Accurate yet Efficient Stochastic Computing Neural Acceleration with High Precision Residual Fusion

Yixuan Hu[1], Tengyu Zhang[1], Renjie Wei[1], Meng Li[213*], Runsheng Wang[134], Yuan Wang[13] and Ru Huang[134]
[1]*School of Integrated Circuits &* [2]*Institute for Artificial Intelligence, Peking University, Beijing, China*
[3]*Beijing Advanced Innovation Center for Integrated Circuits, Beijing, China*
[4]*Institute of Electronic Design Automation, Peking University, Wuxi, China*

***Abstract*—Stochastic computing (SC) emerges as a fault-tolerant and area-efficient computing paradigm for neural acceleration. However, existing SC accelerators suffer from an intrinsic trade-off between inference accuracy and efficiency: accurate SC requires high precision computation but suffers from an exponential increase of bitstream length and inference latency. In this paper, we discover the high precision residual as a key remedy and propose to combine a low precision datapath with a high precision residual to improve inference accuracy with minimum efficiency overhead. We also propose to fuse batch normalization with the activation function to further improve the inference efficiency. The effectiveness of our proposed method is verified on a recently proposed SC accelerator. With extensive results, we show that our proposed SC-friendly network achieves 9.43% accuracy improvements compared to the baseline low precision networks with only 1.3% area-delay product (ADP) increase. We further show 3.01× ADP reduction compared to the baseline SC accelerator with almost iso-accuracy.**

***Index Terms*—Stochastic computing, neural acceleration, low precision datapath, high precision residual**

I. Introduction

Deep neural networks (DNNs) have achieved state-of-the-art (SOTA) accuracy in a wide range of applications. However, the fast scaling of DNNs introduces a rapid increase in network parameters and computation, which poses new challenges to computation resources and power budgets, especially for embedded edge devices. Stochastic computing (SC) emerges as a new computing paradigm and has gained a lot of attention in recent years for low-cost neural acceleration [1]–[7].

In SC, a number is represented by a bitstream in which the probability of 1s denotes its value [5]. The bitstream-based representation makes SC more tolerable to the errors compared to conventional binary representation [5]–[8]. Meanwhile, SC also achieves high area efficiency as its arithmetic operations can be processed by simple logic circuits. For example, multiplication and addition can be performed by a single AND and OR gate, respectively, which has a considerably lower hardware cost than a regular multiplier and adder [5].

Despite the aforementioned benefits, SC encounters an intrinsic trade-off between inference accuracy and efficiency [9], [10]. Specifically, the bitstream length (BSL) of the SC number representation increases exponentially concerning the inference precision. Hence, as shown in Figure 1, while efficient inference prefers low precision computation, e.g., BSL equals to 2 bit, the inference accuracy degrades significantly, e.g., more than 10%. By increasing BSL to 4 bit, the accuracy can be enhanced but at the cost of nearly 3x efficiency overhead.

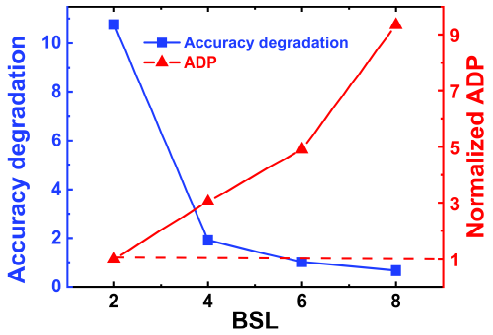

Fig. 1. The trade-off between inference accuracy and efficiency (measured by area-delay product, i.e., ADP): SC with 2b BSL achieves the highest efficiency at the cost of more than 10% accuracy degradation (we fix the weight BSL to 2 bit and sweep the activation BSL).

To enhance the inference efficiency and accuracy for SC, different optimizations have been proposed [4], [6], [9]–[13] to improve the SC multiplier or SC accelerators. However, they either suffer from the exponential increase of BSL for accurate inference [9], [10], [12], [13] or force low precision SC to trade-off inference accuracy with efficiency.

