Research on High-Precision Stochastic Computing VLSI Structures for Deep Neural Network Accelerators

WU Jingguo[1], ZHU Jingwei[1], XIONG Xiankui[2,3], YAO Haidong[2,3], WANG Chengchen[2,3], CHEN Yun[1]

(1. Fudan University, Shanghai 200433, China;
2. State Key Laboratory of Mobile Network and Mobile Multimedia Technology, Shenzhen 518055, China;
3. ZTE Corporation, Shenzhen 518057, China)

DOI: 10.12142/ZTECOM.202404003

https://kns.cnki.net/kcms/detail/34.1294.TN.20241105.1346.002.html, published online November 5, 2024

Manuscript received: 2024-08-04

Abstract: Deep neural networks (DNN) are widely used in image recognition, image classification, and other fields. However, as the model size increases, the DNN hardware accelerators face the challenge of higher area overhead and energy consumption. In recent years, stochastic computing (SC) has been considered a way to realize deep neural networks and reduce hardware consumption. A probabilistic compensation algorithm is proposed to solve the accuracy problem of stochastic calculation, and a fully parallel neural network accelerator based on a deterministic method is designed. The software simulation results show that the accuracy of the probability compensation algorithm on the CIFAR-10 data set is 95.32%, which is 14.98% higher than that of the traditional SC algorithm. The accuracy of the deterministic algorithm on the CIFAR-10 dataset is 95.06%, which is 14.72% higher than that of the traditional SC algorithm. The results of Very Large Scale Integration Circuit (VLSI) hardware tests show that the normalized energy efficiency of the fully parallel neural network accelerator based on the deterministic method is improved by 31% compared with the circuit based on binary computing.

Keywords: stochastic computing; hardware accelerator; deep neural network

Citation (Format 1): WU J G, ZHU J W, XIONG X K, et al. Research on high-precision stochastic computing VLSI structures for deep neural network accelerators [J]. ZTE Communications, 2024, 22(4): 9 – 17. DOI: 10.12142/ZTECOM.202404003
Citation (Format 2): J. G. Wu, J. W. Zhu, X. K. Xiong, et al., "Research on high-precision stochastic computing VLSI structures for deep neural network accelerators," *ZTE Communications*, vol. 22, no. 4, pp. 9 – 17, Dec. 2024. doi: 10.12142/ZTECOM.202404003.

1 Introduction

Today, with the rapid development of high-speed data services such as the Internet and the Internet of Things, a large number of data interactions bring convenience to people's lives and at the same time, the constantly increasing data volume brings challenges to the efficiency of data processing algorithms and hardware performance. Although the integrated circuit industry follows Moore's Law, chip integration is getting higher and higher with the continuous reduction of process nodes, and chip performance also improves. However, as the traditional complementary metal oxide semiconductor (CMOS) process size is getting closer to the physical limit, new structures, new materials, and new lithography techniques cannot stop the argument that "Moore's Law is dead". The continuous reduction of chip feature size leads to many difficult problems in chip manufacturing, such as poor robustness and heat dissipation. These problems cannot be solved by integrating more transistors on the chip, at this time, it is necessary to find a new method to reduce the complexity of signal processing algorithms and circuit power consumption under the existing process conditions. At this time, stochastic computing (SC) comes into view again.

Traditional stochastic computing refers to a computational paradigm that employs randomness as a fundamental resource for information processing. The data are represented and manipulated probabilistically, often using bitstreams or random sequences to encode values. The weighted representation of stochastic computing is different from that of binary. Stochastic computing converts binary input into a probabilistic bit stream, also known as a stochastic sequence, according to a certain data format, and re-designs the corresponding basic unit circuit according to the data format, so that the original complex operation logic can be used to achieve the same purpose with simple logic, that is, the algorithm realized by stochastic computing has lower computational complexity. It can reduce the hardware resource cost required to implement the operation logic. Stochastic computing focuses on the number of "1" in the stochastic sequence, and does not pay much attention to the specific position of "1" and "0" in the generated stochastic sequence. Although the chip manufacturing process such as technology and soft error may bring about the phenomenon of a few bit errors, it will not have a great impact on the final result of stochastic computing. This is the high fault tolerance characteristic of stochastic computing. Since the performance of stochastic computing circuits is essentially anti-aging and is not affected by circuit topology and probabilistic coding, stochastic computing can provide a more relaxed circuit design space, which provides hope for the future application of emerging nanodevices. Stochastic computing not only reduces the complexity of circuit design but also reduces the requirement for device reliability. This suggests that stochastic computing is an alternative to the inherent reliability enhancement design of advanced technology nodes[1].

In 2021, LI et al. optimized stochastic sequence generation, used separate weights and activation memory to load their respective stochastic sequence generator buffers, spread the generation cost of activation flow through cross-row broadcast of activation values, and corrected correlation through training, thus bridging the accuracy gap between stochastic computing and fixed-point neural networks[2]. In 2022, HU et al.[3] proposed a complete stochastic computing architecture and realized the flow sheet, which maximized fault tolerance and robustness, achieved an energy efficiency of 198.9 TOPS/W and an area efficiency of 2 630 GOPS/mm^2, and reduced the accuracy loss by 70%. Ref. [3] shows the great potential of low-cost IoT neural network processors. In 2022, CHEN et al.[4] proposed a low-complexity bitstream expansion method to suppress the computation errors of stochastic computing and proposed a partition scheme with allocation decision to design hybrid stochastic binary computing multiplicative and additive units to improve the processing speed of bitstream with minimal overhead. In 2023, HU et al.[5] proposed a hybrid stochastic multiplier combining unipolar coding and bipolar coding to achieve a balance between high precision and low hardware consumption and proposed a stochastic accumulator parallel counter to attain high precision stochastic bit stream to binary conversion with low hardware consumption. At the same time, the finite state machine was used to realize the high-precision Relu function circuit design. In 2023, FRASSER et al.[6] used correlation and de-correlation to compute, and for the first time embedded a fully parallel convolutional neural network based on stochastic computing into a single FPGA chip, achieving better performance results than traditional binary logic and other stochastic computing implementations. In 2023, XIE et al.[7] proposed a new stochastic computing accelerator for convolutional neural networks, which utilized the nuclear parallelism of convolutional layers to reduce hardware area and energy consumption effectively.

