3 is converted to two bits, while a_2 is converted to 1 bit. The length of these bitstreams is $2^{n/2}$=$2^{4/2}$=4. This process is accomplished by using the proposed HBSG. These 4 bitstreams are then composed by the deterministic approach, i.e., copying $a_3a_3a_2$0 and $a_1a_1a_0$0 four times and rotating $b_3b_3b_2$0 and $b_1b_1b_0$0 four times, respectively, to generate 4 deterministic bitstreams, denoted as a_H, a_L, b_H, and b_L. The length of 4 deterministic bitstreams is $(2^{n/2})^2$=$(2^{4/2})^2$=16, as illustrated in Fig. 4(a).

a_H, a_L, b_H, and b_L are paired into four AND arrays, each of which contains 16 AND gates, as shown in Fig. 4(b). The results are then summed and left-shifted for subsequent usage. Two key points should be noted here, one of which is the theory behind the pairing of 4 bitstreams, while the other is the number of left-shifted bits. The multiplication of two 4-bit binary numbers generates 16 partial products that can be divided into 4 parts surrounded by four boxes, as shown in Fig. 4(c). Each part corresponds to a pairing of two bitstreams; for example, the multiplication of a_3a_2 and b_3b_2 is equivalent to the pairing of a_Hb_H. Thus, the binary multiplication process decides the pairing of parallel bitstreams. In addition, each partial product is one bit, so its length is 1. Thus, a_Hb_H is left-shifted by n=4 bits. The same holds true for other parts. For different values of R and n, the number of left-shifted bits is pre-determined, according to the principle in Fig. 4(c).

IV. Experimental Results

The binary, Booth, serial [5], thermometer [6], and the proposed multipliers are evaluated and they all produce accurate results because of the employed deterministic approach. The binary multipliers are designed in three steps. 1) Generating partial products using AND gates. 2) Compressing partial products using the 4-2 compressor in [14], full adders, and half adders based on the Dadda tree structure. 3) Generating final results using a ripple carry adder. The Booth multipliers are designed using the modified radix-4 method [15], of which the partial products are also compressed by using compressors, full adders, and half adders. All circuits are synthesized by Synopsys Design Compiler with the TSMC 40 nm library. Two commands 'set_target_library_subset' and 'compile_ultra' are used to synthesize the circuits through AND, OR, NOT, XOR, and XNOR gates. This method will avoid the usage of pre-built special modules in the library and keep a balanced comparison. The power is obtained in PrimePower using a vector-free power analysis model.

A. Comparison of Multipliers

Hardware cost: TABLE I lists the area, power, delay (as the critical path delay multiplied by the number of clock cycles), power-delay-product (PDP), area-delay-product (ADP), and energy-delay-product (EDP) versus different R values for 4-, 6-, 8, 16, and 32-bit multipliers (M_{i_j} denotes R=j for the proposed i-bit BipSMul; R=1 means that numbers are not split).

For the 4-bit serial multiplier, its delay is equal to the critical path delay multiplied by $(2^n)^2$=$(2^4)^2$=256 (the power 2 is due to the deterministic approach), which leads to a high energy consumption, PDP, ADP, and EDP. Since half of the elements in the truth table of an n-bit thermometer based SNG with 2^n outputs are 1, the logic expressions of outputs are very complex [8]. These result in an inferior performance of the thermometer based multipliers (shortened as Them), as shown in the simulation results in TABLE I. Among them, the PDPs are reduced by 4.84%, 12.56%, 13.73%, 13.89%, and 29.75% for the 4-, 6-, 8-, 16-, and 32-bit BipSMuls compared to the binary