In this paper, we examine the origin of the accuracy degradation for low precision SC and observe that low precision activation results in limited network representation capability and is the root cause of low inference accuracy. As a remedy, we propose to leverage high precision residual connections to improve the network representation capability while keeping the majority of the computation in low precision. We verify our proposed strategy on a recently proposed SC-based accelerator and demonstrate both high inference accuracy and efficiency. Our contributions can be summarized as follows:

- We study the root cause of accuracy degradation for low precision SC and propose to combine high precision residual with low precision datapath to enable accurate yet efficient SC acceleration.
- We perpose to fuse batch normalization (BN) with activation function to further improve the SC inference efficiency.
- We verify our method on a baseline end-to-end SC-based accelerator and design customized SC modules to enable high precision residual fusion.
- We demonstrate 9.43% accuracy improvement for

*Corresponding author: meng.li@pku.edu.cn

978-3-9819263-7-8/DATE23/© 2023 EDAA

TABLE I
THE CORRESPONDING BINARY PRECISION AND THE REPRESENTED RANGE FOR THERMOMETER CODING OF DIFFERENT BSL.

BSL	Binary Precision	Range	Thermometer Coding
2	-	-1, 0, 1	00, 10, 11
4	2	-2, -1, 0, 1, 2	0000, 1000, 1100, 1110, 1111
8	3	-4, -3 ··· 3, 4	00000000, 10000000 ··· 11111110, 11111111
16	4	-8, -7 ··· 7, 8	0000000000000000, 1000000000000000 ··· 1111111111111110, 1111111111111111

TABLE II
COMPARISON WITH OTHER SC ACCELERATORS.

	Scale	BSL	BN Support	NN Co-Opt	Conv Cost w.r.t. Accuracy
[10], [12], [13]	Multiplier	256b	-	-	Exponential
[3]	Full System	256b - 1024b	No	No	Exponential
[6], [8]	Full System	2b	No	No	Exponential
[11]	Full System	32b	Yes	Yes	Exponential
This work	Full System	2b (Conv) 16b (residual)	Yes	Yes	Constant

ResNet18 on CIFAR10 dataset with almost iso-ADP. We also show 3.01× ADP reduction compared to the baseline SC accelerator with almost the same inference accuracy.

II. BACKGROUND

In this section, we introduce the basic concepts of SC [8] and the baseline fully parallel SC-based accelerator architecture [6]. In Section IV, we verify our proposed method on the baseline SC accelerator *while we emphasize our method can be generally applied to other SC accelerators.*

A. SC Overview

In the conventional binary representation, a n-bit number x represents $\sum_{i=0}^{n-1} x[i]2^i$, where $x[i]$ is the i-th bit of x and has a weight of 2^i. By contrast, SC leverages the uniform weight coding, i.e., a number is represented by a bitstream and each bit has the same weight. We refer interested readers to [5] for a more detailed introduction.

In our work, we leverage the thermometer coding scheme, which is deterministic with all the 1s appearing at the beginning of the bitstream. With thermometer coding, x is represented with a L-bit sequence as

$$x = \alpha x_q = \alpha(\sum_{i=0}^{L-1} x[i] - \frac{L}{2}),$$

where $x_q = \sum_{i=0}^{L-1} x[i] - \frac{L}{2}$ is the quantized value of range $[-\frac{L}{2}, \frac{L}{2}]$ and α is a scaling factor obtained by training. Following [14], α can be limited to the power of 2 to enable hardware-friendly processing. Because each bit $x[i]$ shares the same weights, a bit flip in SC results in a smaller error compared to the binary representation, which makes SC more error resilient [8]. Besides error resilience, SC can also achieve high area efficiency as simple AND and OR gates can implement approximate multiplication and addition operations, respectively. The high area efficiency and error resilience make SC a promising candidate for neural acceleration.