However, the accuracy of traditional stochastic computing is not enough, which makes the final result worse. To improve the calculation accuracy, this paper proposes a probability compensation algorithm based on the relative error distribution of the traditional stochastic computing multiplier, which maps the data to the region with a small relative error through the function and then performs data inverse processing. The accuracy rate of the accelerator on the CIFAR-10 data set is 95.32%. In addition, a fully parallel neural network accelerator based on the deterministic method is designed, and the accuracy of the accelerator on the CIFAR-10 dataset is 95.06%. This design adopts TSMC 28 nm CMOS technology, and the energy efficiency is 1.371 TOPS/W.

The rest of the paper is organized as follows. Section 2 introduces the basic concepts and a brief review of SC. The third section gives the architecture design of the hardware accelerator. In Section 4, the experimental results of the probabilistic compensation algorithm and the accuracy, hardware evaluation, and comparison of all parallel neural network accelerators based on the deterministic method on multiple data sets are introduced. Finally, Section 5 draws the conclusion.

2 Background

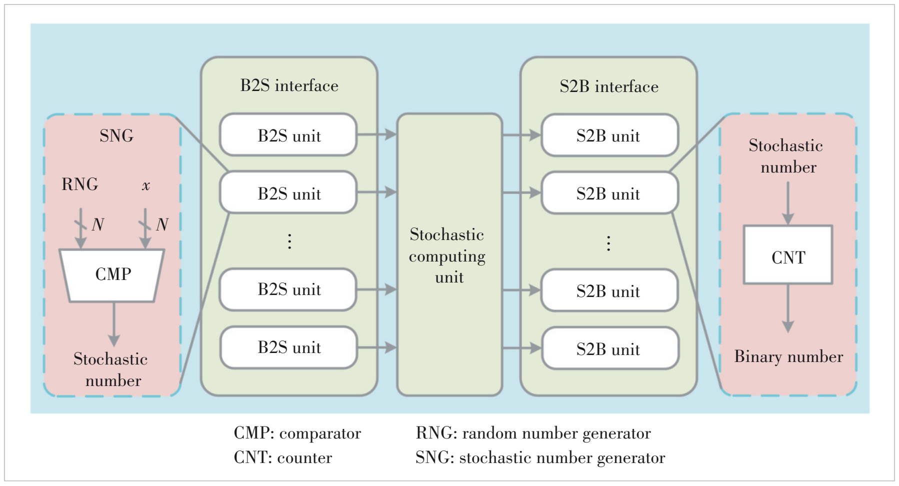
2.1 Stochastic Computing

Different from the binary weight representation method, stochastic computing is represented by converting the binary input into a finite-length stochastic bit stream, that is, the probability of the occurrence of "1" in the stochastic bit stream is represented by the corresponding binary value, which is the most common unipolar data format. For stochastic computing, each "1" in a stochastic bit stream has an equal weight. For a string of n-bit stochastic bit streams represented by the unipolar type, the corresponding binary value x is equal to the probability P_x of "1" appearing in the stochastic bit stream, so the representation range of unipolar type is [0, 1], and the data represented by each "1" is $1/N$, that is, the accuracy of data represented by the unipolar type is $1/N$. The research process of stochastic computing is summarized. The structure of the stochastic computing paradigm in Fig. 1 is obtained.

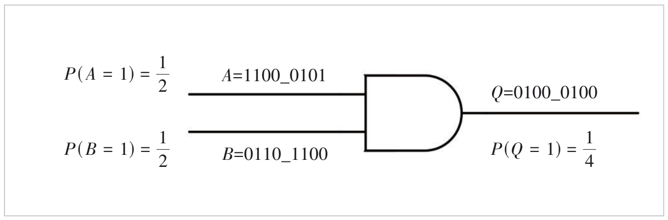
The structure of the stochastic computing paradigm is composed of three main parts. The first is the interface part of binary to a stochastic number, which is composed of a random number generator and a comparator. In the traditional structure, the random number generator uses the linear feedback shift register (LFSR) structure to generate random numbers. The second is the stochastic computing operation unit, which is represented by different basic gate units according to different data formats of stochastic computing. The last part is the stochastic number to the binary interface, which is generally composed of counters under the serial structure.

For stochastic sequences generated by unipolar data formats, multiplication is often performed with gates. An example of multiplication unit input and output for a unipolar data format is shown in Fig. 2.

The traditional stochastic computing multiplier is composed of a stochastic sequence generator, two inputs, a gate and a counter, and its structure is simple and clear. For the design



▲**Figure 1. Stochastic computing paradigm**



▲**Figure 2. A stochastic computing multiplier unit in a unipolar data format**

of this paper, int8 data are selected as the data input. Considering the data contain one symbol bit, int8 data can be split into symbol bits and numerical bits for multiplication. Therefore, the combination of seven-bit LFSR and seven-bit comparator is selected as the stochastic sequence generator. The stochastic sequence is generated by comparing the numerical bits of data input with the corresponding output values of LFSR. The counter is completed by the additional operation on the software side. When the stochastic sequence corresponding to the two inputs is 1, the resulting counter is increased by 1. For the seed selection of the random number generator, as long as the seeds are not the same, there is no fundamental impact on the stochastic computing multiplier. The current design does not consider the specific impact of other seed selections.

3 Accelerator Design

3.1 Probability Compensation Algorithm

The implementation of the stochastic multiplier is very simple, requiring only two inputs and a gate to implement binary multiplication logic, so it is often used to illustrate the simplicity of the logical unit of stochastic computing. However, simple logic often comes with certain disadvantages. Different from the binary accurate calculation, due to the randomness of stochastic computing, the calculation results also have stochastic properties, and the error of the calculation results is not uniform, resulting in poor performance in the performance test of the convolutional neural network built by the traditional stochastic computing multiplier, which is an important problem to be solved in this paper.

3.1.1 Error Analysis of Stochastic Computational Multiplier

The software side uses the pyLFSR library of Python to make the software side model closer to the hardware model. A random int8 data input error test was carried out for the completed traditional stochastic computing multiplier model. The input data was in the form of fixed-point numbers. The errors between the calculated results and theoretical results were measured by mean relative error (MRE), mean error (ME), and maximum relative error (ERR_max), and the corresponding calculation formula was shown as follows.

