Low Cost and Fault Tolerant Parallel Computing Using Stochastic Two-Dimensional Finite State Machine

Xuechun WANG[†], Yuan JI^{†,††a)}, Wendong CHEN[†], Feng RAN^{†,††}, *Nonmembers,*
and Aiyang GUO^{†,††}, *Student Member*

SUMMARY Hardware implementation of neural networks usually have high computational complexity that increase exponentially with the size of a circuit, leading to more uncertain and unreliable circuit performance. This letter presents a novel Radial Basis Function (RBF) neural network based on parallel fault tolerant stochastic computing, in which number is converted from deterministic domain to probabilistic domain. The Gaussian RBF for middle layer neuron is implemented using stochastic structure that reduce the hardware resources significantly. Our experimental results from two pattern recognition tests (the Thomas gestures and the MIT faces) show that the stochastic design is capable to maintain equivalent performance when the stream length set to 10 Kbits. The stochastic hidden neuron uses only 1.2% hardware resource compared with the CORDIC algorithm. Furthermore, the proposed algorithm is very flexible in design tradeoff between computing accuracy, power consumption and chip area.

key words: stochastic computing, fault tolerant, pattern recognition, power consumption

1. Introduction

Parallel computing is receiving increasing attention due to its efficient data processing ability. Computation on stochastic bit stream has become an attractive solution in parallel computing environment for many years [1], [2]. It represents a value m in the unit interval (i.e., $0 \leq x \leq 1$) by a bit stream L , in which the probability of a one is m [3]. Conventional methods generally use multipliers and adders in solving some matrix operation problems. With the increase in matrix size, the number of multipliers and adders is multiplied, and ultimately beyond hardware capabilities [4]. When the probabilistic values are used, the complicated operations can be approximated through simple arithmetic units, (e.g. absolute subtraction can be implemented with only a single XNOR or XOR gate.) resulting in simplified circuit structure and minimized system design cost. Meanwhile, most data processing systems that boast high accuracy work well only under ideal conditions [5]. When the environmental noise or manufacturing parameter deviation corrupts the system, traditional method often add redundancy design to raise anti-jamming capability. Thereby increasing the scale and cost of the system.

An effective solution for reducing area and noise-sensitivity is to move from deterministic values toward probabilistic values. Some stochastic data processing designs have been already introduced [6]–[8]. For example, Brown and Card [9], [10] showed that for complex operations, such as the exponentiation and sine functions, stochastic computing consumes less energy than binary radix computing, and the latency problem can be resolved using parallel processing method or a higher operating frequency owing to its simple circuit structure. Also in [11] Qian et al described a stochastic image processing architecture that is fault tolerant. More recently, some researches give structures of stochastic logic using Back Propagation (BP) neural network that uses much adders to achieve sigmoid function. However, the result is unpractical because scaled addition introduced precision loss. Moreover, we found that the generalization capability of RBF neural network is superior to that of BP neural network [12].

As discussed above, we extend previous work of this algorithm. We encounter the scaled addition problem and best use the advantages of Two-Dimensional (2D) Finite State Machine (FSM) through the case study of RBF neural network. The overall architecture is design from the samples of input and output of the pattern by stochastic logic. The results have shown the effectiveness and capability of presented computing for high error-tolerant, high recognition accuracy and more power-efficient.

2. Stochastic Computing

Stochastic computing uses the probability of “1” (or “0”) appearing in a stochastic bit stream to represent a deterministic value [1], [2]. It has two coding formats: unipolar and bipolar formats. Both formats can coexist in a single system together. Operations at the logic level can be easily parallelized in space to reduce energy and total delay. That is to say, instead of using a bit stream of length L to represent a real value m , we increase the number of logic gates by N . Due to the short bit stream of length L/N , computation time can be much higher reduced. This can be done with little overhead due to the typically low gate area. Figure 1 shows how we can represent a value 0.5 by 2 stochastic bit streams of length 4.

Such streams can be generated with pseudo-random constructs such as Linear Feedback Shift Register (LFSR). As shown in Fig. 2, we use a LFSR and a comparator [9]

Manuscript received January 6, 2017.