However, the uniform weight coding scheme is a double-edged sword. In Table I, we show the range and the binary precision for thermometer coding with different BSL. We can observe an exponential increase of BSL with the binary precision, which renders high precision SC very expensive. To improve the SC efficiency, [10] proposes a novel SC multiplication algorithm based on a deterministic bitstream generation algorithm, which reduces the computation randomness for better accuracy and latency. [12], [13] proposes a counter-based approximate multiplier and introduces new scale bits to improve the inference accuracy. Though effective, [10], [12] and [13] are only at multiplier level and still suffer from the exponential BSL increase for accurate inference. It is also unclear how to support other important NN structures, e.g., BN, etc. In the full level, [3] compares the system impact of different convolution block designs but suffers from high BSL and low efficiency. [6], [8] and [11] propose to directly leverage low precision bitstreams for better efficiency. Specifically, [11] co-optimizes the network and SC accelerator and allows using variable BSL for different layers while [6], [8] directly implements ternary convolutions. However, they all ignore the BN support and suffer from a large accuracy degradation. In Table II, we compare our work with the previous SC accelerator designs. Our design features co-designing the networks and the accelerator to achieve constant convolution cost with high inference accuracy.

Fig. 2. The fully SC architecture in [6]. It is designed for ternary neural networks and purely uses thermometer coding.

B. Baseline SC-based Architecture

To verify our proposed method, in this paper, we select a recently proposed SC-based accelerator [6] for a case study. [6] is end-to-end SC-based without the need of frequent conversion between SC and binary formats. It leverages the thermometer coding scheme with 2b BSL for both weights and activations. The accelerator features 3 major components as shown in Figure 2, including an exact SC multiplier, a bitonic sorting network (BSN), and a selective interconnect. The BSN ranks all the 1s in the front of the bitstream and implements exact accumulation in the thermometer coding scheme. The selective interconnect encodes different functions by controlling the input-to-output connections. Compared to other designs [3], it achieves higher accuracy, better parallelism, and more flexibility to support different activation functions.

The accelerator [6] is free of random fluctuations and achieves very high efficiency due to the low precision operands and highly parallel accumulation and activation functions. However, it suffers from a large accuracy degradation for large datasets. Directly increasing the inference precision of the accelerator incurs a large efficiency overhead as shown in Figure 1. Meanwhile, it is also unclear how to deal with network structures like residual, BN, etc, in the original design.

III. SC-FRIENDLY LOW PRECISION NETWORK

In this section, we analyze the origin of the accuracy degradation for low precision networks. Based on the analysis, we propose to augment the low precision networks with high

TABLE III
NETWORK ACCURACY COMPARISON OF DIFFERENT QUANTIZED NETWORKS ON CIFAR10: THE LOW PRECISION ACTIVATION NETWORK SUFFERS FROM SIGNIFICANT ACCURACY DEGRADATION COMPARED TO THE FLOATING POINT BASELINE AND LOW PRECISION WEIGHT NETWORK.

Network	Weight/BSL	Act/BSL	Top-1 Accuracy (%)
baseline	FP	FP	94.27
weight quantized	2	FP	93.98
activation quantized	FP	2	84.18
fully quantized	2	2	83.51

precision residual and SC-friendly operator fusion to improve both the inference accuracy and efficiency.

A. Accuracy Degradation of Low precision Networks

As shown in Figure 1, low precision network with 2b BSL achieves high inference efficiency but suffers from more than 10% accuracy degradation. Though efficient, such a large accuracy gap is clearly unacceptable for real-world applications. To understand the origin of the accuracy degradation, we quantize the network weights and activations to low precisions separately. As shown in Table III, the low precision weight network achieves very similar accuracy compared to the floating point baseline. By contrast, activation quantization with 2b BSL leads to 10% accuracy degradation. Hence, low precision activation is the root cause of the accuracy loss.

We hypothesize the poor performance of the low precision activation network is caused by its low representation capacity. Following [15], we define the representation capability of a tensor as the number of all possible configurations. As shown in Figure 3(a), after the low precision quantization, the range of activation is $\{-1, 0, +1\}$ for 2b BSL and hence, its total representation capacity is $3^{H\times W\times C}$, where H, W, and C denote the activation height, width, and channels, respectively. Compared to the high precision activation of the baseline network, the representation capability is reduced drastically.