TABLE I
THE COMPARISON OF 4-, 6-, 8-, 16-, AND 32-BIT BPSMULS VERSUS VARIOUS *R* AND EXISTING MULTIPLIERS

Bit	Design	Area (um^2)	Power (uW)	Delay (ns)	PDP (fJ)	ADP ($um^2 \cdot ns$)	EDP ($fJ \cdot ns$)
4	M_{4_1}	132.12	7.6	1.05	8.03	138.73	8.43
	M_{4_2}	133.18	7.6	1.03	7.86	137.18	8.10
	M_{4_4}	128.42	7.5	1.28	9.64	164.38	12.34
	Binary	135.30	7.7	1.07	8.26	144.77	8.84
	Booth	178.16	9.7	1.41	13.67	251.21	19.28
	Serial [5]	250.66	17.3	269	4656.4	67379	1251646
	Them [6]	1689.56	59.4	3.12	185.34	5271.42	578.26
6	M_{6_1}	318.05	16.7	1.61	26.88	512.06	43.28
	M_{6_2}	315.76	16.1	1.73	27.90	546.26	48.27
	M_{6_3}	344.69	17.2	1.72	29.59	592.86	50.89
	M_{6_6}	312.23	16.6	1.53	25.34	477.71	38.77
	Binary	315.76	16.9	1.71	28.98	539.94	49.56
	Booth	388.26	22.1	1.94	42.96	753.22	83.35
	Them [6]	27783	114.4	5.51	630.45	153082	3473.80
8	M_{8_1}	36755	1323	5.84	7728.7	214652	45135
	M_{8_2}	580.89	28.8	2.10	60.47	1219.86	126.98
	M_{8_4}	639.98	31.1	2.43	75.63	1555.15	183.78
	M_{8_8}	578.59	29.2	2.27	66.35	1313.40	150.62
	Binary	583.53	30.7	2.28	70.09	1330.45	159.80
	Booth	641.39	39.2	2.93	114.74	1879.27	336.18
	Them [6]	412145	12349	8.21	101386	3383711	832380
16	M_{16_2}	552870	17938	8.47	151936	4682805	1286895
	M_{16_4}	2534.3	132.2	5.92	782.62	15003.3	4633.1
	M_{16_8}	2702.1	144.2	5.13	739.75	13861.7	3794.9
	M_{16_16}	2430.3	134.5	4.99	671.16	12127.0	3349.1
	Binary	2384.8	143.8	5.42	779.40	12925.4	4224.3
	Booth	2211.0	162.9	4.87	793.32	10767.6	3863.5
32	M_{32_8}	9047.0	510.9	7.95	4061.7	71923.9	32290
	M_{32_16}	9171.0	529.3	7.64	4043.9	70066.7	30895
	M_{32_32}	8069.9	507.1	7.56	3833.7	61008.8	28983
	Binary	9616.6	665.5	8.20	5457.1	78856.3	44748
	Booth	8170.5	675.3	8.45	5706.3	69040.7	48218

designs, and 42.50%, 41.01%, 47.30%, 15.40%, and 32.82% compared to the Booth designs. Note that, since the used deterministic approach, the bitstream length of M_{8_1} is $(2^8)^2=2^{16}$, so very complex SUM and accumulation in Fig. 3 are consumed, leading to abnormal hardware costs.

Fault-tolerance: To investigate the fault tolerance of these 4-bit multipliers, Monte Carlo simulations are carried out by 100000 times for each probability of error in [0 1] with a step size of $1/2^4$. Errors are injected into each computing element to flip the outputs. This is implemented by an XOR gate. The values of two inputs of a multiplier are then traversed to compute the mean error distance (MED) as follows

$$\mathrm{MED} = \frac{1}{2^{2n}} \sum_{i=1}^{2^{2n}} \left| ED_i \right|, \tag{1}$$

where n is the bit width of a multiplier and ED_i is the arithmetic distance between exact and stochastic results for each input.

As can be seen in the MEDs in Fig. 5, the thermometer, serial, and M_{4_1} multipliers result in similar MEDs, which are slightly smaller than that of the binary design, because these three multipliers generate $(2^4)^2$=256-bit outputs accumulated through the same ripple carry adders. The Booth multiplier shows the highest MEDs, so its fault tolerance is the worst among these multipliers, while the M_{4_4} presents the best results. Although the MEDs of the M_{4_2} are higher than that of the M_{4_4}, the M_{4_2} is still used later, because of its lower hardware costs (in TABLE I).

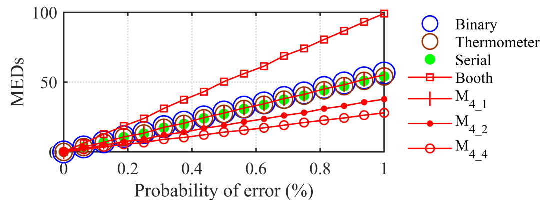


Fig. 5. Mean error distance versus the probability of error for 4-bit multipliers.

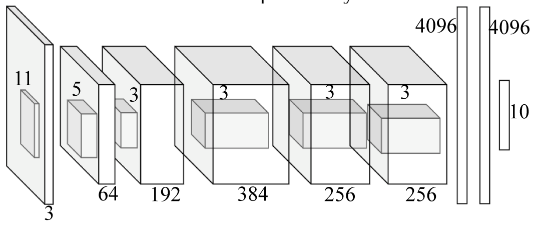


Fig. 6. The AlexNet structure. The first six outer boxes indicate the input data. The corresponding numbers denote the filter sizes and the number of channels for each input to the convolutional layer. The last three rectangles are the output of each fully-connected layers and the number denotes the dimension. The last layer has 10 elements, standing for the 10 classes of the images.

