A Simple Design and Implementation of Reconfigurable Neural Networks

Hazem M. El-Bakry, and Nikos Mastorakis

***Abstract*—There are some problems in hardware implementation of digital combinational circuits. In contrast, analog design has the advantages of both economy and easy to implement compared with the digital design. In this paper, a simple design and implementation of analog reconfigurable artificial neural network is presented. A novel design of arithmetic unit that including full-adder, full-subtractor, 2-bit digital multiplier and 2-bit digital divider is introduced. The proposed neural network has been realized by hardware components and the results are simulated using H-spice program. Practical results confirm the theoretical considerations.**

I. INTRODUCTION

Advances in MOS VLSI have made it possible to integrate neural networks of large sizes on a single-chip [1,2]. Hardware realizations make it possible to execute the forward pass operation of neural networks at high speeds, thus making neural networks possible candidates for real-time applications. Other advantages of hardware realizations as compared to software implementations are the lower per unit cost and small size system.

Analog circuit techniques provide area-efficient implementations of the functions required in a neural network, namely, multiplication, summation, and the sigmoid transfer characteristic [3]. In this paper, we describe the design of a reconfigurable neural network in analog hardware and demonstrate experimentally how a reconfigurable artificial neural network approach is used in implementation of arithmetic unit that including full-adder, full-subtractor, 2-bit digital multiplier, and 2-bit digital divider.

One of the main reasons for using analog electronics to realize network hardware is that simple analog circuits (for example adders, sigmoid, and multipliers) can realize several of the operations in neural networks. Nowadays, there is a growing demand for large as well as fast neural processors to provide solutions for difficult problems. Designers may use either analog or digital technologies to implement neural network models. The analog approach boasts compactness and high speed. On the other hand, digital implementations offer flexibility and adaptability, but only at the expense of speed and silicon area consumption.

Manuscript received December 14, 2008.

H. M. El-Bakry is with the Faculty of Computer Science & Information Systems, Mansoura University, EGYPT (phone: +2-050-2317356; fax: +2-050-2221442; e-mail: helbakry20@yahoo.com).

N. Mastorakis is with Department of Computer Science, Military Institutions of University Education (MIUE) -Hellenic Naval Academy, Greece.

II. ANALOG IMPLEMENTATION OF RECONFIGURABLE NEURAL NETWORKS

A) Implementation of artificial neuron

Implementation of analog neural networks means that using only analog computation [4,6,8]. Artificial neural network as the name indicates, is the interconnection of artificial neurons that tend to simulate the nervous system of human brain [5]. Neural networks are modeled as simple processors (neurons) that are connected together via weights. The weights can be positive (excitatory) or negative (inhibitory). Such weights can be realized by resistors as shown in Fig. 1.

The computed weights may have positive or negative values. The corresponding resistors that represent these weights can be determined as follow [6]:

$$w_{in} = -R_f / R_{in} \qquad i = 1, 2, \ldots\ldots, n \quad (1)$$

$$Wpp = \frac{\left(1 + \sum_{i1}^{n} Win\right)\frac{Ro}{Rpp}}{\left(1 + \frac{Ro}{R1p} + \frac{Ro}{R2p} + \ldots\ldots\ldots\ldots\ldots\ldots + \frac{Ro}{Rpp}\right)} \quad (2)$$

The exact values of these resistors can be calculated as presented in [4,8]. The summing circuit accumulates all the input-weighted signals and then passes to the output through the transfer function [3]. The main problem with the electronic neural networks is the realization of resistors which are fixed and have many problems in hardware implementation [7]. Such resistors are not easily adjustable or controllable. As a consequence, they can be used neither for learning, nor can they be used for recall when another task needs to be solved. So the calculated resistors corresponding to the obtainable weights can be implemented by using CMOS transistors operating in continuous mode (triode region) as shown in Fig. 2. The equivalent resistance between terminal 1 and 2 is given by [9]:

$$R_{eq} = 1/[K(V_g - 2V_{th})] \quad (3)$$

B) Reconfigurability

The interconnection of synapses and neurons determines the topology of a neural network. Reconfigurability is defined as the ability to alter the topology of the neural network [9]. Using switches in the interconnections between synapses and neurons permits one to change the network topology as shown in Fig.3. These switches are called "reconfiguration switches".

The concept of reconfigurability should not be confused with *weight programmability*. Weight programmability is defined as the ability to alter the values of the weights in each synapse. In Fig.3, weight programmability involves setting the values of the weights w_1, w_2, w_3, ... , w_n. Although reconfigurability can be achieved by setting weights of some synapses to zero value, this would be very inefficient in hardware.