To enhance the network representation capability, one naive solution is to increase the quantization precision and enhance the range of the activation. However, such solution comes at a large efficiency cost as shown in Figure 1. We observe the widely used residual connections can act as a remedy and propose to use high precision residual connections to improve the network representation capacity. As shown in Figure 3(b), the residual connects the inputs to the activation low precision quantization layer to the output of the convolution layer, and these two activations are added before the BN. The proposed structure enables efficient BN processing as detailed in Section III-B. We keep the residual in high precision (e.g., 16b BSL, i.e., 4 bit binary precision), which, on one hand, does not impact the efficiency of the convolution; on the other hand, increases the range of the activation to $\{-8, -7, \ldots, 7, 8\}$ and enhance the representation capability to $17^{H\times W\times C}$. As we will show in the experiments, high precision residual helps to improve the inference accuracy significantly while incurring minimum efficiency overhead.

B. BN Fusion for High Inference Efficiency

Besides the high precision residual, another remaining question is how to efficiently process BN. BN is proposed to enable

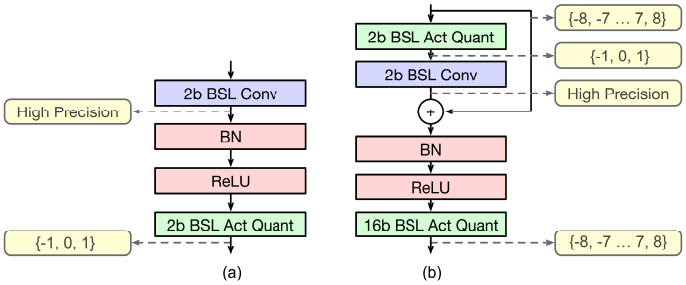

Fig. 3. High precision residual helps to achieve better representation capability.

training very deep networks and is essential for training low precision networks [16]. Previous works [17], [18] only focus on optimizing the convolution layers and use floating point computation for BN, which is unfriendly to SC and incurs a large overhead. We propose to fuse BN with the ReLU activation function as below:

$$\begin{aligned} \mathrm{BN}(x) &= \gamma(x-\beta) \\ \mathrm{ReLU}(\mathrm{BN}(x)) &= \begin{cases} \gamma(x-\beta) & x \geq \beta \\ 0 & x < \beta \end{cases} \end{aligned} \tag{1}$$

where γ and β are trainable parameters for BN. The fused BN and ReLU function can be accurately and efficiently processed in SC through the selective interconnect as explained in Section IV-B.

IV. CASE STUDY: END-TO-END SC ACCELERATOR WITH HIGH PRECISION RESIDUAL

In this section, we demonstrate the effectiveness of the proposed high precision residual and BN fusion on the SC accelerator proposed in [6]. While we select an end-to-end SC accelerator for the case study, our proposed principle is general and can be applied to other SC accelerator designs.

A. Example of a Convolution Layer

We first use an example of a single convolution layer shown in Figure 4 to explain the hardware blocks required to support our proposed high precision residual. Assume we use 16b and 2b BSL for the high precision residual and low precision convolution, respectively. Then, a convolution layer involves 5 major steps:

① The 16b BSL high precision activations are first subsampled to 2b BSL before feeding to the convolution. To keep the range of values constant, the scaling factor α of the activation is updated to $1/4 \times 16 \div 2 = 2$.

② 2b BSL activations and weights are multiplied using the low precision multipliers from the baseline accelerator [6].

③ Due to the scaling factor mismatch between the residual and the multiplication products, we re-scale the residual first. Thanks to the uniform weight coding in SC , we just need to replicate the residual 2 times to reduce the scaling factor α from $1/4$ to $1/8$.

④ The residual and the multiplication products are concatenated and accumulated through the sorting network.

⑤ The fused operation of BN, ReLU, and activation requantization is conducted by the selective interconnect.

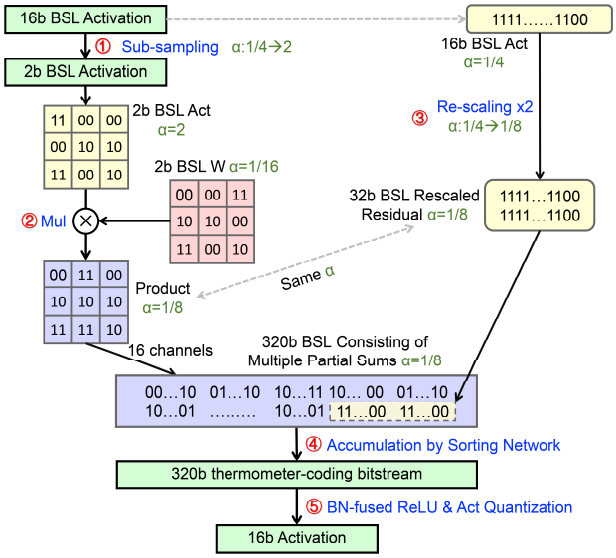

Fig. 4. A convolution layer of the proposed SC accelerator.

