HOKKAIDO UNIVERSITY

Title	Digital implementation of a multilayer perception based on stochastic computing with learning function
Author(s)	Sasaki, Yoshiaki; Muramatsu, Seiya; Nishida, Kohei et al.
Citation	Nonlinear theory and its applications, IEICE, 13(2), 324-329 https://doi.org/10.1587/nolta.13.324
Issue Date	2022
Doc URL	https://hdl.handle.net/2115/85558
Rights	Copyright ©2022 The Institute of Electronics, Information and Communication Engineers
Type	journal article
File Information	13_324.pdf

Hokkaido University Collection of Scholarly and Academic Papers : HUSCAP

NOLTA, IEICE

Paper

Digital implementation of a multilayer perceptron based on stochastic computing with online learning function

Yoshiaki Sasaki [1 a)], *Seiya Muramatsu* [1], *Kohei Nishida* [2], *Megumi Akai-Kasaya* [3,4], *and Tetsuya Asai* [3]

[1] *Graduate School of IST, Hokkaido University*

[2] *Faculty School of Engineering, Hokkaido University*

[3] *Faculty School of IST, Hokkaido University, Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan*

[4] *Department of Chemistry, Graduate School of Science, Osaka University 1-1 Machikaneyama, Toyonaka, Osaka 560-0043, Japan*

[a)] *sasaki.yoshiaki.3s@ist.hokudai.ac.jp*

Received October 18, 2021; Revised December 18, 2021; Published April 1, 2022

Abstract: Stochastic Computing (SC)[2] is a probability-based computing method, which enables the performance of various operations with a small number of logic gates (i.e., low power) in exchange for high accuracy. Using SC for edge artificial intelligence (AI) integrated circuits can help circumvent the limitations inherent in the power and area required for edge AI.

In this study, a three-layered Neural Network (NN) is presented with an online learning function that introduces pseudo-activation, pseudo-subtraction, and imperfect addition into the SC framework. This method may expand the options for edge AI integrated circuits using SC.

Key Words: stochastic computing, machine learning, neural networks, edge AI integrated circuit

1. Introduction

Artificial intelligence (AI) has made remarkable progress in recent years; however, simultaneously, the cost of sum-of-products operations has substansially increased. In particular, the cost barrier is significantly large for edge AI integrated circuits because they have significant limitations in terms of chip area and power consumption. Therefore, edge AI integrated circuits are required to be as efficient as possible.

Stochastic computing (SC) was first proposed in the 1960s, and is an arithmetic method that significantly reduces the circuit area by adopting probability (Ref. [1]). It remained obscure for a considerable time owing to the variability in computational results caused by the nature of probability. However, recently SCs have been demonstrated to be applicable to various applications such as image

Nonlinear Theory and Its Applications, IEICE, vol. 13, no. 2, pp. 324–329 ©IEICE 2022 DOI: 10.1587/nolta.13.324

processing and deep learning, where a certain degree of error is permissible in the calculation results; therefore, we can expect to benefit from the introduction of SC. Addition and subtraction are difficult in SC-based operations. In a previous study, edge AI integrated circuits with SC required decoding at each layer of the layered network, thereby resulting in severely high computational cost. In addition, there have been no proposals for edge AI integrated circuits with learning functions or the introduction of arithmetic circuits for learning.

In this study, we propose a novel method for edge AI based on the assumption of the input to the activation function. Here, we propose a method to realize addition and subtraction without decoding by assuming the input to the activation function. We also propose a digital implementation of a three-layer perceptron with a learning function, including a digital implementation of a three-layer perceptron with a learning function that can learn and infer simple linear regression problems, and non-linear regression problems.

