

ASCEND: Accurate yet Efficient End-to-End Stochastic Computing Acceleration of Vision Transformer

Tong Xie^{1†}, Yixuan Hu^{1†}, Renjie Wei^{1†}, Meng Li^{213*}, Yuan Wang¹³, Runsheng Wang¹³⁴, and Ru Huang¹³⁴

¹School of Integrated Circuits & ²Institute for Artificial Intelligence, Peking University, Beijing, China

³Beijing Advanced Innovation Center for Integrated Circuits, Beijing, China

⁴Institute of Electronic Design Automation, Peking University, Wuxi, China

[†]Equal contribution *Corresponding author: meng.li@pku.edu.cn

Abstract—Stochastic computing (SC) has emerged as a promising computing paradigm for neural acceleration. However, how to accelerate the state-of-the-art Vision Transformer (ViT) with SC remains unclear. Unlike convolutional neural networks, ViTs introduce notable compatibility and efficiency challenges because of their nonlinear functions, e.g., softmax and Gaussian Error Linear Units (GELU). In this paper, for the first time, a ViT accelerator based on end-to-end SC, dubbed ASCEND, is proposed. ASCEND co-designs the SC circuits and ViT networks to enable accurate yet efficient acceleration. To overcome the compatibility challenges, ASCEND proposes a novel deterministic SC block for GELU and leverages an SC-friendly iterative approximate algorithm to design an accurate and efficient softmax circuit. To improve inference efficiency, ASCEND develops a two-stage training pipeline to produce accurate low-precision ViTs. With extensive experiments, we show the proposed GELU and softmax blocks achieve 56.3% and 22.6% error reduction compared to existing SC designs, respectively, and reduce the area-delay product (ADP) by 5.29 \times and 12.6 \times , respectively. Moreover, compared to the baseline low-precision ViTs, ASCEND also achieves significant accuracy improvements on CIFAR10 and CIFAR100.

Index Terms—Stochastic computing, vision transformer, approximate computation, circuit/network co-design

I. INTRODUCTION

Transformer has been widely applied to computer vision tasks, including classification [1], object detection [2], segmentation [3], etc. With the high model capacity and large receptive field [1], vision transformers (ViTs) have achieved state-of-the-art (SOTA) accuracy compared to convolutional neural networks (CNNs). However, the high accuracy is achieved at the cost of a rapid increase in ViT parameters and computation [4], which calls for more efficient neural accelerators.

Stochastic computing (SC) emerges as a new computing paradigm and has attracted much attention for neural acceleration in recent years [5]–[10]. By representing a number with a stochastic bitstream, in which the probability of 1's denotes the value, SC achieves simplified arithmetic logics and improved fault tolerance [11]. End-to-end SC avoids the back-and-forth conversion between the binary and the SC representations, achieving even higher hardware efficiency [5], [10].

This work was supported in part by the National Key R&D Program of China (2020YFB2205502), NSFC (62125401) and the 111 Project (B18001).

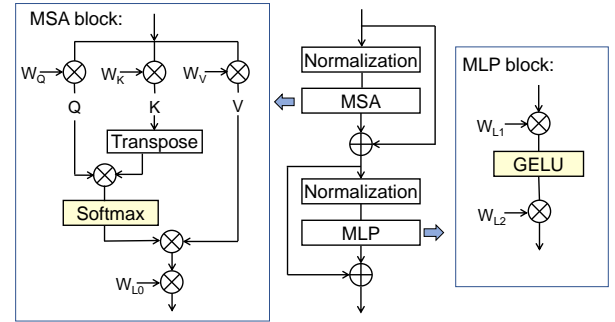


Fig. 1. The diagram of a transformer encoder block.

However, we observe end-to-end SC encounters major compatibility and efficiency challenges for ViT acceleration. On one hand, ViTs have more complex nonlinear functions such as Gaussian Error Linear Unit (GELU) in the multi-head self-attention (MSA) block and softmax in the multi-layer perceptron (MLP) block [1] as shown in Fig. 1. While these functions are important to the accuracy of ViTs [12], their SC implementations are not well studied in existing works. On the other hand, ViTs can be harder to quantize and require higher computation precision [13], [14], resulting in an exponential increase of SC representation bitstream length (BSL) [15] and drastic degradation of inference efficiency.