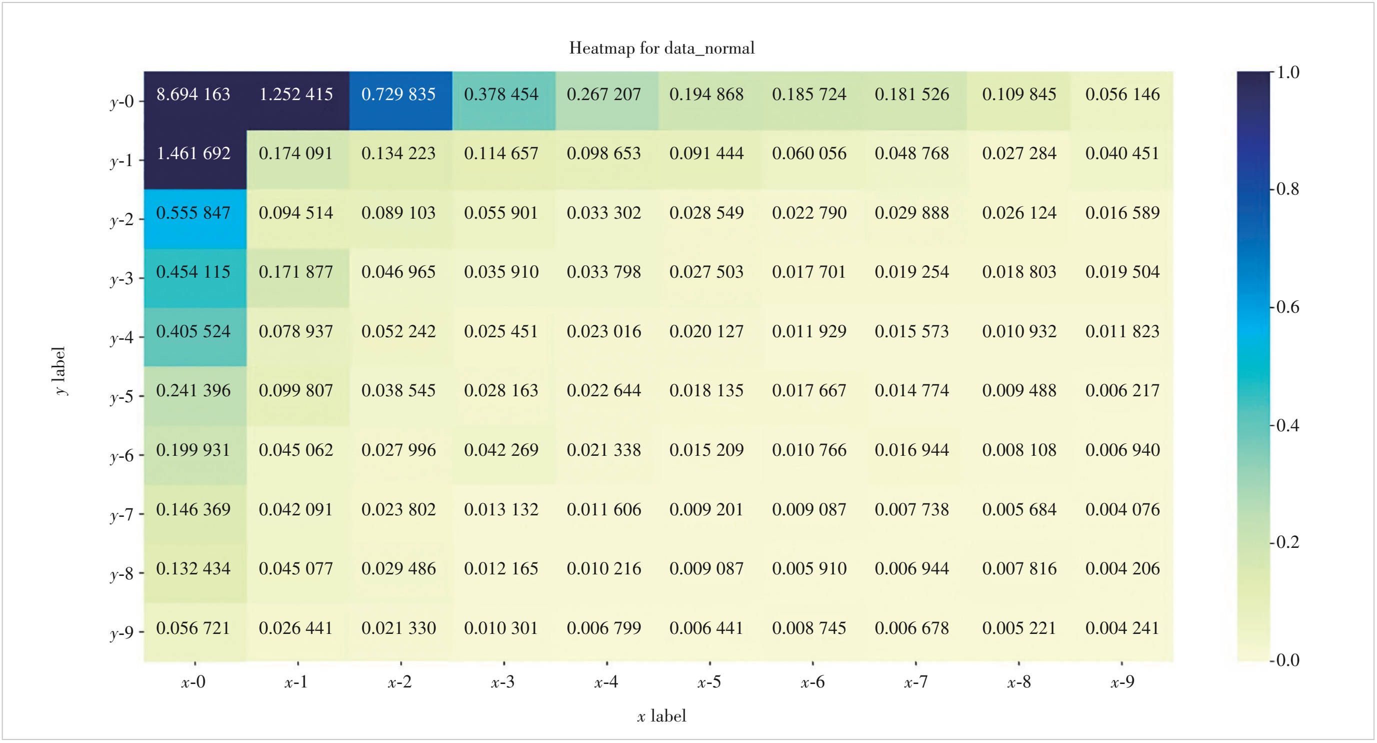
$$\text{MRE} = \frac{1}{N} \cdot \sum_{i=1}^{N} \left| \frac{x_i - x\_\text{real}_i}{x\_\text{real}_i} \right|, \tag{1}$$

$$\text{ME} = \frac{1}{N} \cdot \sum_{i=1}^{N} \frac{x_i - x\_\text{real}_i}{x\_\text{real}_i}, \tag{2}$$

$$\text{ERR\_max} = \max\left(\frac{x_i - x\_\text{real}_i}{x\_\text{real}_i} \right), \tag{3}$$

where N is the total number of test data, x_i is the calculation result of the stochastic computing multiplier, and x_real_i is the theoretical calculation result.

Random number generator seeds are selected as [1, 1, 0, 0, 1, 1, 1] and [1, 0, 1, 1, 1], and fixed-point random data input is generated by the randint function in the Python library. The results of the 10 000 random samples simulation show that MRE and ME of a traditional stochastic multiplier are 15.47% and 10.23%, and ERR_max is 127. To better understand the distribution of relative errors and make appropriate adjustments, the input data are divided into 10 intervals according to the seven-bit numerical bits on average. There are 100 two-dimensional intervals corresponding to the two inputs in pairs. Each interval generates 1 000 random fixed points within the interval range, and the simulation of relative error distribution is carried out. The resulting relative error distribution is shown in Fig. 3. To facilitate observation, the squares with small relative errors in the thermal map are filled with a light color system, and the squares with large relative errors in the thermal map are filled with a dark color system. The darker the color is, the larger the relative errors are. The color and



▲Figure 3. Heatmap of the relative computational error distribution of the traditional stochastic computing multiplier

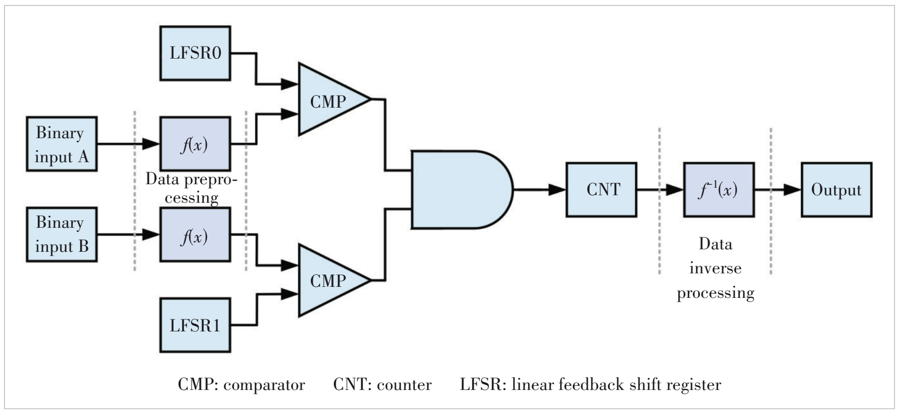
corresponding value are given in the legend on the right of the figure, and the relative error value of the square is marked in each square. By observing the thermal distribution diagram of relative error, it is found that the dark grid is concentrated in the upper left corner of the thermal map, that is, when the two input data are both small, the relative error of the calculated result is large, which is consistent with the previous ERRmax. In heat maps, the relative error is mostly less than 4%, and the further you go to the left, the greater the relative error is. Theoretically speaking, since x_real_i is in the denominator of the relative error formula, when the absolute error between x_real_i and x_i is at the same magnitude, the smaller x_real_i is, the greater the relative error will be. The relative error distribution of the thermal map is in agreement with the theoretical analysis results.