V. APPLICATIONS

Recent machine learning models can compete with humans in many tasks related to recognition and detections thanks to the development of deep convolutional neural network models [16]. A convolutional network, however, usually involves a huge amount of multiply-and-accumulate (MAC) operations. It incurs high energy and delay to both move data between memory and computing units and perform the MAC. Many efforts have been devoted to quantizing the parameters so that the model can be compressed and calculated with reduced energy and delay. Advanced schemes can quantize a single-precision floating-point (FP) model to a 4-bit one, leading to roughly an energy efficiency improvement by 8 times. The proposed design can then be used to execute the 4-bit multiplications in these quantized models by replacing the conventional binary multipliers, further lowering the energy consumption and delay.

Without losing generality, AlexNet [17] is used to illustrate this idea. It consists of 5 convolutional layers for feature extraction and 3 fully connected layers for classification. Fig. 6 shows the structure of an AlexNet.

An AlexNet model contains a lot of parameters, usually referred to as weights. They can be trained or adjusted to perform specific recognition or classification tasks while staying unchanged when making the inference. In the convolutional layers, the weights are grouped as kernels or filters and convolved with the input using multiplications and additions, followed by a Rectified Linear Unit (ReLU) function called activation function, e.g., y=max(0, x), where x and y are the input and output of the ReLU function, respectively [3]. The ReLU function generates a 0 if the input is less than 0, or it outputs the input directly. After that, the results or the number of outputs are optionally compressed through a pooling layer. In the fully connected layers, the output is the ReLU results of the matrix multiplication of the inputs and the weights. The last stage is a softmax unit, used for multi-class classification.

We use the Brevitas package [18] to quantize the parameters and intermediate variables in the network. The straight-through

TABLE II
RECOGNITION ACCURACY OF ALEXNET USING DIFFERENT MULTIPLIERS

AlexNet models	FP baseline	4-bit quantized model using the conventional binary multipliers	4-bit quantized model using the proposed stochastic multipliers
Accuracy	83.5%	80.9%	80.9%

TABLE III
THE HARDWARE RESOURCES OF m-INPUT MACS USING 8-BIT MULTIPLIERS

Designs		2	4	8	16
Binary	Area (um^2)	882.71	1752.00	3486.37	7053.35
	Power (mW)	0.0633	0.1191	0.2298	0.4425
	Delay (ns)	1.87	1.87	1.8700	3.31
	PDP (pJ)	0.12	0.22	0.43	1.46
	ADP ($um^2\cdot ns$)	1650.66	3276.25	6519.51	23346.60
	EDP ($fJ\cdot ns$)	224.4	411.4	804.1	4832.6
Proposed	Area (um^2)	875.30	1740.01	3458.67	6894.24
	Power (mW)	0.0616	0.1161	0.2238	0.4378
	Delay (ns)	1.48	1.46	1.47	1.45
	PDP (pJ)	0.09	0.17	0.33	0.63
	ADP ($um^2\cdot ns$)	1295.44	2540.41	5084.25	9996.65
	EDP ($fJ\cdot ns$)	133.2	248.2	485.1	913.5

estimator [19] is used to perform quantization-aware training and the weights and inputs to each layer are quantized to 4-bit width. The first layer (for the input images) is not quantized in order to maintain a high accuracy. Thus, all the multiplications except the first layer can be implemented by either a 4×4 binary multiplier or the proposed multiplier.

The AlexNet is trained for image recognition on the CIFAR10 dataset. This image recognition task is to classify the color images in the dataset into 10 classes. It contains 50,000 images for training and 10,000 images for testing. The recognition accuracy is compared for a baseline FP AlexNet model, a 4-bit quantized model using a conventional multiplier, and a 4-bit quantized model using the proposed stochastic multiplier. The results are shown in TABLE II.

As can be seen, the recognition accuracy of the quantized models is slightly degraded due to the quantization error compared to the FP model. However, the quantized model using stochastic and binary multipliers shows the same accuracy since the proposed stochastic multiplier produces exactly the same 4×4 computation result as a binary multiplier. In terms of energy, since the 4-bit stochastic multiplier is more efficient than its binary counterpart as shown in TABLE I, we can estimate that using the 4-bit proposed2-type multiplier, energy can be saved by around 2.6 μJ compared to the binary design because there are in total 710M 4×4 multiplications in the quantized NN at a clock frequency of 100 MHz. TABLE III provides the hardware measurements of an m-input MAC (m=2, 4, 8, 16), which also shows the higher energy efficiency of the proposed multiplier.