Therefore, a natural question to ask is *whether it is possible to leverage SC to realize accurate yet efficient ViT acceleration*. In this paper, we provide an affirmative answer and propose ASCEND, the first end-to-end SC-based ViT accelerator. ASCEND features a co-design of the SC circuits and ViT networks to address the aforementioned challenges. At the circuit level, we observe implementing the division and exponential functions in GELU and softmax directly is challenging for SC. Therefore, we propose a novel deterministic SC block for the GELU function. We leverage an SC-friendly iterative approximate algorithm and implement the corresponding SC circuits to achieve an accurate and efficient softmax function. At the network level, a two-stage pipeline is proposed for SC-friendly ViT training, which improves the accuracy of low-precision ViTs through progressive quantization and SC circuit-aware fine-tuning. Our contributions can be summarized as follows:

TABLE I
GAP BETWEEN MODEL REQUIREMENTS AND EXISTING SC CIRCUITS.

SC Design	Supported Model	Encoding Format	Supported Function	Implementation Method
[6]–[8]	CNN	stochastic	tanh, sigmoid	FSM
[9]	CNN	stochastic	ReLU	FSM
[16], [17]	CNN	stochastic	softmax	FSM, binary units
[5], [15]	CNN	deterministic	ReLU	SI
Ours	ViT	deterministic	GELU, softmax	Gate-Assis. SI, BSN

- We propose the first end-to-end SC accelerator for ViT and design novel SC blocks for GELU and softmax.
- We propose a two-stage training pipeline for SC-friendly low-precision ViT, which features progressive quantization and SC circuit-aware fine-tuning.
- We demonstrate 56.3% and 22.6% mean average error (MAE) reduction compared to existing baselines for GELU and softmax, respectively while achieving $5.29\times$ and $12.6\times$ area-delay product (ADP) reduction. ASCEND also achieves significant accuracy improvement compared to baseline low-precision ViTs.

II. BACKGROUND

A. SC Overview

Unlike traditional binary encoding, each bit in an SC representation carries equal weight. It leverages the probability of 1's in the bitstream to represent values. Since probabilities inherently lie within $[0, 1]$, SC encoding relies on scaling factors to map the probability to a desired range.

The simplest unipolar encoding expresses the value of $[0, 1]$ with the probability of 1's, denoted as p , directly. To represent both positive and negative values, bipolar encoding defines the value as $2p - 1$, thus mapping the probability to the range of $[-1, 1]$. For these encoding schemes, multiplications and additions can be implemented with simple logics, e.g., AND and MUX gates for unipolar encoding [7]. However, the stochastic nature of these encoding schemes usually leads to large computation errors and fluctuation [10].

Another encoding format known as thermometer encoding [10] offers a deterministic approach where all the 1's appear at the beginning of the bitstream. A data x is represented with an L -bit sequence as $x = \alpha x_q$, where α is the scaling factor and $x_q = \sum_{i=0}^{L-1} x[i] - \frac{L}{2} \in [-\frac{L}{2}, \frac{L}{2}]$. The multiplication can thus be implemented based on a truth table [10]. The addition can be realized by concatenating the input bitstreams together using a Bitonic Sorting Network (BSN) [5].

B. SC-based Method for Nonlinear Functions

There are three categories of SC designs for nonlinear functions. The first category is based on finite state machines (FSMs) and saturated counters. By adjusting the FSM transition conditions, different nonlinear functions can be implemented [6]–[9]. However, due to the nature of sequential processing, the design often requires very long bitstreams while the output error is still large.

The second category relies on the Bernstein polynomial to approximate nonlinear functions [18]. This algorithm requires

high degree polynomials which have many terms and long input bitstreams to reduce the approximation error. Moreover, a large number of stochastic number generators (SNGs) are needed, resulting in a very high hardware cost.

The third category, namely selective interconnect (SI), is proposed for thermometer coding. SI processes the whole input bitstream in parallel and computes nonlinear functions by controlling the position of output transitions, which enables accurate computation with a relatively short bitstream [5], [15]. However, it can only support monotonic functions.

There are many works employing these methods for nonlinear functions in CNNs, but none of them can support GELU and softmax accurately, as shown in Table I. [6]–[9] design FSMs for tanh, sigmoid, and ReLU in traditional stochastic encoding format. But they all need very long bitstreams to reduce the computation fluctuations. [16], [17] implements softmax for the last layer of CNNs with various binary compute units and multiple SC-binary conversions. However, only the relative order of outputs is preserved while the computed values still exhibit a large error. [5], [15] are representative end-to-end SC architectures for CNNs employing deterministic thermometer encoding format, but only monotonic nonlinear functions, e.g., ReLU, sigmoid, can be supported.

III. CHALLENGES OF SC-BASED ViT ACCELERATION

In this section, we discuss the challenges of designing SC blocks for GELU and softmax.

A. SC Blocks for GELU

Existing methods introduced in Section II-B all suffer from different limitations when implementing GELU:

FSM-based methods struggle with implementing GELU for both positive and negative input ranges. As shown in Fig. 2 (a), when the input is a small negative value, the output of FSM-based methods saturates at 0, resulting in systematic errors [9]. For the positive range, they are forced to handle a very long BSL to reduce random errors, significantly impacting hardware efficiency [15].