2. Stochastic computing

2.1 Numeric representation

In SC, a numerical value is represented by the ratio of logical 1s in a bit sequence. When an n-bit value E is adopted for arithmetic operations, a random n-bit value R is generated for every clock cycle. Subsequently, numerical conversion is performed by comparing E and R with a digital comparator that outputs logic 1 when $R < E$, and logic 0 when $R \geq E$. The bit sequence $X = \{X_1, X_2, ..., X_L\}$ is expressed by the following equation:

$$p(X = 1) = E/L. \tag{1}$$

In SC, operations are performed using these converted numbers and various logic circuits. A general

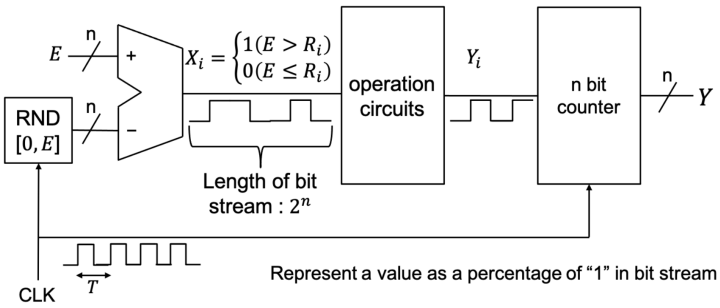


Fig. 1: Conceptual diagram of SC.

digital multiplier comprises a large number of adders; however, in SC, it is possible to realize a multiplier with only a single AND gate. Assuming that the input bit strings are X_1 and X_2, and the output bit string is Y, the relationship between the probability of occurrence of logic 0 and logic 1 at the input and output of the AND gate is expressed in the following equation:

$$Y = X_1 \& X_2 \tag{2}$$

$$p(Y = 1) = p(X_1 = 1) \times p(X_2 = 1). \tag{3}$$

2.2 Two types of adders

There are two types of additions in SC: weighted and imperfect additions.

First, in the SC framework, the operation using a multiplexer is called weighted addition. In general digital adders, it is necessary to arrange all adders with the same number of bits as the number of values to be added together; however, in SC, it is possible to realize n-bit adders (n is an arbitrary natural number) with a single multiplexer. Let X_1 and X_2 be the input bit strings, S be the selected signal, respectively, and Y be the output bit string. The relationship between the probability of the occurrence of logic 0 and logic 1 at the input and output of the multiplexer is expressed in following equation:

$$p(Y=1) = p(X_1=1) \times p(S=1) + p(X_2=1) \times p(S=0). \tag{4}$$

In particular, when the selected signal whose probability of occurrence of logic 0 and logic 1 is 1/2, we can realize addition with a single multiplexer, such that the output is normalized by a factor of 1/2.

Second, in SC flamework, the operation via OR gate is called imperfect addition. If the input bit strings are X_1 and X_2, and the output bit string is Y, the relationship between the probability of occurrence of logic 0 and logic 1 at the input and output of the OR gate, respectively, is expressed in the following equation:

$$\begin{aligned} p_0 &= p_1 p_2 + p_1(1-p_2) + (1-p_1)p_2 \\ &= p_1 + p_2 - p_1 p_2. \end{aligned} \tag{5}$$

Therefore, when the occurrence of the two inputs p_1 and p_2 are sufficiently small, the OR gate operation can be approximated as an addition.

2.3 Differential representation

In general, probability can only represent positive values. However, in the machine learning field, we are required to handle both positive and negative values. Consequently, we introduce differential representation here. If the values taken by two bit strings at a certain time are X^+ and X^-, the differential representation of the value X to be represented is defined by the following equation:

$$X \equiv X^+ - X^-. \tag{6}$$

3. Previous research

The method presented in Refs. [3],[4] requires a large number of bit registers in the finite-state machine (FSM), which increases the circuit size of the entire network. In addition, the subtraction, which is performed as a preprocessing of the activation function, requires a decoding process with U/D counters. The operation time increases in proportion to the number of layers in the network.

In the method presented in Ref. [5], the summation operation is performed by solely adopting the decoding process with the U/D counter. The multiplication for each node of the neuron is performed sequentially, thereby requiring considerable operation time.

4. Proposed three methods

First, we propose pseudo-activation function. Let the input probabilities of the OR gate be p_1 and p_2 and the output probability be p_0; when the two inputs are equal, the following equation is satisfied:

$$p_0 = p_1 + p_2 - p_1 p_2 = 2p_1 - p_1^2. \tag{7}$$

The above equation demonstrates that nonlinear operations can be realized using OR gates. The circuit that realizes the activation function using this nonlinearity is presented in Fig. 2. Here, the independence of the input of the logic gate is maintained by delaying the bit sequence using the D-FF. From Eq.(7), the input/output of the circuit illustrated in Fig. 2 satisfies the following equation:

$$Y = g(X) = (2X - X^2)^2. \tag{8}$$

We present the input–output relationship when the bit string length is $L = 255$ in Fig. 3. We deduce that the threshold of this activation function is approximately 0.5.