Manuscript revised April 29, 2017.

Manuscript publicized July 18, 2017.

[†]The authors are with Microelectronic Research and Development Center, Shanghai University, Shanghai, 200072, China.

^{††}The authors are also with School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, 200072, China.

a) E-mail: jiyuan@shu.edu.cn

DOI: 10.1587/transinf.2017PAL0003

Fig. 1 Representations of stochastic bit streams.

Fig. 2 Binary radix encoding to stochastic bit stream converter.

Fig. 3 2D finite state machine state transition diagram.

to explain the operation of the proposed stochastic computing. The LFSR generates a different random number in each clock cycle. Assume that we convert a value m from its binary radix encoding to L -bits stochastic encoding. The comparator produces a one if the random number is less than the input number, and it produces a zero for the others.

After converting the real number to random stream, the mathematic operations, like addition, subtraction, division can be implemented using simple digital logics. Thus, this approach enables simple hardware and massively parallel processing. However, 2^m bits is needed for a stochastic encoding transform of m bits binary encoding which requires too much clock cycles to finish. This motivates us to design the complex function through a finite-state machine in terms of faster clock frequency.

Here we provide a two-dimensional state array [11] shown in Fig. 3, which has two inputs (the input stream and the modulation stream) and total of $M \times N$ states (S_0 to S_{MN-1}). The numbers on the arrows represent conditions that must be satisfied to proceed along the transition. Based on the Markov forecast method, when the state transit fairly enough, the state transition configuration is considered as probability distribution which is independent of initial states. Consequently, based on the probability of the stochastic stream, some complex functions simulation with parallel processing would be possible.

Fig. 4 Architecture of the proposed stochastic RBF neural network.

3. Stochastic Logic Applied to RBF Neural Network

The proposed case study of RBF neural network is a feed forward network embedded in three layers: input, middle and output layers [12]. The input layer distribute the variable $\mathbf{x} = [x_1, x_2, \dots, x_i]^T$ to all neurons in middle layer where the vector is processed and further transmitted to the output layer to give a linear combination of output weight. For RBF responses in the hidden layer, an RBF network utilizes several kernel functions. In this work, the most commonly used Gaussian function is chosen and we expand and rewrite it as follows:

$$y_j = \exp(-|x_1 - c_{1j}|^2 / \sigma^2) \times \exp(-|x_2 - c_{2j}|^2 / \sigma^2) \times \dots \times \exp(-|x_i - c_{ij}|^2 / \sigma^2) \quad (1)$$

where σ is a positive real constant or kernel radius which measures the smoothness of fitting function and $|\mathbf{x} - \mathbf{c}_{ij}|$ denotes the Euclidean distance. The centroid c_{ij} and constants σ are determined accordingly to the training data process. The middle layer operation can be seen clearly by referring to the right-side of Fig. 4, which presents a detailed architecture of stochastic RBF neural network. The input value x and the cluster center value c_{ij} are subtracted through XOR gates. The Gaussian function is synthesized based on 2D-FSM topology using the modulation streams k and q . They are connected by shift registers. The middle layer output results are obtained by ANDING all FSM outputs as described above.

On the left-side of Fig. 4 shows the stochastic units applies to output layer of neural network. We use a scaling deterministic-to-stochastic converter to scale down the value of ω within the range of $[-1, 1]$. The output of middle neuron y and the weight of the output neuron ω are multiplied through XOR gates. Note that addition operation will lead to declined accuracy, so we transfer the multiplication results into deterministic parameters using the sum of binary adder, and retrieve the real data z_k .

Correlations among the stochastic numbers often lead to inaccuracies which implies the need for many random sources. There has been some effort by using techniques for generating multiple uncorrelated pseudorandom sources [6], [17]. However, they still dominate the area cost of the architecture. The solution described in this paper is to make full use of the LFSR. As we know, the middle layer inputs must

be correlated, which is assured by assigning them a same random source. For the modulation streams k and q , we use the same random source and obtain shift from the main sequence by tap shift registers. All those registers are integrated into control system. As for the output center vector ω , the computation is processed in the binary adder which requires only one LFSR. The area cost of these resources is minor since they are small, which gives an acceptably small overhead for neural network chip.