Bernstein polynomial-based methods approximate nonlinear functions through polynomial fitting. Their hardware cost increases with the degree of polynomials. As shown in Fig. 2 (b), low-degree Bernstein polynomial fitting suffers from a high computation error while high-degree fitting results in a drastic increase in hardware cost. Moreover, this design also exhibits noticeable computation fluctuations, which forces to use long bitstreams and further increases the inference cost.

SI-based methods can only implement monotonic functions while GELU is non-monotonic [5]. As shown in Fig. 2 (c), naively adopting SI-based methods suffers from a high computation error for the negative input range. However, it should be noted the positive range can be accurately implemented even for short bitstreams.

B. SC Blocks for Softmax

Softmax involves computing the division and exponential functions, both of which are challenging to support with SC. The exponential function usually demands high precisions for its input and output, leading to an exponential increase in

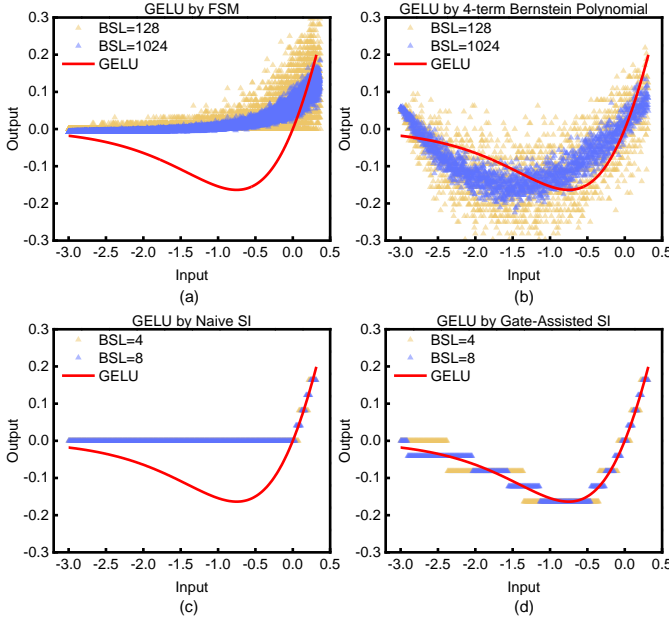


Fig. 2. GELU by (a) FSM-based design, (b) 4-term Bernstein polynomial, (c) naive SI-based design, and (d) the proposed gate-assisted SI design.

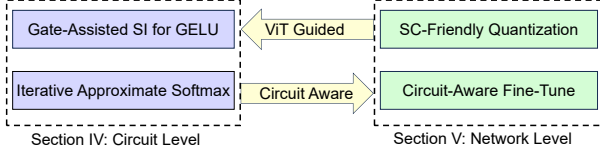


Fig. 3. The circuit/network co-design of the proposed ASCEND.

operands' BSLs [15]. The SC-based division function incurs a high hardware cost due to extensive usage of SNGs with JK flip-flops [19] or adaptive digital elements [20] or comparators [20]. It also suffers from high computation error as only approximate divisions like $p_x/(p_x + p_y)$ can be implemented rather than p_x/p_y [20], [21]. Moreover, existing SC implementations for division and exponential function all leverage sequential bitstream processing with high random fluctuations.

C. Efficiency

Apart from the compatibility of nonlinear functions, SC-based ViT accelerators also encounter efficiency bottlenecks. Compared to CNNs, ViT often requires significantly larger computational resources and the problem is further exacerbated by the coding efficiency of SC. As mentioned in Section II-A, a bitstream of L BSL can only represent $L + 1$ values. Hence, data of n -bit binary precision require 2^n BSL to represent in SC, indicating an exponential increase of BSL. In parallel SC circuits, higher precision results in an exponential increase in area [15], whereas in serial SC circuits, it signifies an exponential growth in delay [7]. This trade-off becomes even more crucial for ViT, given its already demanding computational resource requirements.

IV. EFFICIENT SC-BASED ViT ACCELERATOR CIRCUIT

To address the challenges in Section III, we propose the ASCEND framework for co-designing the SC circuits in this section and ViT networks in Section V. The overall flow is illustrated in Fig. 3.

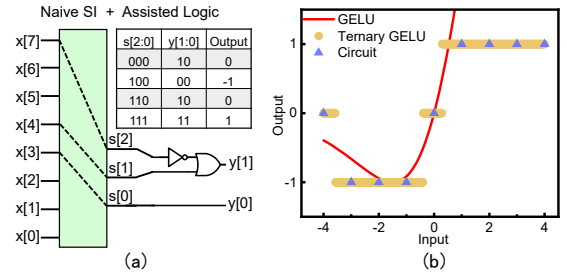


Fig. 4. (a) Gate-assisted SI implements non-monotonic functions with the help of simple combinational logics. (b) Ternary GELU implemented by (a).