Next, we propose a pseudo-subtraction method that mimics synaptic transmission. There are two types of synapses that transmit information in a neuron: excitatory synapses, which stimulate the firing of postsynaptic cells, and shunting synapses, which suppress firing. A function that mimics the behavior of these two types of synapses is shown in following equation:

$$X = X^+ \times (1 - X^-), \tag{9}$$

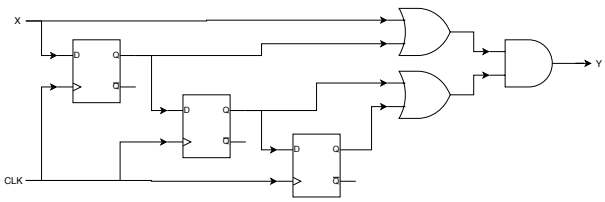


Fig. 2: Activation function with OR gates.

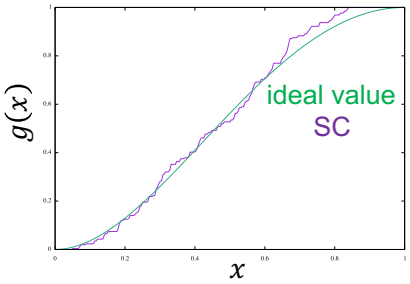


Fig. 3: Input and output of the activation function g(x) using an OR gate.

where X, X^+, and X^- corresponds to the output of a synapse, and excitatory synapse, and a shunting synapse, respectively. All subtractions in the network to perform the subtraction shown in Eq.6 are replaced by this operation.

Third, we propose summation utilized by OR gates. As aforementioned, there are two types of additions in SC. In the previous studies, weighted addition has been mainly adopted; however, when weighted addition is used for the summation operation, the output is normalized to $1/n$, according to the number of inputs n. In the previous research, to input normalized values to activation function, decording for addition is required. In other words, addition significantly increased the circuit area and calculation time. In contrast, imperfect adders have a small circuit area and do not normalize the output. In addition, when the input probabilities are both small, imperfect addition can be approximated to general addition. Accordingly, we decided to introduce an imperfect adder into the MLP based on SC.

5. Implementation of multilayer perceptron (MLP) with SC

5.1 Proposed network and comparison to the previous research

In this study, we propose a three-layer network with SC. The number of neurons in the input layer is N_X, the number of neurons in the hidden layer is N_V, and the number of neurons in the output layer is N_Y. Let neurons in each layer be $X_k(k = 0, 1, ..., N_X - 1)$,$V_j(j = 0, 1, ..., N_V - 1)$, and $Y_i(i = 0, 1, ..., N_Y - 1)$, the weight from the input layer to the hidden layer is $W_{j,k}$, and the learning rate is $W_{i,j}$. The learning method is stochastic gradient descent, and the learning rate is η. We implement the MLP with SC by employing the three methods described above. The forward operation of the three methods is represented by the following equation:

$$h_j^+ = OR\sum_{k=0}^{N_X-1}(w_{j,k}^+ * X_k), h_j^- = OR\sum_{k=0}^{N_X-1}(w_{j,k}^- * X_k), V_j = g(h_j^+ * (1 - h_j^-)),$$
$$h_i^+ = OR\sum_{j=0}^{N_V-1}(W_{i,j}^+ * V_j), h_i^- = OR\sum_{j=0}^{N_V-1}(W_{i,j}^- * V_j), Y_i = g(h_i^+ * (1 - h_i^-)),$$