4. Experimental Results and Discussions

4.1 Gesture Recognition

The Thomas Data set contains 40 samples of 25 species hand gesture, and the pixel size of gesture is 248×256 . There are a large number of pixel points in digital image. The input will be somewhat redundant because the values of adjacent pixels in an image are highly correlated. Meanwhile, the recognition process will be inefficient if the image training is given to each pixel point. Therefore, we adopt principal component analysis first to effectively approximate the input with a much lower dimensional one, while incurring very little error.

We randomly divided the data set into training samples and testing samples. Each sample have 25 gesture images (20 images for each gesture). The RBF neural network is set to 39 input neurons, 15 hidden neurons and 25 output neurons. We take the MATLAB language as the experimental procedure, and train the network through orthogonal least squares algorithm. The proposed and deterministic logic are simulated in the same environment. In deterministic logic, the recognition rate reached 96.80% and the Mean Square Error (MSE) is 0.0159.

There are several interesting results in Fig. 5. The most important point is that the stochastic computing bit length has a considerable impact on network performance, both for the recognition rate and MSE. When the stream length is 100 bits, the recognition rate reaches 60.33% and the MSE is 0.0609. With the stream length reaches to 5 Kbits, the results change significantly with 58% improvements for the recognition rate and 72% for the MSE. Besides, when the stream length is greater than 50 Kbits, the recognition rate and the MSE almost converge to the rate of the deterministic network.

Stochastic logics can be used to implement the neuron network with simple circuits, so it can be integrated as an independent hardware component on silicon microdisplay. The performance can achieve almost perfect results even if the circuit components are noisy and uncertain [15], [16]. In general, the input data is more easily to corrupt with noise than the internal structure of chips. Therefore, their input referred noise could be very high, which in our experimental architecture can simulate the noise of the input interface. Here we inject different level noises into the stochastic bit stream in the input layer. The stream length is set to 10 Kbits. As shown in Fig. 6, the recognition rate reaches

Fig. 5 Gesture recognition rate and MSE, stochastic computing RBF neural network VS deterministic network (DN).

Fig. 6 The error injection test result for the stochastic computing RBF neural network in the gesture recognition experiment.

96.10% and the MSE is 0.0165 when the noise is set to 5%. When we increase the rate of noise to 15%, the recognition rate is still as high as 94.20%. MSE increases correspondingly when the noise level increases. Thus, the architecture of stochastic RBF neural network has good error immunity when the uniformly distributed noise is less than 15%. The precision of the results is dependent most on the statistics of the bit-streams, and so the computation can tolerate errors gracefully.

4.2 Face Recognition

In this experiment, similarly we separate the MIT faces database into two parts: face library and non-face library. A total of 100 pictures are randomly chosen for each part, all having a resolution of 20×20 pixel.

A stochastic RBF neural network with 400 input neurons, 20 hidden neurons and 2 output neurons is simulated to do the facial recognition. The output value '01' and '10' represent the face and non-face results, respectively. In deterministic logic, the recognition rate reaches 95.0% and the MSE reaches 0.0722. For the stochastic logic, the 2D finite state machine with eight states is used to fit the Gaussian function. The deviation of the output between stochastic network and deterministic network calculated for the network evaluation is characterized by MSE. The stochastic network runs with different stream length and each result is repeated for one thousand times to reduce the random variance.

As shown in Table 1, when the stream length increases from 1 Kbits to 10 Kbits, the results do not change significantly with 4–5% improvements for the recognition rate and 3–4% for MSE. When the stream length is greater than

Table 1 Facial recognition rate and MSE for stochastic computing RBF neural network and deterministic network.