A. Gate-Assisted SI for GELU Activation

Since the three methods mentioned above all have their limitations, we propose gate-assisted SI to implement the GELU function.

SI-based designs leverage parallel bitstreams and can obtain all the input information for the output bit, allowing for accurate results. But naive SI directly outputs selected bits, which increases the number of 1's in the output as the number of 1's in the input grows, limiting it to monotonic functions. To address this issue, we propose gate-assisted SI, which introduces extra gates and outputs the logical results of selected bits rather than directly outputting them, as shown in Fig. 4 (a). For example, a NOT gate and an AND gate can assist in implementing non-linear functions like GELU, which exhibit a decrease followed by an increase.

In the case of ternary GELU with an 8b BSL input and a 2b BSL output, as illustrated in Fig. 4, we use assisted logic to set the outputs as $y[1] = !s[2] \& s[1]$, $y[0] = s[0]$. The selected bits are controlled by selection signals derived from the input. When the input is small, all the selected bits are 0, resulting in the output $y[1:0]$ being "10" corresponding to the value 0. As the input increases, $x[7]$ to $x[0]$ gradually transitions from 0 to 1, and so do the selected bits. When $s[2]$ becomes 1, the output $y[1:0]$ changes to "00" corresponding to a decrease to -1. When $s[1]$ also becomes 1, the output $y[1:0]$ reverts to "10" corresponding to an increase back to 0. When the input increases further and $s[0]$ becomes 1, the output $y[1:0]$ changes to "11" corresponding to an increase to 1.

Fig. 2 (d) illustrates the GELU functions implemented by gate-assisted SI with different precision. The proposed GELU design is free of random fluctuations, allowing the exact implementation of the required GELU function.

B. SC Block of Iterative Approximate Softmax

Due to the limitations mentioned in Section III-B, existing research has not been able to accurately implement an efficient SC-based softmax. We propose introducing an iterative approximation of softmax [22] to address this issue, as shown in Algorithm 1. With an m -dimensional parameterized function $y(t) = \text{softmax}(tx)$, where $y(0) = 1/m$ and $y(1) = \text{softmax}(x)$, we can approximate $y(1)$, i.e., the softmax, from $y(0)$ using a summation of k terms instead of an integral:

$$y(1) = y(0) + \int_0^1 y'(t)dt \approx y(0) + \sum_{j=0}^{k-1} y'(\frac{j}{k}) \cdot \frac{1}{k}$$

Algorithm 1: Iterative Approximation of Softmax

Input : Softmax input vector x of dimension m
Output: Vector y after k iterations

- 1 Initialize $y_i^0 = \frac{1}{m} \quad \forall i \in \{0, 1, \dots, m-1\}$;
 - 2 **for** $j = 1 : k$ **do**
 - 3 $z_i = x_i \cdot y_i^{j-1} \quad \forall i \in \{0, 1, \dots, m-1\}$;
 - 4 $y_i^j = y_i^{j-1} + [z_i - y_i^{j-1} \cdot \text{sum}(z)]/k \quad \forall i \in \{0, 1, \dots, m-1\}$;
 - 5 **return** y ;
-

TABLE II
PARAMETERS IN THE PROPOSED SOFTMAX CIRCUIT BLOCK.

Symbol	Meaning
m	length of the row vector
k	count of iteration
B_x	bitstream length of x
α_x	scaling factor of x
B_y	bitstream length of y
α_y	scaling factor of y
s_1	sub-sample rate of $\text{sum}(z)$
s_2	sub-sample rate of $\text{sum}(z) \times y$

Due to the exponential term in $y(t)$, we compute and observe that $y'(t)$ can be straightforwardly expressed based on the value of $y(t)$. Therefore, we can iteratively calculate and obtain $y'(0), y'(1/k), y'(1/k), y'(2/k), \dots, y'(1)$, starting from the known value of $y(0) = 1/m$.

This iterative approximation simplifies the intricate computations of an m -dimensional vector x into iterations of t , avoiding the challenges of implementing division, exponentiation, and higher-order multiplications between bitstreams in SC. Instead, only multiplication and accumulation operations are required, along with the division of bitstreams by a constant k , which enables the efficient softmax circuits in SC.

Based on Algorithm 1, we propose the corresponding SC circuit in Fig. 5. Here, multiplication and accumulation can be implemented as mentioned in Section II-A, and division by a constant k can be implemented by just dividing the scaling factor by k without any operation on the bitstream.