3.1.2 Research on Compensation Mechanism of Traditional Stochastic Computing Multiplier

To eliminate the serious influence of the local relative error on the whole relative error shown in the thermal map, the compensation mechanism of the stochastic computing multiplier is studied. Considering that the position with the greatest relative error appears on the far left and top, and the worst case is concentrated in the upper left corner, that is, the relative error of the calculation result is larger when one input datum is small, and the relative error of the calculation result is large when the two input data are small at the same time, which explores a situation where the data are mapped from the upper left corner to the lower right corner by using accurate intermediate calculation. After the calculation, the accuracy compensation is carried out by reflecting in the upper left corner. The compensation method is called probability compensation, which means increasing the probability of the data interval with a small relative error in the calculation to improve the overall data accuracy. Fig. 4 is the flow chart of the probability compensation algorithm.

In Fig. 4, the probability compensation algorithm adds two steps of data preprocessing and data inverse processing based on traditional stochastic computing. After data preprocessing, binary input enters the comparator together with the random numbers generated by the random number generator LFSR for comparison. After stochastic computing and multiplication unit, the preprocessed stochastic sequence is converted into binary for data inverse processing. Utilizing data mapping, the probability compensation algorithm converts the data involved in the operation to the data region with high accuracy, to compensate for the accuracy of the traditional stochastic computing and multiplication algorithm.

For probability compensation, a function $f(x)$ should be found to satisfy certain conditions, and the data should be preprocessed by function mapping. To ensure fairness in mapping the two input data, the same function $f(x)$ is used for mapping. The data representation range of unipolar stochastic computing is [0, 1], that is, the value range of the two input data is [0, 1]. Since the input data with large relative error is concentrated in the data interval close to 0, the function $f(x)$ should



▲**Figure 4. Flow chart of a probability compensation algorithm**

map the data close to 0 to the data interval close to 1. The input data with small relative error should remain unchanged or change in a small amplitude as much as possible, so it is necessary to find a function, of which the domain and range are both [0, 1]. At the same time, since the inverse function of $f(x)$ needs to be used for data inverse processing, the function $f(x)$ must meet the requirement of the existence of an inverse function, and the influence of two input data mappings must be offset at the same time. To sum up, the function required for probability compensation should meet the following requirements.

1) The domain of $f(x)$is [0, 1], and the range is also [0, 1];

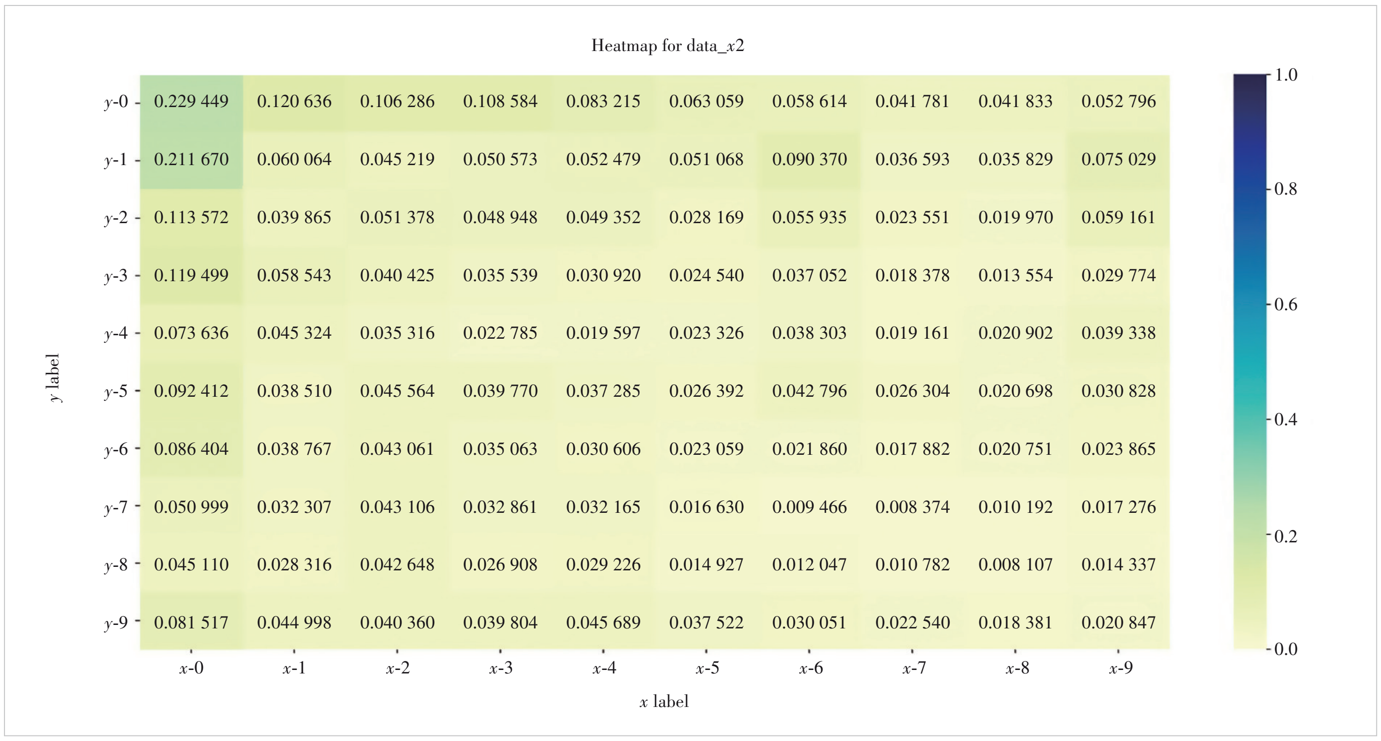
2) $f(x)$ is monotone in the domain of definition, and most of the values are in the numerical interval with small relative error, to meet the requirement of probability mapping and the existence of inverse function;

3) $f(x, y) = f(x) \times f(y)$, where x and y represent two input data. If this condition is met, the inverse function can simultaneously offset the influence of the mapping of two input data.