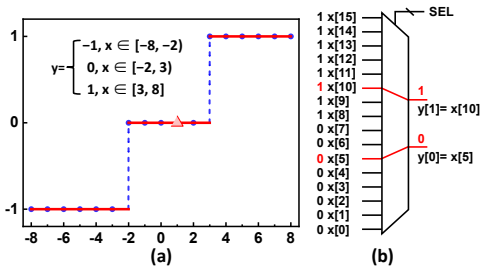

Fig. 5. A 16-to-2 activation sub-sampling block.

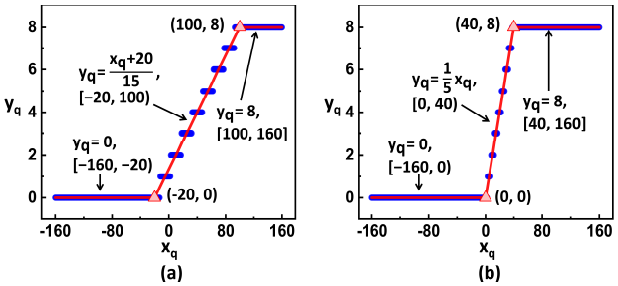

Fig. 6. Two examples of BN-fused activation function with 16b BSL output. The red lines are the expected BN-fused ReLU curve mentioned in Equation 1 and the blue dots are the outputs of the proposed design.

B. SC Accelerator with High Precision Residual

Based on the example described above, we can summarize the major components to support the high precision residual in SC as follows:

- Activation sub-sampling block: sub-sample the high precision activation to low precision for efficient convolution computation.
- Low precision multiplier: multiply the low precision weights and activations.
- Residual re-scaling block: re-scale the residual to match the scaling factor of the residual and multiplication products before accumulation.
- BSN-based accumulation block: sum the multiplication products and the residual together.
- Selective interconnect for the activation function: handle the fused operation of BN, activation function (e.g., ReLU), and activation re-quantization.

The low precision multiplier and the accumulation block can be directly inherited from the baseline accelerator as introduced in Section II-B. In this section, we focus on introducing the design of the other blocks.

Activation Sub-sampling Block and Activation Function Block: Both blocks can be implemented with the selective interconnect. We use the sub-sampling block as an example to explain the implementation. The activation sub-sampling block implements the following conversion:

$$y_q = \text{round}(\frac{\alpha_x}{\alpha_y} x_q)$$

where x_q and y_q denote the high precision and low precision quantized activations, respectively. Assume $\frac{\alpha_x}{\alpha_y} = \frac{1}{4}$, then, we have

$$y_q = \begin{cases} -1 & (00), \quad -8 \le x_q < -2 \\ 0 & (10), \quad -2 \le x_q < 3 \\ 1 & (11), \quad 3 \le x_q \le 8 \end{cases}.$$

Hence, we can derive $y_q[1]$ and $y_q[0]$ as

$$y_q[1] = \begin{cases} 0, & -8 \le x_q < -2 \\ 1, & -2 \le x_q \le 8 \end{cases}, \quad y_q[0] = \begin{cases} 0, & -8 \le x_q < 3 \\ 1, & 3 \le x_q \le 8 \end{cases}.$$

As we use thermometer coding for both x_q and y_q, we can further derive $y_q[1] = x_q[5]$ and $y_q[0] = x_q[10]$. Therefore, for a given $\frac{\alpha_x}{\alpha_y}$, the sub-sampling block is equivalent to connecting each output bit with a certain input bit. To further support flexible scaling factor ratios, two multiplexers with the corresponding select signal are added to control the connection. In Figure 5, we show the sub-sampling function and the hardware implementation for the example above.

Following a similar methodology, we can implement the activation function block. We omit the detailed explanation but show two examples in Figure 6. As can be observed, with selective interconnect, we can achieve a very accurate approximation of the fused activation function.