The circuit block consists of m compute units for m elements of the softmax row vector and a global BSN ①. Each compute unit takes inputs x_i and y_i^{j-1} and computes y_i^j for the current iteration. Multiplier ① executes the multiplication for obtaining z_i in the 3rd line of Algorithm 1. Multiplier ② performs the multiplication $y_i^{j-1} \cdot \text{sum}(z)$. Meanwhile, BSN ① calculates the summation of z_i .

Up to this point, we have acquired all terms in the 4th line of Algorithm 1. We utilize two re-scaling blocks [15] to align their scaling factors. And BSN ② carries out the final accumulation to obtain y_i^j , which yields $y_i^j = y_i^{j-1} + [z_i - y_i^{j-1} \cdot \text{sum}(z)]/k$ as the output of this iteration. The final result of softmax is available after k iterations. Table II lists the parameters used to describe the detail of the circuit.

Furthermore, we emphasize the general SC-friendly nature of iterative approximate softmax and it can be also applied to other SC designs.

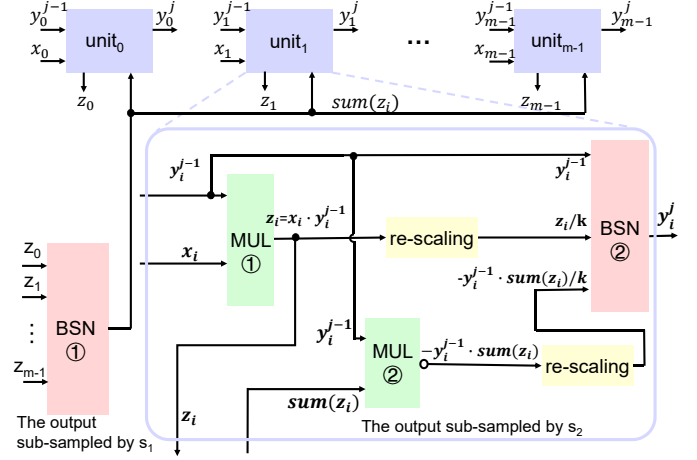


Fig. 5. SC circuit block of Iterative Approximate Softmax.

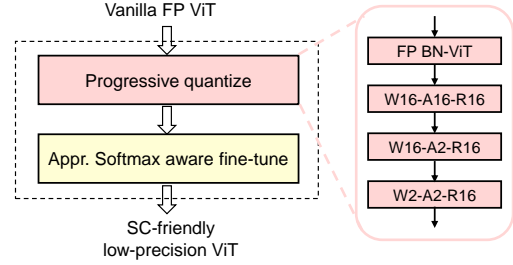


Fig. 6. The proposed SC-friendly low-precision ViT training pipeline.

V. SC-FRIENDLY LOW-PRECISION ViT

The circuit design mentioned above solves the challenges posed by GELU and softmax in ViT. To further enhance ViT's compatibility with SC, we substitute layer normalization (LN) with batch normalization (BN). With knowledge distillation (KD), the replacement leads to less than 0.1% accuracy impact on CIFAR10 and CIFAR100 datasets. To address the challenge in efficiency, we quantize weight and activation to 2b BSL, and residual to 16b BSL, denoted as W2-A2-R16, following [15]. However, we find that direct quantization to low-precision ViT leads to a severe accuracy drop as shown in Table V. Therefore, we propose a two-stage training pipeline for SC-friendly low-precision ViT as shown in Fig. 6.

The first stage is the progressive quantization. Inspired by [14], [23], we start from the full-precision (FP) model and use a three-step procedure, i.e., $\text{FP} \rightarrow \text{W16-A16-R16} \rightarrow \text{W16-A2-R16} \rightarrow \text{W2-A2-R16}$ to achieve the low-precision model. In each step, we use the output from the last step as initialization. We also use KD to guide the training of low-precision model. We use the FP model as the teacher of the first step, and for the last two steps, we use W16-A16-R16 as the teacher, which is closer to the resulting model and provides sufficient information for the student to learn. The KD objective is

$$\text{Loss} = \ell_{KL}(Z_s, Z_t) + \beta \cdot \frac{1}{M} \sum_{i=1}^M \ell_{MSE}(S_i, T_i)$$

where ℓ_{KL} and ℓ_{MSE} are the Kullback-Leibler (KL) divergence loss and mean squared error (MSE) loss, respectively. Z_s and Z_t denote the logits of the teacher and student model

respectively. S_i and T_i denote the output of the i -th layer of the student model and teacher model, with a total of M layers. β is the coefficient balancing the KL loss and the MSE loss, which is set to 2 experimentally.