To find a function that meets the requirements, the entry point lies in the requirement that $f(x, y) = f(x) \times f(y)$. After deliberation, the power function of x can meet this requirement, so based on the power function, the function that meets the other two requirements is found. To satisfy that most of the values of $f(x)$ are in the numerical interval with small relative error, the time function $f(x)$ should be above $f(x)=x$. Since the domain contains the point x=0, the satisfying function $f(x)$ is shown in Eq. (4).

$$f(x) = x^a, a \in (0, 1). \tag{4}$$

To facilitate subsequent data reverse processing and consider the complexity problem, the corresponding function a=1/2 is selected for simulation. The algorithm flow is consistent with that of Fig. 4. Random data with the same relative error distribution thermal map as that of the traditional stochastic computing multiplier in Fig. 3 are used, and the thermal map results obtained are shown in Fig. 5. The preprocessing func-



	x-0	x-1	x-2	x-3	x-4	x-5	x-6	x-7	x-8	x-9
y-0	0.229 449	0.120 636	0.106 286	0.108 584	0.083 215	0.063 059	0.058 614	0.041 781	0.041 833	0.052 796
y-1	0.211 670	0.060 064	0.045 219	0.050 573	0.052 479	0.051 068	0.090 370	0.036 593	0.035 829	0.075 029
y-2	0.113 572	0.039 865	0.051 378	0.048 948	0.049 352	0.028 169	0.055 935	0.023 551	0.019 970	0.059 161
y-3	0.119 499	0.058 543	0.040 425	0.035 539	0.030 920	0.024 540	0.037 052	0.018 378	0.013 554	0.029 774
y-4	0.073 636	0.045 324	0.035 316	0.022 785	0.019 597	0.023 326	0.038 303	0.019 161	0.020 902	0.039 338
y-5	0.092 412	0.038 510	0.045 564	0.039 770	0.037 285	0.026 392	0.042 796	0.026 304	0.020 698	0.030 828
y-6	0.086 404	0.038 767	0.043 061	0.035 063	0.030 606	0.023 059	0.021 860	0.017 882	0.020 751	0.023 865
y-7	0.050 999	0.032 307	0.043 106	0.032 861	0.032 165	0.016 630	0.009 466	0.008 374	0.010 192	0.017 276
y-8	0.045 110	0.028 316	0.042 648	0.026 908	0.029 226	0.014 927	0.012 047	0.010 782	0.008 107	0.014 337
y-9	0.081 517	0.044 998	0.040 360	0.039 804	0.045 689	0.037 522	0.030 051	0.022 540	0.018 381	0.020 847

▲**Figure 5. Heatmap of the relative computational error distribution of the stochastic computing multiplier with probability compensation**

tion is named according to the preprocessing function, where a=1/2 is denoted as probability compensation. After using the idea of probability compensation, the dark grid in the thermal map is completely eliminated, which means that the relatively large relative error value is perfectly compensated, and the feasibility of the idea of probability compensation is verified.

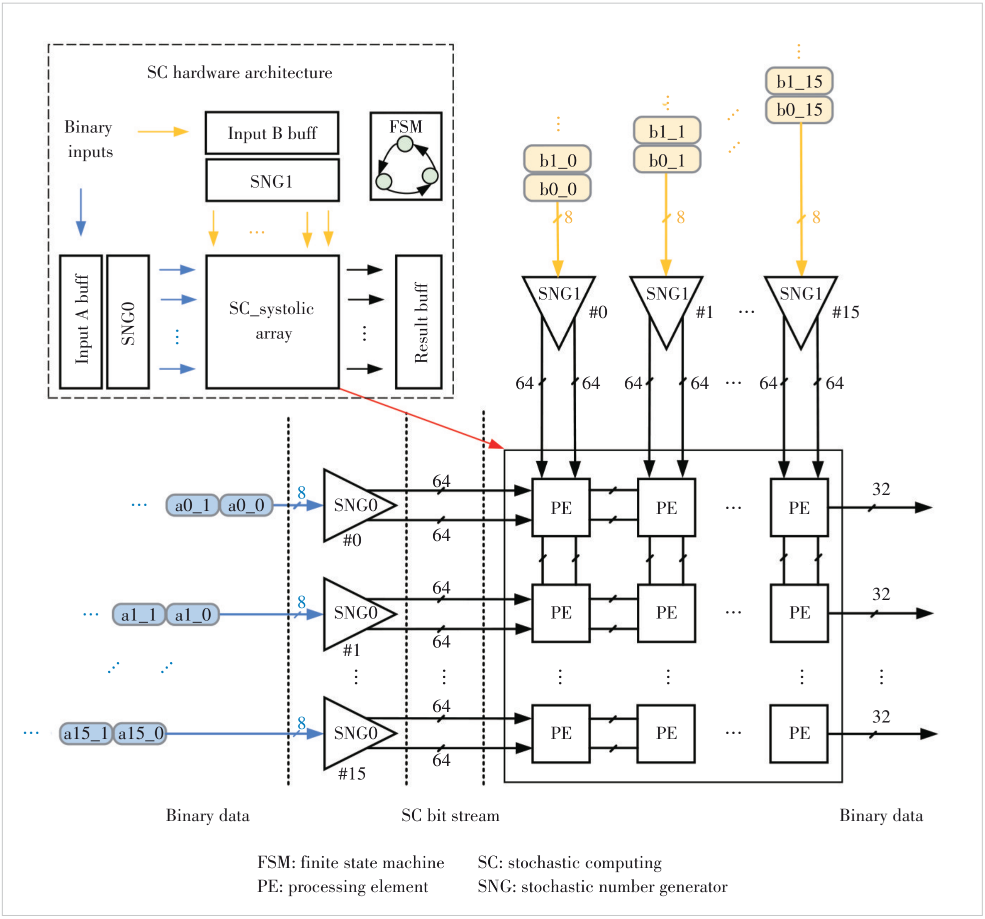
3.2 Fully Parallel Accelerator Hardware Design

3.2.1 Accelerator Architecture

In the aspect of the fully parallel accelerator, the accuracy of matrix multiplication is improved based on the deterministic method. As shown in Fig. 6, the fully parallel stochastic computing systolic array accelerator consists of 7 modules: input A buff, input B buff, stochastic number generator 0 (SNG0), stochastic number generator 1 (SNG1), finite state machine (FSM), result buff, and SC_systolic array.