Residual Re-scaling Block: The residual re-scaling block aligns the scaling factors for the residual and multiplication products before the accumulation. Since we limit the scaling factors to be the power of 2 following [14], in the re-scaling block, we just need to multiply or divide the residual by a factor 2^N, where N is an integer. In Algorithm 1, we describe the logic for the residual re-scaling block assuming 16b BSL residual. To multiply the residual by 2^N, we just need to replicate the residual by 2^N times. This is realized by writing the same residual to the buffer 2^N times. When dividing the residual by 2^N, we make a 1-out-of-2 bit selection of the residual every cycle and generate the final result after N cycles. To keep the BSL for residual constant, we always concatenate 8b'11110000 (equal to 0) in each cycle of the division.

Algorithm 1: Residual Re-scaling Block.

Input: 16b BSL input D_{in}, scaling factor 2^N, mode M (multiplication or division)
Output: Calculation result $Dout$
if $M == "multiplication"$ **then**
  for $i = 1 : 2^N$ **do**
    D_{out} = D_{in};
    output D_{out}; // Output D_{out} every cycle
  end
else
  for $i = 1 : N$ **do**
    $D_{out} = \{D_{in}[15:0:2], 8'b11110000\}$;
  end
  output D_{out}; // Output D_{out} only once
end

V. Experimental Results

A. Experiment setup

Models and Datasets: We evaluate our methods on two modern convolutional networks (ResNet18 and ResNet34 [19]) that are widely used and two datasets (CIFAR10 and CIFAR100 [20]). We adjust the total strides of both networks from 32 to 8 to deal with the small input resolutions of CIFAR datasets.

Training Settings: We train all the models for 310 epochs with 10 warm-up epochs and use an AdamW optimizer [21] with a momentum of 0.9, an initial learning rate of 7.5e-4, and a cosine learning rate schedule. We set the training batch size to 128. We follow [17] for the data augmentation and use learned step size quantization (LSQ) to quantize both weights and activations [22]. To improve the accuracy of the quantized network, we follow [17] to use the progressive quantization algorithm, i.e., we first train a network with floating point weights and quantized activations; then, in the second step, we re-load the checkpoint and train a fully quantized network. We use knowledge distillation (KD) in both steps with the floating point model as a teacher.

Hardware Evaluation: We implement the register transfer level (RTL) code for both baseline SC accelerator and our improved design and then, synthesize both designs using Synopsys Design Compiler with TSMC 28nm technology library. We report the hardware metrics based on the synthesis results. The implemented SC accelerator has 144 multipliers, each supporting 2b BSL for weights, 2/4 bit BSL for activations, and 2/4/8/16 bit BSL for the residual. Depending on the BSL for the activations, a 320-bit BSN and a 640-bit BSN are implemented for the accumulation and residual connection.

B. Main Results

Network Accuracy Comparison: We compare the fully ternarized network with our proposed SC-friendly network that features high precision residual and BN fusion. We have the following findings based on the results shown in Table IV:

- With the high precision residual, network accuracy is improved significantly by 8.69% and 8.12% for low precision ResNet18 on CIFAR10 and CIFAR100, respectively. Combined with the novel training techniques, network accuracy can be improved in total by 9.43% and 15.42%, respectively.
- Low precision networks with 2b BSL for weights and activations can be hard to converge when the network gets deeper and the dataset gets harder (see low precision ResNet34 trained on CIFAR100). By contrast, high precision residual improves the training convergence and helps the network to gain higher accuracy.

We also observe the low precision ResNet34 in general performs worse compared to the ResNet18. We hypothesize this is because deep, low precision networks are harder to converge. We leave more in-depth study as our future work.

TABLE IV
Improve inference accuracy with high precision residual (i.e., 16b BSL) and novel network training strategies. Note that fully ternarized ResNet34 does not converge on CIFAR100.

Network	Variant	CIFAR10 (%)	CIFAR100 (%)
ResNet18	2b BSL W & Act	82.58	55.89
	+ 16b BSL Residual	91.29	64.01
	+ Two-step Quant	91.95	68.58
	+ KD	92.01	71.31
ResNet34	2b BSL W & Act	76.11	-
	+ 16b BSL Residual	91.13	66.77
	+ Two-step Quant	91.77	68.12
	+ KD	91.75	70.80

TABLE V
Impact of residual precision on ResNet18 accuracy.