where $OR\sum$ is defined as the sum operation via imperfect addition. In addition, we introduced a learning function based on the stochastic gradient descent method by performing differentiation on the error function with the proposed method. Hence the block diagram of the MLP with SC is illustrated in Fig. 4. Here, a 9-bit UD counter is adopted as a decoder to match the bit string length ($L = 255$), and an 8-bit register is used to store the weights. In addition, an 8-bit LFSR was used as a random-number generator for encoding. Compared to the previous research, the proposed MLP has the capability of learning. In addition, it has a time advantage in forward calculation. Compared to the previous research in Ref. [4], the proposed MLP performes the sum-of-products operation in parallel (i.e., no decoding is required for the sum operation). In other words, the cost, namely time, decreases in proportion to the total number of neurons in the network. Compared to the previous research in Ref. [5], the introduction of the pseudo-subtraction eliminates the need for decoding, which was required for differential representation.

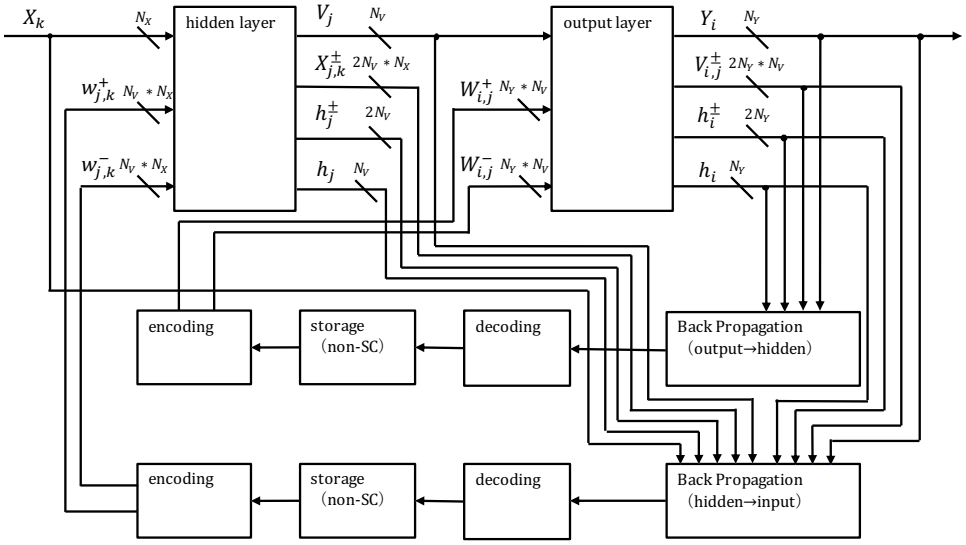


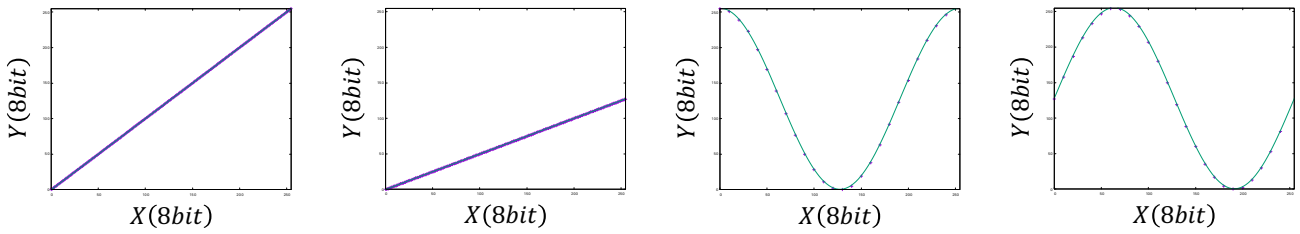
Fig. 4: Block diagram of MLP with SC.

5.2 Assessment of Learning

By simulating the proposed MLP on C++, we trained MLP to perform various logic operations, linear regression problems, and nonlinear regression problems. The parameters used for training and settings of teacher and train data are shown in Table I, and the results are shown in the Fig. 5.

Table I: MLP parameters and settings of teacher and train data.

Type	Linear regressions		Nonlinear regressions	
Teach label	$y = x$	$y = \frac{1}{2}x$	$y = cos(x)$	$y = sin(x)$
Number of training data	100		256	
Number of test data	256		25	
Network size	9-32-8		9-128-8	
Learning rate	0.3			
Stream length	255			



(a) y=x **(b)** y=1/2x **(c)** y=cos(x) **(d)** y=sin(x)

Fig. 5: Results of training for the proposed MLP based on C++ simulation.