These modules are divided into three parts: the first part is composed of input A buff, input B buff, SNG0, and SNG1, which mainly completes the conversion of binary elements that need to be calculated in matrices A and B into the bit stream required for stochastic computing. The second part is composed of the SC_systolic array and result buff. This part is the core computing unit in the entire hardware structure, which multiplies the stochastic computing bit stream that completes accumulation inside the unit, and finally puts the result into the result buff for caching. The third part is FSM, which completes the generation of data flow control signals, such as calculation start signals, result shift signals, and calculation end signals.

Similar to the serial accelerator architecture, the fully parallel accelerator proposed in this paper organizes all processing element (PE) units into a systolic array in the form of input and weight flow and partial product immobility. Input data enter the systolic array in a step form from left to right. PE units in the same row share the same stochastic computing parallel bit stream (from matrix A). This parallel bit stream will be passed one by one in the same row of PE cells on a clock cycle. The weight data also enter the systolic array in the form of a ladder from top to bottom, and the processing units in the same column share the same stochastic computing parallel bit stream (from matrix B), which will be passed one by one in the PE cells within the same column in the number of clock cycles. The final calculation result is controlled by a shift control signal and flows between PE units on a clock cycle, and is finally transmitted successively to the result buff.



▲Figure 6. Hardware architecture of the fully parallel stochastic computing of the systolic array

3.2.2 PE Unit and Systolic Array Design

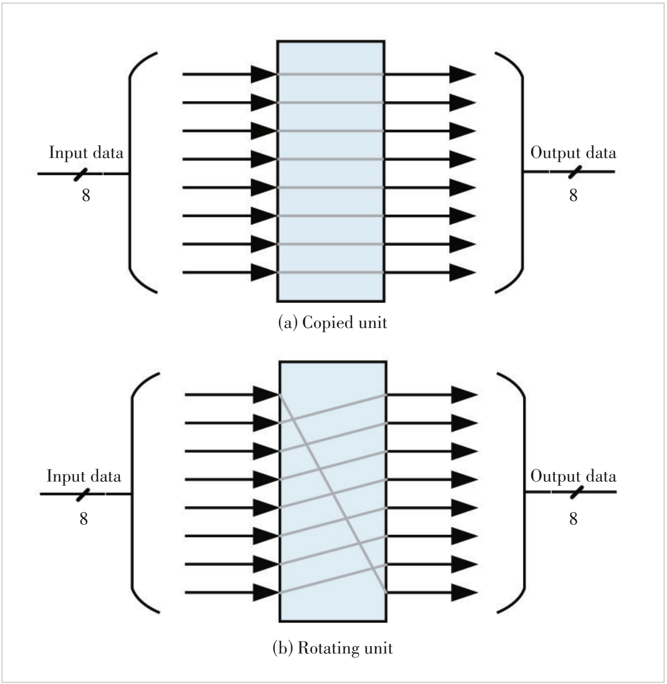
The PE unit in the fully parallel stochastic computing systolic array receives the parallel bit stream generated by the deterministic method and completes the dot product operation of the stochastic computing bit stream. In addition to the PE unit directly connected to the stochastic sequence generator, the remaining PE units receive the elements, the result, and the shift control signal from the previous PE unit. Ten internal registers are used to cache data and intermediate results and control signals.

In a fully parallel scheme, the multiplication of PE cells is performed using the partial product form, and four stochastic parallel bit streams, A_high_sc, A_low_sc, B_high_sc, and B_low_sc, are generated by a stochastic sequence generator. After each stochastic parallel bit stream is matched with a stochastic parallel bit stream generated by an-other matrix, the number of "1" in the phase and the result are calculated through an addition tree structure, where the addition tree used is equivalent to 64 inputs, and each input is a 1-bit full adder. After the results of the full adder are shifted, the final calculation result is obtained. The shift signal is shift controlled and sent by the FSM to determine when the final cumulative results inside the PE unit will be transmitted. When the processing unit PE has not completed the calculation, the shift signal is "0", and the PE unit is responsible for receiving the result of the previous unit and passing it to the next unit. When the PE unit completes the final calculation, the shift signal is set to "1" and the final calculation results are transmitted in turn.

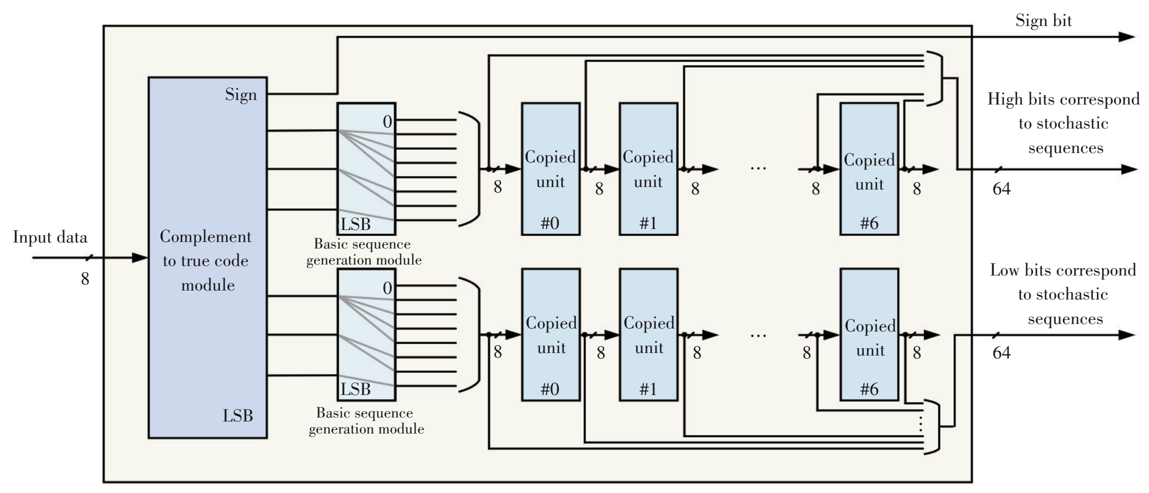
3.2.3 Stochastic Number Generator Design