The second stage is approximate softmax aware fine-tuning. Considering the hardware efficiency, we replace the softmax in the W2-A2-R16 ViT with the iterative approximate softmax. However, this brings an accuracy degradation to the model. Thus we fine-tune the model after the replacement to adapt it to the approximate softmax, which will make up for the accuracy loss. So far, we have obtained SC-friendly low-precision ViT.

VI. EXPERIMENT

A. Experiment Setup

Hardware Evaluation: we implement the register transfer level (RTL) code of our SC designs and existing SC designs and then, synthesize them with Synopsys Design Compiler using TSMC 28nm technology library. We report the hardware metrics based on the synthesis results.

We evaluate different SC designs in terms of area, delay, and computation error. We also calculate the area-delay product (ADP) for hardware cost comparison. To evaluate computation errors, we collect the input vectors of softmax for each layer in ViT and generate test vectors sampled from the overall distribution. We calculate the mean average error (MAE) between the SC circuit outputs and the correct results.

Network Evaluation: we evaluate the proposed training method on a lightweight ViT, which has 7 layers and 4 heads on CIFAR10 and CIFAR100 datasets following [24]. We use learned step size quantization (LSQ) to quantize both weights and activations [25]. For the training settings, we use AdamW optimizer [26] with a momentum of 0.9. We set the batch size to 128. For the first stage in the training pipeline, we train the model for 300 epochs with an initial learning rate of 7.5×10^{-4} . For the second stage, we replace the softmax with iterative approximate softmax and fine-tune the model for 30 epochs with an initial learning rate of 5×10^{-6} .

B. Main Results

1) *SC Circuits Comparison:* We evaluate our proposed blocks for the GELU and softmax function. The design space for different approximate configurations of iterative approximate softmax is also explored.

GELU Block Comparison We focus on comparing the proposed gate-assisted SI with Bernstein polynomial-based design since it is the only baseline that can implement GELU in the negative input range as described in Section II-B.

While the Bernstein polynomial-based design requires numerous cycles for sequential computation, our design is fully parallel, enabling significant latency reduction. As shown in Table III, the 8b BSL gate-assisted SI achieves ADP reduction from $3.36 \times$ to $5.29 \times$ compared to the 1024b BSL baseline method with different polynomial terms. At the same time, we also reduce computation errors by 71.7% to 56.3%. If a larger computational error is allowed, we can further reduce the ADP by $4.15 \times$ from 1420 to 342 $\text{um}^2 \cdot \text{us}$. More ADP and MAE comparisons are also visualized in Fig. 7.

TABLE III
COMPARE AREA, DELAY, AREA-DELAY PRODUCT (ADP), AND MEAN AVERAGE ERROR (MAE) ACROSS DIFFERENT SC CIRCUITS FOR GELU.

Design	Area↓ (um^2)	Delay↓ (ns)	ADP↓ ($\text{um}^2 \cdot \text{ns}$)	MAE↓
Bernstein 4-term poly	58.2	81.92	4769	0.0548
polynomial 5-term poly	76.3	81.92	6254	0.0413
[18] 6-term poly	91.6	81.92	7506	0.0355
2b BSL	645.1	0.55	342	0.0410
Ours 4b BSL	1290.6	0.55	710	0.0252
8b BSL	2581.7	0.55	1420	0.0155

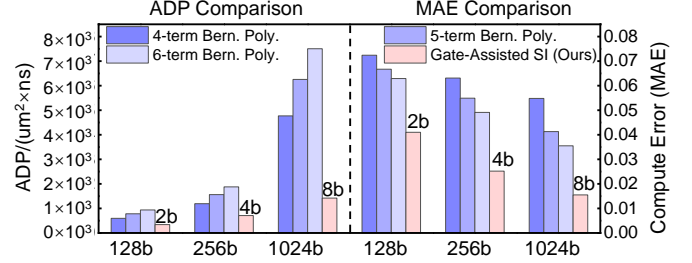


Fig. 7. GELU block comparison with different BSLs.

TABLE IV
COMPARE AREA, DELAY, AREA-DELAY PRODUCT (ADP), AND MEAN AVERAGE ERROR (MAE) ACROSS DIFFERENT SC CIRCUITS FOR SOFTMAX.