C) The need for reconfigurable systems

Reconfigurability is desirable for several reasons [1]:

1. Providing a general problem-solving environment.
2. Correcting offsets.
3. Ease of testing.
4. Reconfiguration for isolating defects.

III. DESING OF ARITHMETIC UNIT USING RECONFIGURABLE ANNS

In previous paper [12], a neural design for logic functions by using modular neural networks was presented. Here, a simple design for the arithmetic unit using reconfigurable neural networks is presented. The aim is to have a complete design for ALU by using the benefits of both modular and reconfigurable neural networks.

A) Full-Adder/Full-Subtractor Implementation Using ANN

Full-adder/full-subtractor problem is solved practically and a neural network is simulated and implemented using the back-propagation algorithm for the purpose of learning this network [10]. The network is learned to map the functions of full-adder and full-subtractor. The problem is to classify the patterns shown in Table 1 correctly.

The computed values of weights and their corresponding values of resistors are described in Table 2. After completing the design of the network, simulations are carried out to test both the design and performance of this network by using H-spice. Experimental results confirm the proposed theoretical considerations.

Fig. 4 shows the construction of full-adder/full-subtractor neural network. The network consists of three neurons and 12-connection weights.

B) 2-Bit Digital Multiplier Implementation

2-bit digital multiplier can be realized easily using the traditional feed-forward artificial neural network [11]. As shown in Fig. 5, the implementation of 2-bit digital multiplier using the traditional architecture of a feed-forward artificial neural network requires 4-neurons, 20-synaptic weights in the input-hidden layer, and 4-neurons, 20-synaptic weights in the hidden-output layer. Hence, the total number of neurons is 8-neurons with 40-synaptic weights.

In the present work, a new design of 2-bit digital multiplier has been adopted. The new design requires only 5-neurons with 20-synaptic weights as shown in Fig. 6. The network receives two digital words, each word has 2-bit, and the output of the network gives the resulting multiplication. The network is trained by the training set shown in Table 3. During the training phase, these input/output pairs are fed to the network and in each iteration; the weights are modified until reached to the optimal values. The optimal value of the weights and their corresponding resistance values are shown in Table 4. The proposed circuit has been realized by hardware means and the results have been tested using H-spice computer program. Both the actual and computer results are found to be very close to the correct results.

C) 2-Bit Digital Divider Implementation

2-bit digital divider can be realized easily using the artificial neural network. As shown in Fig. 7, the implementation of 2-bit digital divider using neural network requires 4-neurons, 20-synaptic weights in the input-hidden layer, and 4-neurons, 15-synaptic weights in the hidden-output layer. Hence, the total number of neurons is 8-neurons with 35-synaptic weights. The network receives two digital words, each word has 2-bit, and the output of the network gives two digital words one for the resulting division and the other for the resulting remainder. The network is trained by the training set shown in Table 5.

The values of the weights and their corresponding resistance values are shown in Table 6. The results have been tested using H-spice computer program. Computer results are found to be very close to the correct results.

Arithmetic operations namely, addition, subtraction, multiplication, and division can be realized easily using a reconfigurable artificial neural network. The proposed network consists of only 8-neurons, 67-connection weights, and 32-reconfiguration switches. Fig. 8 shows the block diagram of the arithmetic operation using reconfigurable neural network. The network includes full-adder, full-subtractor, 2-bit digital multiplier, and 2-bit digital divider. The proposed circuit is realized by hardware means and the results are tested using H-spice computer program. Both the actual and computer results are found to be very close to the correct results.

The computed values of weights and their corresponding values of resistors are described in Tables 2,4,6. After completing the design of the network, simulations are carried out to test both the design and performance of this network by using H-

spice. Experimental results confirm the proposed theoretical considerations as shown in Tables 7,8.

VI. Conclusion

A new concept for realizing arithmetic unit that includes full-adder, full-subtractor, 2-bit digital multiplier, and 2-bit digital divider by using analog reconfigurable artificial neural networks has been presented. The proposed full-network has been realized by hardware means and the results have been tested using H-spice computer program. Both the actual and computer results are found to be very close to the correct results.

References

[1] Srinagesh Satyanarayna, Yannis P. Tsividis, and Hans Peter graf, "A Reconfigurable VLSI Neural Network," IEEE Journal of Solid State Circuits, vol. 27, no. 1, January 1992.

[2] E. R. Vittos, "Analog VLSI Implementation of Neural Networks," in proc. Int. Symp. Circuits Syst. (new Orleans, LA), 1990, pp. 2524-2527.

[3] H. P. graf and L. D. Jackel, "Analog Electronic Neural Network Circuits," IEEE Circuits Devices Mag., vol. 5, pp. 44-49, July 1989.