The stochastic sequence generator used in the fully parallel scheme uses a deterministic method to generate a stochastic computing parallel bit stream. The data in int8 format are divided into a 1-bit symbol bit and two 3-bit numerical bits by bit segmentation, and the lowest 1-bit numerical bit is discarded to match the partial product allocation, which is recorded as the round bit segmentation method. Fig. 7 shows the internal structure diagram of SNG in the hardware design of a fully parallel accelerator. When two 3-bit numerical bits enter the basic sequence generation module, each 3-bit numerical bit is copied by binary weight, and the highest bit is fixed as "0", because the results generated by the copied and rotating methods are independent of the sorting method of the basic stochastic sequence. A base stochastic sequence of 8-bit without introducing additional hardware consumption is generated.

One datum in the int8 format corresponds to two basic stochastic sequences. In the matrix multiplication operation, the int8 input data in matrix A correspond to the internal operation of SNG0, and the two basic stochastic sequences generated in SNG0 are continuously copied 7 times using the copied unit. The high 64-bit width corresponds to the stochastic parallel bit stream A_high_sc, and the low bits correspond to the stochastic parallel bit stream A_low_sc. The internal line connection of the copied unit is shown in Fig. 8a. The input data of int8 in matrix B correspond to the internal operation of SNG1. SNG1 uses the rotating unit to replace the copied unit in SNG0, and the rest are completely consistent with SNG0. The SNG1 internal structure diagram is no longer listed here. The two basic stochastic sequences generated in



▲Figure 8. Copied and rotating units


Sign bit
Sign
Complement to true code module
LSB
Input data
8
0
LSB
Basic sequence generation module
Copied unit #0
Copied unit #1
Copied unit #6
8
...
High bits correspond to stochastic sequences
64
Low bits correspond to stochastic sequences
64

LSB: least significant bit

▲Figure 7. Fully parallel random number generator

SNG1 are rotated 7 times by the rotating unit to obtain a 64-bit high corresponding to a stochastic parallel bit stream B_high_sc and a 64-bit low corresponding to a stochastic parallel bit stream B_low_sc. The internal line connections of the rotating unit are shown in Fig. 8b. Finally, four stochastic parallel bits are generated and transmitted into the systolic array. The generation process only changes the order of connections and the number of replicated connections without introducing additional hardware consumption.

4 Experiment and Analysis

4.1 Software Simulation Results and Analysis

The designed two accelerators are applied to the neural network through the img2col algorithm. The seeds used for each random number generator are the same as previously mentioned. The network used is Resnet 18[8], and the quantization bit is 1-bit symbol bit plus 7-bit numerical bit. The MNIST[9] and the CIFAR-10[10] data sets are used for testing, and the test results obtained are shown in Table 1.

In the MNIST data set, there is no significant difference between schemes. The accuracy of all schemes except traditional schemes can reach more than 99% in the MNIST data set, and the accuracy of traditional SC schemes can also reach 98.56%. When tested on the more complex CIFAR-10 dataset, the accuracy of the traditional SC scheme is only 80.34%, 15% lower than that of binary. The precision is improved to 95.32% by the $\sqrt{x}$ probability compensation SC scheme and 95.06% by the deterministic SC scheme.

4.2 Back-End Implementation Results and Analysis

The layout and cabling of the TSMC 28 nm process with process nodes are completed by using the layout tool IC compiler (ICC) and the core voltage is 0.9 V. The layout parameters achieved are shown in Table 2.

According to the process node scaling method[11], the 65 nm process node used by the Eyeriss v2 accelerator[12] is normalized to the 28 nm process node used in this work, and the clock frequency scaling parameters are shown in Eq. (5).

▼Table 1. Test results of neural network accelerator application based on stochastic computing

DataSet	Method	Network Accuracy/%
MNIST	Binary	99.55
	Traditional SC	98.56
	$\sqrt{x}$ Probability compensation SC	99.50
	Deterministic method SC	99.43
CIFAR-10	Binary	95.52
	Traditional SC	80.34
	$\sqrt{x}$ Probability compensation SC	95.32
	Deterministic method SC	95.06

SC: stochastic computing

▼Table 2. Layout parameters of the neural network accelerator based on stochastic computing

Method	Type	Clock Frequency/MHz	Area/mm^2	Power/mW
Traditional SC	Uncompensated	1 000	0.104	81.3
Fully parallel	Deterministic method	313	0.601	116.7

SC: stochastic computing

$$N_c = \frac{65}{28}. \tag{5}$$

Load capacitance scaling parameters are shown in Eq. (6).

$$N_{\mathrm{CL}} = \frac{65}{28}. \tag{6}$$

Dynamic power consumption is the main power consumption in high-speed chips, and the dynamic power consumption formula of switches is shown in Eq. (7).

$$P = V^2 \cdot C_{\mathrm{Load}} \cdot f. \tag{7}$$

Therefore, power scaling parameters are shown in Eq. (8).

$$N_P = \left(\frac{0.9}{1.2}\right)^2 \cdot \frac{28}{65} \cdot \frac{65}{28} = 0.562\,5. \tag{8}$$

The energy efficiency ratio formula is shown as Eq. (9).

$$\mathrm{GOPS/W} = \frac{\text{calculations per second}}{P}. \tag{9}$$

The energy efficiency ratio corresponding to the designed accelerator can be obtained according to Eq. (9). By combining Eqs. (5), (8), and (9), the normalized parameters of the energy efficiency ratio can be calculated as shown in Eq. (10).

$$N = \frac{1}{0.562\,5} \times \frac{65}{28} = 4.126\,98. \tag{10}$$

Table 3 compares the results of the energy efficiency ratio between the neural network accelerator implemented in this work and the Eyeriss v2 accelerator[12], in which the energy efficiency ratio in the Eyeriss v2 accelerator is normalized. It can be seen from the results of Table 3 that compared with the normalized energy efficiency ratio of the Eyeriss v2 accelera-

▼Table 3. Comparsion of stochastic computing DNN implementations with other very large scale integration circuit (VLSI) deep neural networks (DNN)

Accelerator	Type	Process Node/nm	EER (GOPS/W)	Normalized EER (GOPS/W)
Eyeriss v2[12]	Binary computing	65	253.2	1 045.0
Classic serial	Uncompensated	28	6 297.7	6 297.7
Fully parallel	Deterministic method	28	1 371.0	1 371.0

EER: energy efficiency ratio

tor, the fully parallel stochastic computing neural network accelerator based on the deterministic method has improved by 31%. Although the normalized energy efficiency ratio of traditional stochastic computing neural network accelerators is the highest, the data set of this scheme is less accurate and should not be compared with.