Design	Area↓ (um^2)	Delay↓ (ns)	ADP↓ ($\text{um}^2 \cdot \text{ns}$)	MAE↓
FSM [17] 128b BSL	1.26×10^4	328	4.14×10^6	0.108
256b BSL	1.26×10^4	655	8.28×10^6	0.103
1024b BSL	1.26×10^4	2621	3.31×10^7	0.099
$B_y = 4$	4.23×10^4	16.12	6.81×10^5	0.106
Ours $B_y = 8$	1.62×10^5	16.20	2.62×10^6	0.0766
$B_y = 16$	8.73×10^5	16.28	1.42×10^7	0.0427

Softmax Block Comparison For the proposed iterative approximate softmax circuit, we selected the design with $B_x = 4$ and evaluated different B_y s. We compared our softmax block with the FSM-based softmax design proposed in [17]. The number of inputs to the softmax function m is set to 64. As shown in Table IV, for the iterative approximate softmax with $B_y = 8$, it has reduced the ADP by $1.58 \times$ to $12.6 \times$ with 29.1% to 22.6% MAE reduction compared to FSM-based designs. Meanwhile, reducing B_y from 8 to 4 can further reduce the ADP by $3.85 \times$ or decrease 44.3% MAE.

Design Space Exploration We explore the design space of the softmax block by varying parameters listed in Table II and there are 2916 possible designs in total. For $B_x = 2$, we found 12 Pareto optimums, where ADP varies from $2.45 \times 10^5 \text{um}^2 \cdot \text{ns}$ to $1.89 \times 10^7 \text{um}^2 \cdot \text{ns}$ and MAE from 0.0098 to 0.0714. Similarly, there are 21 Pareto optimums in the design space of $B_x = 4$.

The Pareto frontier in the design space illustrates the flexibility of approximate designs in balancing circuit efficiency and computational accuracy.

2) *Network Accuracy Comparison:* As is shown in Table V, the baseline low-precision ViT without using our proposed training pipeline suffers from a large accuracy degradation even with KD. Our proposed progressive quantization strategy improves the accuracy by 32.99% and 21.4% on CIFAR10 and CIFAR100, respectively. The approximate softmax aware fine-

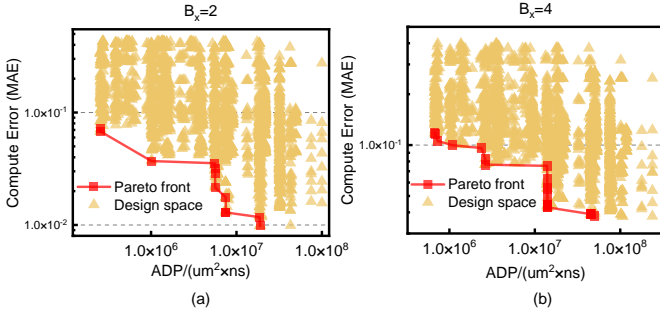


Fig. 8. Design space exploration for the iterative approximate softmax block for (a) $B_x = 2$ and (b) $B_x = 4$.

TABLE V

ACCURACY COMPARISON OF ViT AFTER USING PROGRESSIVE QUANTIZATION AND APPROXIMATE SOFTMAX AWARE FINE-TUNING.

Model	CIFAR10	CIFAR100
FP LN-ViT [24]	94.52	73.80
Baseline low-precision BN-ViT	58.13	45.76
BN-ViT + progressive quant	91.12	67.16
BN-ViT + progressive quant + appr	89.27	65.36
BN-ViT + progressive quant + appr-aware ft	90.79	66.18

TABLE VI

SC ACCELERATOR PERFORMANCE WITH DIFFERENT CONFIGURATIONS.

Configuration [B_y, s_1, s_2, k]	Softmax area (um ²)	*Accelerator area (um ²)	Accuracy	
			CIFAR10	CIFAR100
[4, 128, 2, 2]	3.15×10^4	4.24×10^6	89.72	63.51
[8, 32, 8, 3]	8.82×10^4	4.47×10^6	90.79	66.18
[16, 128, 16, 4]	4.65×10^5	6.04×10^6	91.07	66.63
[32, 128, 16, 4]	1.16×10^6	8.84×10^6	91.25	66.78

*In an accelerator, there are k softmax blocks to ensure the fully parallel.

tuning strategy improves the accuracy by 1.52% and 0.82% on the two datasets. With this two-stage training pipeline, our SC-friendly low-precision ViT achieves 32.66% and 20.42% higher accuracy than the baseline model on CIFAR10 and CIFAR100, respectively.

3) *Evaluation of ViT Accelerator*: We further conduct accelerator-level evaluations to show the impact of softmax blocks. Specifically, we select different softmax block configurations along the Pareto front and evaluate the area and accuracy impact. As shown in Table VI, our softmax block only takes a small portion of the total area, e.g., 1.48% for small computation BSLs and iterations. With the increase of BSLs and iterations, although the inference accuracy increases by more than 1.5%, the softmax block area increases drastically by more than 30 \times , leading to more than 2 \times total area overhead. Therefore, we might recommend choosing the configuration of [8, 32, 8, 3], which enables to achieve an accuracy of over 90% on CIFAR10 with only a marginal increase in total area compared to the configuration with minimum area.