VI. CONCLUSION

Research on accelerating stochastic computing (SC) based neural networks (NNs) has received extensive attention in recent years. In this paper, a hybrid bit split generator (HBSG) is proposed to reduce the delay in the generation of bitstreams by splitting binary numbers into several segments in advance and then encoding them through hardwired connections. The HBSG is then used for a binary-interfaced parallel stochastic multiplier (BipSMul) design. Synthesized results show that the proposed BipSMul outperforms its stochastic counterparts and conventional binary designs in overall hardware performance. Experimental results for an NN application indicate that with a limited loss of accuracy, the proposed BipSMul indeed reduces the required energy consumption, compared with conventional binary designs.

REFERENCES

[1] A. Alaghi, W. Qian, and J. Hayes, "The promise and challenge of stochastic computing," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 37, no. 8, pp. 1515-1531, Aug. 2018.

[2] W. Gross and V. Gaudet, *Stochastic computing: Techniques and applications*. Springer Nature Switzerland AG: Springer, 2019.

[3] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, "A survey of stochastic computing neural networks for machine learning applications," *IEEE Trans. Neural Netw. Learn. Syst.,* vol. 32, no. 7, pp. 2809-2824, Jul. 2021.

[4] H. Najafi and D. Lilja, "High quality down-sampling for deterministic approaches to stochastic computing," *IEEE Trans. Emerging Top. Comput.*, vol. 9, no. 1, pp. 7-14, Mar. 2018.

[5] Z. Lin, G. Xie, W. Xu, J. Han, and Y. Zhang, "Accelerating stochastic computing using deterministic halton sequences," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 68, no. 10, pp. 3351-3355, Oct. 2021.

[6] Y. Zhang, R. Wang, X. Zhang, Y. Wang, and R. Huang, "Parallel hybrid stochastic-binary-based neural network accelerators," *IEEE Trans. Circuits Syst., II, Exp. Briefs,* vol. 67, no. 12, pp. 3387-3391, Dec. 2020.

[7] S. Shenoi, "A comparative study on methods for stochastic number generation," M.S., College of Engineering and Applied Sciences, University of Cincinnati, Ann Arbor, 2017.

[8] Y. Zhang *et al.*, "A parallel bitstream generator for stochastic computing," presented at the 2019 Silicon Nanoelectronics Workshop, Kyoto, Japan, 9-10 Jun., 2019.

[9] V. Sehwag, N. Prasad, and I. Chakrabarti, "A parallel stochastic number generator with bit permutation networks," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 65, no. 2, pp. 231-235, Feb. 2018.

[10] D. Jenson and M. Riedel, "A deterministic approach to stochastic computation," presented at the 2016 IEEE/ACM International Conference on Computer-Aided Design, Austin, TX, USA, 7-10 Nov., 2016.

[11] S. Dutt, S. Dash, S. Nandi, and G. Trivedi, "Analysis, modeling and optimization of equal segment based approximate adders," *IEEE Trans. Comput.*, vol. 68, no. 3, pp. 314-330, Mar. 2019.

[12] M. Najafi, S. Faraji, B. Li, D. Lilja, and K. Bazargan, "Accelerating deterministic bit-stream computing with resolution splitting," presented at the 20th International Symposium on Quality Electronic Design, Santa Clara, CA, 6-7 Mar., 2019.

[13] R. Wang, J. Han, B. Cockburn, and D. Elliott, "Stochastic circuit design and performance evaluation of vector quantization for different error measures," *IEEE Trans. Very Large Scale Integr. VLSI Syst.,* Article vol. 24, no. 10, pp. 3169-3183, Oct. 2016.

[14] C. Chang, J. Gu, and M. Zhang, "Ultra low-voltage low-power cmos 4-2 and 5-2 compressors for fast arithmetic circuits," *IEEE Trans. Circuits Syst. I-Regul. Pap.*, vol. 51, no. 10, pp. 1985-1997, Oct. 2004.

[15] H. Waris, C. H. Wang, and W. Q. Liu, "Hybrid low radix encoding-based approximate booth multipliers," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 67, no. 12, pp. 3367-3371, Dec. 2020.

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature,* vol. 521, no. 7553, pp. 436-444, May 2015.

[17] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM,* vol. 60, no. 6, pp. 84-90, Jun. 2017.

[18] A. Pappalardo. (2021). *Xilinx/brevitas*. Available: https://github.com/Xilinx/brevitas

[19] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," presented at the Proceedings of the 30th International Conference on Neural Information Processing Systems, Barcelona, Spain, 2016.