5 Conclusions

In this paper, a probability compensation algorithm is proposed based on the relative error distribution of the traditional stochastic multiplier. The accuracy of the accelerator on the CIFAR-10 dataset is 95.32%. In addition, a fully parallel neural network accelerator based on the deterministic method is designed, and the accuracy of the accelerator on the CIFAR-10 dataset is 95.06%. This design adopts TSMC 28 nm CMOS technology, and the energy efficiency is 1.371 TOPS/W. Through hardware and software evaluation, the implementation results show that the proposed design is superior to the hardware implementation of DNN based on traditional binary computing logic in terms of energy efficiency ratio, and the network accuracy is superior to the traditional SC-DNN implementation.

References

[1] ZHANG Z D, WANG R S, ZHANG Z, et al. Circuit reliability comparison between stochastic computing and binary computing [J]. IEEE transactions on circuits and systems II: express briefs, 2020, 67(12): 3342 – 3346. DOI: 10.1109/tcsii.2020.2993273

[2] LI T M, ROMASZKAN W, PAMARTI S, et al. GEO: Generation and execution optimized stochastic computing accelerator for neural networks [C]//Proceedings of 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2021: 689 – 694. DOI: 10.23919/date51398.2021.9473911

[3] HU Y X, ZHANG Y W, WANG R S, et al. A 28-nm 198.9-TOPS/W fault-tolerant stochastic computing neural network processor [J]. IEEE solid-state circuits letters, 2022, 5: 198 – 201. DOI: 10.1109/lssc.2022.3194954

[4] CHEN Z Y, MA Y F, WANG Z F. Hybrid stochastic-binary computing for low-latency and high-precision inference of CNNs [J]. IEEE transactions on circuits and systems I: regular papers, 2022, 69(7): 2707 – 2720. DOI: 10.1109/tcsi.2022.3166524

[5] HU S, HAN K N, HU J H. High performance and hardware efficient stochastic computing elements for deep neural network [C]//Proceedings of 6th World Conference on Computing and Communication Technologies (WCCCT). IEEE, 2023: 181 – 186. DOI: 10.1109/wccct56755.2023.10052402

[6] FRASSER C F, LINARES-SERRANO P, DE LOS RÍOS I D, et al. Fully parallel stochastic computing hardware implementation of convolutional neural networks for edge computing applications [J]. IEEE transactions on neural networks and learning systems, 2023, 34(12): 10408 – 10418. DOI: 10.1109/tnnls.2022.3166799

[7] XIE Z P, YUAN C Y, LI L K, et al. Energy-efficient stochastic computing for convolutional neural networks by using kernel-wise parallelism [C]//Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2023: 1 – 5. DOI: 10.1109/iscas46773.2023.10181378

[8] HE K M, ZHANG X Y, REN S Q, et al. Deep residual learning for image recognition [C]//Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016: 770 – 778. DOI: 10.1109/cvpr.2016.90

[9] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition [J]. Proceedings of the IEEE, 1998, 86(11): 2278 – 2324. DOI: 10.1109/5.726791

[10] KRIZHEVSKY A. Learning multiple layers of features from tiny images [EB/OL]. (2012-05-20) [2024-08-01]. https://www.researchgate.net/publication/265748773_Learning_Multiple_Layers_of_Features_from_Tiny_Images

[11] NISHI Y, DOERONG R. Handbook of semiconductor manufacturing technology [M]. Boca Raton, USA: CRC Press, 2008

[12] CHEN Y H, YANG T J, EMER J S, et al. Eyeriss v2: a flexible accelerator for emerging deep neural networks on mobile devices [J]. IEEE journal on emerging and selected topics in circuits and systems, 2019, 9(2): 292 – 308. DOI: 10.1109/JETCAS.2019.2910232

Biographies

WU Jingguo received his BS degree in microelectronics science and engineering from Xi'an Jiaotong University, China in 2021. He is currently pursuing for a master's degree at Fudan University, China. His main research interest is stochastic computing.

ZHU Jingwei is a PhD candidate in the School of Microelectronics, Fudan University, China. He received his MASc degree from Fudan University. His current research interests include neuromorphic computing, spiking neural networks and NoC.

XIONG Xiankui graduated from University of Electronic Science and Technology of China. He is the chief architect of the Wireless Department and the leader of the Prospect Team of the Smart Computing Technical Committee, ZTE Corporation. He has been engaged in long-term research on computing systems and architectures, advanced computing paradigms, and heterogeneous computing accelerators. He has led the system architecture design of the ZTE ATCA advanced telecom computing platform, server storage platform, smart NIC, and AI accelerator.

YAO Haidong graduated from the University of Science and Technology of China, and he is a senior expert in the field of wireless communications at ZTE Corporation. He is mainly engaged in the research and design of deep learning, large language model network architecture, and compilation and conversion technology.

WANG Chengchen graduated from the Department of Precision Instrument, Tsinghua University, China and is now working in ZTE Corporation. His research directions include computing in memory, optical calculation, and probability calculation.

CHEN Yun (chenyun@fudan.edu.cn) received her BSc degree from University of Science and Technology of China in 2000, and PhD degree from Fudan University, China in 2007. She joined Fudan University at the same year, where she has been a professor with the State Key Laboratory of ASIC and Systems. She has published more than 60 articles in renowned international journals and conferences and applied for more than 20 patents. Her research interests include baseband processing technologies for wireless communication and ultralow power FEC IC design. Pro. CHEN is a member of the Steering Committee of SIPS and the ASICON Technical Committee. She serves as a TPC Member for ASSCC. She also serves as the Chair Secretary for the Shanghai Chapter of IEEE SSCS, and the Co-Chair for the Circuit System Division, the Chinese Institute of Electronics. She is a senior member of IEEE.