Stochastic computing RBF neural network								DN
Bit length	100	500	1K	5K	10K	50K	100K	/
Recognition rate (%)	72.25	85.90	89.50	93.22	93.74	94.80	94.86	95.00
MSE	0.3717	0.1336	0.1027	0.0779	0.0753	0.0731	0.0725	0.0722

Table 2 The error injection test result for the stochastic computing RBF neural network in the facial recognition experiment.

Stochastic computing RBF neural network							
Error rate	0	5	10	15	20	25	30
Recognition rate (%)	93.74	93.60	93.20	92.90	91.30	88.90	83.40
MSE	0.0753	0.0769	0.0820	0.0914	0.1039	0.1233	0.1493

50 Kbits, the results gradually converge to deterministic network. The main issue of stochastic computing technique is its long time-consuming. General solution is using a faster clock frequency and/or using parallel computing in pattern process.

Table 2 illustrates the noise tolerant of the stochastic RBF neural network in facial recognition. The error given by the stochastic bit stream of input layer is up to 30%, and the stream length is set to 10 Kbits. Clearly, the stochastic RBF neural network shows satisfied robustness of the input noise. Compared with the gesture recognition test, a much shorter stream length is needed to obtain the same recognition rate level, mainly because there are more inputs and fewer outputs in the facial recognition test.

5. Hardware Comparison and Application Forecast

To validate the hardware cost of the proposed algorithm, we implement the stochastic logic RBF network based on Altera Cyclone III FPGA (EP3C80F780C8) with 81264 logic elements and 430 pins. The network parameters c, k, q, ω are stored in the nonvolatile memory and serial shifted to the stochastic network. The data width is equivalent to the bit length of the stochastic logic. Three different deterministic algorithms with the same network structure are chosen and the device utilization is given in Table 3 [3]. The proposed approach uses only 22 logic elements, which is far less than the other methods. A Look-Up Table (LUT) is an array that replaces runtime computation with a simpler array indexing operation. It takes only one clock cycle to complete the operation, but consume a large amount of cache and area. Pre-computation with interpolation can shrink the LUT size but may sacrifice the computation time. Coordinated Rotation Digital Computer (CORDIC) only has simple shift-and-add operations, which is suitable for the implementation on hardware of complex function operation. We also propose the binary design by three cost-performance normalizations (M0, M1 and M2) in each major system component. The higher normalized score means better performance of the whole system. In general, the 2D-FSM is the best choice in terms of the circuit power which is determined by area and speed. As compared with the LUT with interpolation

Table 3 Comparison between different architecture of middle neuron. (A: circuit area, C: clock cycles per operation, O: number of operations per second).

Hardware structure	Data width	A	C	O	M0= 1/AC	M1= 1/A ² C	M2= O/A
2D-FSM	10bits	22	1024	244	1	1	1
	12bits	22	4096	61	0.20	0.25	0.23
	14bits	22	16384	15	<0.01	0.06	0.04
LUT	12bits	33898	1	10000	0.64	<0.01	<0.01
LUT with interpolation	12bits	1853	15	3333	0.80	0.01	0.11
CORDIC	12bits	1079	22	5455	0.95	0.02	0.28

architecture, the stochastic-based 2D-FSM requires 98.81% less area usage and has higher area efficiency. The vector M1, emphasizing the metrics of circuit area, implies that 2D-FSM architecture performs better than the typical implementation when 12 bits of data width assigned. The area efficiency of 12 bits 2D-FSM is found to be competitive with the CORDIC algorithm by the vector M2. It is concluded that our stochastic architecture occupy less area than traditional hardware implementations.

As matter of fact, current researches mainly focus on the theory framework and software realization of deep learning neural network [13], [14], while the method for hardware implementation is still lacking. Furthermore, under the constraint of the traditional computing methods. The neuron occupied undesirable resource consumption, which leads to only tens to hundreds of neurons in one single-chip. Therefore, according to the analyses and discussions above, we confirm that stochastic computing is an acceptable way to solve hardware constraint of deep learning and a large number of applications with high hardware resources cost can benefit from this parallel algorithm.