5.3 Power evaluation

Next, we evaluate the power consumption of the MLP proposed in this thesis. In the evaluation, we assumed minimnist as the training set. Therefore, the number of neurons in each layer was set to $N_X = 197$, $N_V = 64$, $N_Y = 10$. Table II illustrates the estimated power consumption of the proposed MLP by using the Synopsys design compiler. From Table II, when we use proposed MLP with training, power consumption is approximately 7.5 times higher than that of the case without training. The significant increase in power consumption is primarily owing to the large number of multipliers required for training, and more efficient learning circuits are required to reduce the power consumption. Next, for comparison to the previous research depicted in Ref. [5], which has a learning function, the scaled power values with scaling laws applied are presented in Table II. However,

because the supply voltage was not mentioned in Ref. [5], it was assumed to be an estimated value. The number of neurons in the proposed MLP framework is larger than those used in the previous research. Moreover, owing to the implementation of multi-layer back propagation, the computation for learning is more complicated; nevertheless, the power consumption remains approximately 2.5 times of value mentioned in Ref. [5]. In addition, the computation time has been reduced in the proposed MLP framework, which has increased proportionally to the number of neurons owing to sequential multiplication in the previous research.

Table II: Power evaluation of the proposed MLP.

	This work		Ref. [5]	This work (scaled)
Evaluation method	Synopsys design compiler		actural measured	scaled
Technology	UMC 0.18 μm 1P6M CMOS		0.6 μm	
Function	Learning and Inference	Inference	Learning and Inference	
Network Configuration	197 - 64 - 10		50 neurons	197 - 64 - 10
Supply Voltage	1.8 V		5 V (estimated)	
Frequency	100 MHz		30 MHz	
Dynamic Power	57.4 mW	4.45 mW	———	———
Static Power	32.9 mW	7.66 mW	———	———
Total Power	90.3 mW	12.1 mW	330 mW	836 mW

6. Conclusion

In this study, as a method for introducing SC to MLP, we first proposed an area-efficient activation function, focusing on the nonlinearity of OR gates. Next, we proposed a subtractor that mimics the synaptic transmission of neural circuits without requiring decoding in the middle of the operation; in addition we proposed a summation operation that utilizes imperfect addition by assuming threshold processing using the activation function. Using the proposed method, we implemented a digital MLP based on SC. The implemented MLPs were adopted to learn and infer linear and nonlinear regression problems. We demonstrated the feasibility of using SC-based edge AI integrated circuits. In the future, it will be crucial to search for more power-efficient learning circuits and appropriate parameters for learning more complex datasets. In addition, SCs will generally require low-power and area-efficient encoders, decoders, and memory elements.

7. Acknowledgment

This study was supported in part by JSPS KAKENHI (Grant No. 18H05288), Japan.

References

[1] Yoshiaki Sasaki, Kohei Nishida, and Tetsuya Asai, “Learning device, subtraction circuit, and activation function circuit,” Japanese patent application No.2021-118326.

[2] B.R. Gaines, “Stochastic computing systems,” Adv. Informat. Sys. Sci., pp.37-172, Springer, 1969.

[3] B.D. Brown and H.C. Card, “Stochastic neural computation I. Computational elements,” IEEE Trans. Comput., vol.50, no.9, pp.891-905, Sept. 2001, DOI: 10.1109/12.954505.

[4] Shigeo Sato, Ken Nemoto, Shunsuke Akimoto, Mitsunaga Kinjo, and Koji Nakajima, “Implementation of a new neurochip using stochastic logic,” IEEE Trans. Neural Netw., vol.14, no.5, Sept. 2003, DOI:10.1109/TNN.2003.816341.

[5] Naoya Onizawa, Kazumichi Matsumiya, and Takahiro Hanyu, “Energy-efficient brainware LSI based on stochastic computation,” IEICE Fundamentals Review, vol.11, no.1, pp.28-39, July. 2017, DOI:10.1587/essfr.11.1_28.