Network	Weight/BSL	Act/BSL	Res/BSL	*Accuracy (%)
Baseline	2	2	2	86.23
ResNet18	2	2	4	90.40
	2	2	6	91.61
	2	2	8	91.64
	2	2	16	92.01
	2	2	FP	92.16

*All (including baseline) using the novel training techniques in Section V-A.

Impact of High Precision Residual: We take ResNet18 trained on CIFAR10 to analyze the impact of residual precision on network accuracy. We fix the weight and activation precision to 2b BSL and sweep the residual precision from 2b BSL to 16b BSL. We report the network accuracy in Table V. We also report the network with floating point residual although it cannot be directly supported by the SC accelerator. As shown in Table V, increasing the residual precision from 2b BSL to 4b BSL enhances the inference accuracy by more than 4%. While the accuracy gain slows down with the increase of residual precision, the network with 16b BSL residual achieves almost the same accuracy as the network with floating point residual.

Accelerator Efficiency Comparison: We now compare different accelerators with 3 different precision configurations on their area, delay, and ADP. As shown in Table VI, compared to the baseline accelerator with 2b BSL for the weight, activation, and residual, our proposed design achieves 9.43% accuracy improvements with only 1.3% ADP overhead. Meanwhile, compared to the accelerator with 2b BSL weight and 4b BSL activation/residual, we achieve 3.01× ADP reduction with almost iso-accuracy (less than 0.35% difference).

Accelerator Area Breakdown and Energy Breakdown: We further show the area and energy breakdown of the accelerator

TABLE VI
INFERENCE EFFICIENCY AND ACCURACY COMPARISON.

	W-A-R/BSL	Area (um²)	Delay (ns)	ADP (um²·us)	Accuracy (%)
Baseline [6]	2-2-2	4349.7	51.81	225.36	82.58
	2-4-4	10683.3	64.35	687.47	92.35
This work	2-2-16	4406.9	51.81	228.32	92.01

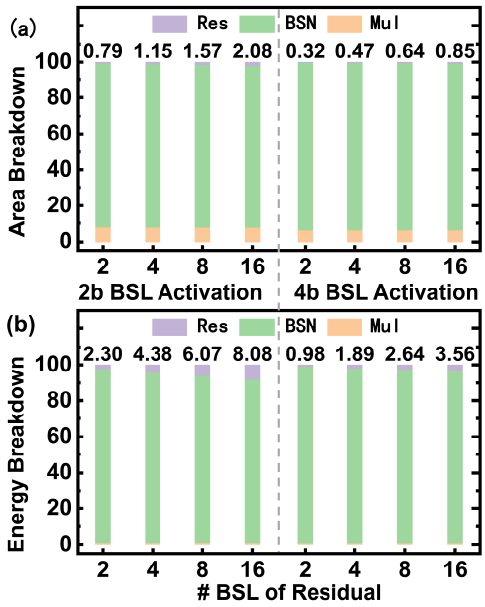

Fig. 7. For a 3x3x512 convolution kernel, the a) area breakdown and b) energy breakdown of the SC accelerator for different BSLs of residual from 2 to 16. The left part corresponds to 2b BSL activation and the right part corresponds to 4b BSL activation. And 2b BSL weight is used in all cases.

with different precisions for the residual. We fix the weight precision to 2b BSL and evaluate the breakdown for both 2b BSL and 4b BSL activations. We sweep the precision of the residual and the results are shown in Figure 7. As we can see, while the area and energy to handle the residual increases with the residual precision, it only accounts for a very small portion, i.e., 2.08% area and 8.08% energy for 2b BSL activations. When increasing the activation precision to 4b BSL, the residual takes an even smaller area and energy. This further verifies the low overhead and high efficiency of our proposed method.