6. Conclusion

In this paper, the pattern recognition experiment of the stochastic computing is simulated and analyzed as a good beginning example of depth study especially for the design of realizable, simple, inexpensive and robust system with brilliant prospects in applications. Under the same network structure, by changing the length of the stream we can form different computation precision without changing the hard-

ware structure of network, which gives the system designers more flexibility to weigh the pattern accuracy, power consumption and the speed of calculation. Moreover, noise input result shows that the stochastic logic improves the ability of fault tolerant remarkably for the given error rates. Device fault will not affect the performance and accuracy of the entire circuit. This will open up more opportunities for the new type of micro-mechanical manufacturing application and promotion.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant No. 61674100 and 61376028) and by the Shanghai Science and Technology Committee (Grant No. 16JC1400602). The authors also would like to thank the chair and the committee members of the information and systems society of IEICE, for providing this opportunity to write this survey paper.

References

- [1] B.R. Gaines, "Stochastic computing systems," *Advances in Information Systems Science*, ed. J. Tou, pp.37–172, Springer US, 1969.
- [2] J.P. Hayes, "Introduction to stochastic computing and its challenges," *Proc. 52nd Annual Design Automation Conference*, San Francisco, Calif, USA, pp.1–3, June 2015.
- [3] Y. Ji, F. Ran, C. Ma, and D.J. Lilja, "A hardware implementation of a radial basis function neural network using stochastic logic," *Proc. 2015 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pp.880–883, March 2015.
- [4] S.L. Bade and B.L. Hutchings, "FPGA-based stochastic neural networks-implementation," *Proc. IEEE Workshop on FPGA's for Custom Computing Machines*, Napa, CA, pp.189–198, April 1994.
- [5] M. Alawad and M. Lin, "FIR filter based on stochastic computing with reconfigurable digital fabric," *2015 IEEE 23rd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Vancouver, BC, Canada, pp.92–95, May 2015.
- [6] A. Alaghi and J.P. Hayes, "Survey of stochastic computing," *ACM Trans. Embed. Comput. Syst.*, vol.12, no.2s, pp.1–19, May 2013.
- [7] J. Han, H. Chen, J. Liang, P. Zhu, Z. Yang, and F. Lombardi, "A stochastic computational approach for accurate and efficient reliability evaluation," *IEEE Trans. Comput.*, vol.63, no.6, pp.1336–1350, Nov. 2014.
- [8] H. Li, D. Zhang, and S.Y. Foo, "A stochastic digital implementation of a neural network controller for small wind turbine systems," *IEEE Trans. Power Electron.*, vol.21, no.5, pp.1502–1507, Sept. 2006.
- [9] 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.
- [10] B.D. Brown and H.C. Card, "Stochastic neural computation. II. Soft competitive learning," *IEEE Trans. Comput.*, vol.50, no.9, pp.906–920, Sept. 2001.
- [11] W. Qian, X. Li, M.D. Riedel, K. Bazargan, and D.J. Lilja, "An architecture for fault-tolerant computation with stochastic logic," *IEEE Trans. Comput.*, vol.60, no.1, pp.93–105, Oct. 2010.
- [12] H.K. Lee, "Intelligent image retrieval using neural network," *IEICE Trans. Inf. & Syst.*, vol.E84-D, no.12, pp.1810–1819, Jan. 2001.
- [13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol.521, no.7553, pp.436–444, May 2015.
- [14] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology," *Molecular Systems Biology*, vol.12, no. 7, pp.878, July 2016.
- [15] W. Qian, M.D. Riedel, H. Zhou, and J. Bruck, "Transforming probabilities with combinational logic," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol.30, no.9, pp.1279–1292, Sept. 2011.
- [16] B. Moons and M. Verhelst, "Energy-efficiency and accuracy of stochastic computing circuits in emerging technologies," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol.4, no.4, pp.475–486, Oct. 2014.
- [17] A. Alaghi, C. Li, and J.P. Hayes, "Stochastic circuits for real-time image-processing applications," *Design Automation Conference*, Austin, TX, USA, pp.1–6, June 2013.