VII. CONCLUSION

In this paper, we investigate the challenges of implementing ViT in SC and propose ASCEND, the first ViT acceleration with end-to-end SC. We propose novel SC circuit blocks for the GELU and the approximate softmax with a parameterized design space and a Pareto optimization. The proposed GELU

and softmax blocks achieve 56.3% and 22.6% error reduction compared to existing SC designs, respectively, and reduce the ADP by 5.29 \times and 12.6 \times , respectively. We also develop an SC-friendly low-precision ViT through a two-stage training pipeline including progressive quantization and approximate softmax aware fine-tuning, which significantly improves the accuracy over the baseline low-precision network.

In summary, the proposed ASCEND is accurate yet efficient. Its diverse set of approximation configurations enables flexible adaptation to meet the hardware efficiency and computational accuracy requirements of various applications.

REFERENCES

- [1] A. Dosovitskiy *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *Proc. ICLR*, 2021.
- [2] N. Carion *et al.*, “End-to-end object detection with transformers,” in *Proc. ECCV*, 2020.
- [3] J. Gu *et al.*, “Multi-scale high-resolution vision transformer for semantic segmentation,” in *Proc. CVPR*, 2022.
- [4] Z. Liu *et al.*, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proc. ICCV*, 2021.
- [5] Y. Zhang *et al.*, “When sorting network meets parallel bitstreams: A fault-tolerant parallel ternary neural network accelerator based on stochastic computing,” in *Proc. DATE*. IEEE, 2020.
- [6] K. Kim *et al.*, “Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks,” in *Proc. DAC*, 2016.
- [7] A. Ren *et al.*, “Sc-dcn: Highly-scalable deep convolutional neural network using stochastic computing,” in *Proc. ASPLOS*. ACM, 2017.
- [8] J. Li *et al.*, “Towards acceleration of deep convolutional neural networks using stochastic computing,” in *Proc. ASPDAC*. IEEE, 2017.
- [9] Z. Li *et al.*, “Heif: Highly efficient stochastic computing-based inference framework for deep neural networks,” *IEEE TCAD*, vol. 38, no. 8, pp. 1543–1556, 2018.
- [10] Y. Zhang *et al.*, “Accurate and energy-efficient implementation of non-linear adder in parallel stochastic computing using sorting network,” in *Proc. ISCAS*. IEEE, 2020.
- [11] P. Li *et al.*, “Using stochastic computing to implement digital image processing algorithms,” in *Proc. ICCD*. IEEE, 2011.
- [12] W. Zeng *et al.*, “Mpcvit: Searching for mpc-friendly vision transformer with heterogeneous attention,” in *Proc. ICCV*, 2023.
- [13] H. Bai *et al.*, “Binarybert: Pushing the limit of bert quantization,” in *Proc. ACL*, 2021.
- [14] Z. Liu *et al.*, “Bit: Robustly binarized multi-distilled transformer,” in *NeurIPS*, 2022.
- [15] Y. Hu *et al.*, “Accurate yet efficient stochastic computing neural acceleration with high precision residual fusion,” in *Proc. DATE*. IEEE, 2023.
- [16] Z. Yuan *et al.*, “Softmax regression design for stochastic computing based deep convolutional neural networks,” in *GLSVLSI*, 2017.
- [17] R. Hu *et al.*, “Efficient hardware architecture of softmax layer in deep neural network,” in *DSP*. IEEE, 2018.
- [18] W. Qian *et al.*, “Uniform approximation and bernstein polynomials with coefficients in the unit interval,” *European Journal of Combinatorics*, vol. 32, no. 3, pp. 448–463, 2011.
- [19] A. Naderi *et al.*, “Delayed stochastic decoding of ldpc codes,” *IEEE Transactions on Signal Processing*, vol. 59, 2011.
- [20] T.-H. Chen *et al.*, “Design of division circuits for stochastic computing,” in *Proc. ISVLSI*. IEEE, 2016, pp. 116–121.
- [21] V. Canals *et al.*, “A new stochastic computing methodology for efficient neural network implementation,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 551–564, 2015.
- [22] Q. Zhang *et al.*, “Morse-stf: Improved protocols for privacy-preserving machine learning,” *arXiv:2109.11726*, 2021.
- [23] B. Martinez *et al.*, “Training binary neural networks with real-to-binary convolutions,” *arXiv:2003.11535*, 2020.
- [24] A. Hassani *et al.*, “Escaping the big data paradigm with compact transformers,” *arXiv:2104.05704*, 2021.
- [25] S. K. Esser *et al.*, “Learned step size quantization,” *Proc. ICLR*, 2020.
- [26] I. Loshchilov *et al.*, “Decoupled weight decay regularization,” *Proc. ICLR*, 2019.