VI. CONCLUSION

In this paper, we study the accuracy-efficiency trade-off of SC-based neural acceleration. We discover the low precision activation as the major accuracy bottleneck and propose to leverage the high precision residual as a remedy. We combine the low precision computation with a high precision residual to enable accurate yet efficient SC inference. The proposed method is verified on a recently proposed end-to-end SC-based accelerator. Compared to the baseline design, we achieve 9.43% accuracy improvement with only 1.3% efficiency overhead and achieve 3× efficiency improvement with comparable accuracy.

REFERENCES

[1] Z. Xia, R. Wan, J. Chen, and R. Wang, "Reconfigurable spatial parallel stochastic computing for accelerating sparse convolutional neural networks," in *SCIENCE CHINA Information Sciences*, 2022.

[2] S. Liu and J. Han, "Energy efficient stochastic computing with sobol sequences," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 650–653.

[3] A. Ren, Z. Li, C. Ding, Q. Qiu, Y. Wang, J. Li, X. Qian, and B. Yuan, "Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing," in *International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. ACM, 2017, p. 405–418.

[4] R. Hojabr, K. Givaki, S. R. Tayaranian, P. Esfahanian, A. Khonsari, D. Rahmati, and M. H. Najafi, "Skippynn: An embedded stochastic-computing accelerator for convolutional neural networks," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[5] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, "A survey of stochastic computing neural networks for machine learning applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 2809–2824, 2020.

[6] Y. Zhang, S. Lin, R. Wang, Y. Wang, Y. Wang, W. Qian, and R. Huang, "When sorting network meets parallel bitstreams: A fault-tolerant parallel ternary neural network accelerator based on stochastic computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 1287–1290.

[7] Y. Song, E. H.-M. Sha, Q. Zhuge, R. Xu, Y. Zhang, B. Li, and L. Yang, "Bsc: Block-based stochastic computing to enable accurate and efficient tinyml," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2022, pp. 314–319.

[8] Y. Hu, Y. Zhang, R. Wang, Z. Zhang, J. Song, X. Tang, W. Qian, Y. Wang, Y. Wang, and R. Huang, "A 28-nm 198.9-tops/w fault-tolerant stochastic computing neural network processor," *IEEE Solid-State Circuits Letters*, vol. 5, pp. 198–201, 2022.

[9] H. Xiong, G. He *et al.*, "Hardware implementation of an improved stochastic computing based deep neural network using short sequence length," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 11, pp. 2667–2671, 2020.

[10] H. Sim and J. Lee, "A new stochastic computing multiplier with application to deep convolutional neural networks," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[11] W. Romaszkan, T. Li, T. Melton, S. Pamarti, and P. Gupta, "Acoustic: Accelerating convolutional neural networks through or-unipolar skipped stochastic computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2020, pp. 768–773.

[12] S. Yu, Y. Liu, and S. X.-D. Tan, "Cosaim: Counter-based stochastic-behaving approximate integer multiplier for deep neural networks," in *ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 499–504.

[13] S. Yu and S. X.-D. Tan, "Scaled-cbsc: scaled counting-based stochastic computing multiplication for improved accuracy," in *ACM/IEEE Design Automation Conference (DAC)*, 2022, pp. 1003–1008.

[14] L. Lai, N. Suda, and V. Chandra, "Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus," *arXiv preprint arXiv:1801.06601*, 2018.

[15] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, "Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm," in *European conference on computer vision (ECCV)*, 2018, pp. 722–737.

[16] T. Chen, Z. Zhang, X. Ouyang, Z. Liu, Z. Shen, and Z. Wang, "Bnn-bn=?: Training binary neural networks without batch normalization," in *IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021, pp. 4619–4629.

[17] B. Martinez, J. Yang, A. Bulat, and G. Tzimiropoulos, "Training binary neural networks with real-to-binary convolutions," *arXiv preprint arXiv:2003.11535*, 2020.

[18] Z. Liu, B. Oguz, A. Pappu, L. Xiao, S. Yih, M. Li, R. Krishnamoorthi, and Y. Mehdad, "Bit: Robustly binarized multi-distilled transformer," *arXiv preprint arXiv:2205.13016*, 2022.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[20] A. Krizhevsky, "Learning multiple layers of features from tiny images," in *Tech Report*, 2009.

[21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[22] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, "Learned step size quantization," *arXiv preprint arXiv:1902.08153*, 2019.