[4] H. M. EL-Bakry, M. A. Abo-Elsoud, and H. H. Soliman and H. A. El-Mikati " Design and Implementation of 2-bit Logic functions Using Artificial Neural Networks ," Proc. of the 6th International Conference on Microelectronics (ICM'96), Cairo, Egypt, 16-18 Dec. , 1996.

[5] Simon Haykin, "Neural Network : A comprehensive foundation", Macmillan college publishing company, 1994.

[6] Jack M. Zurada, "Introduction to Artificial Neural Systems," West Publishing Company, 1992.

[7] C. Mead, and M. Ismail, "Analog VLSI Implementation of Neural Systems," Kluwer Academic Publishers, USA, 1989

[8] H. M. EL-Bakry, M. A. Abo-Elsoud, and H. H. Soliman and H. A. El-Mikati " Implementation of 2-bit Logic functions Using Artificial Neural Networks ," Proc. of the 6th International Conference on Computer Theory and Applications, Alex., Egypt, 3-5 Sept. , 1996, pp. 283-288.

[9] I. S. Han and S. B. Park, "Voltage-Controlled Linear Resistor by Using two MOS Transistors and its Applications to RC Active Filter MOS Integration," Proceedings of the IEEE, Vol.72, No.11, Nov. 1984, pp. 1655-1657.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagation Error," Nature, vol.323, pp.533-536,1986.

[11] Laurene Fausett, "Fundamentals of Neural Network : Architectures, Algorithms, and Applications," Prentice Hall International.

[12] H. M. El-bakry, "Complexity Reduction Using Modular Neural Networks," Proc. of IEEE IJCNN'03, Portland, Oregon, pp. 2202-2207, July, 20-24, 2003.

Table I
Truth table of full-adder/full-subtractor

I/P			*Full-Adder*		*Full-Subtractor*	
x	*y*	*z*	*S*	*C*	*D*	*B*
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	1	1	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	1	0	0	0
1	1	0	1	0	0	0
1	1	1	1	1	1	1

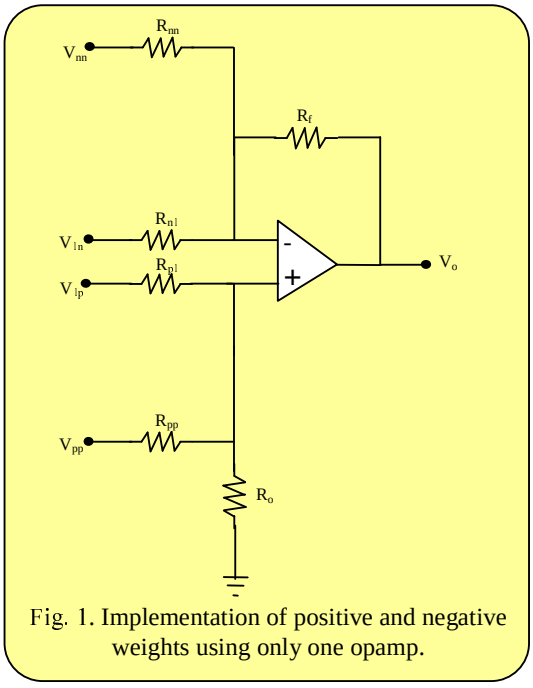

Fig. 1. Implementation of positive and negative weights using only one opamp.

Table II
Computed weights and their corresponding resistances of full-adder/full-subtractor

I / P	Neuron (1)		Neuron (2)		Neuron (3)	
	Weight	Resistance	Weight	Resistance	Weight	Resistance
1	7.5	11.8 R_o	15	6.06 R_o	15	6.06 R_o
2	7.5	11.8 R_o	15	6.06 R_o	15	6.06 R_o
3	7.5	11.8 R_o	-10	0.1 R_f	-10	0.1 R_f
Bias	-10.0	0.1 R_f	-10	0.1 R_f	-10	0.1 R_f

Table III
2-bit digital multiplier training set

I/P				O/P			
B_2	B_1	A_2	A_1	O_4	O_3	O_2	O_1
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Table IV
Weight values and their corresponding resistance values

Neuron	I/P	W. Value	Resistor
(1)	A_1	7.5	1200
	B_1	7.5	1200
	Bias	-10.0	100
(2)	A_1	7.5	1450
	B_2	7.5	1450
	Bias	-10.0	100
	N_4	-30.0	33
	N_5	20.0	618
(3)	A2	7.5	1200
	B2	7.5	1200
	bias	-10.0	100
	N4	-10.0	100
(4)	A1	3.0	1200
	A2	3.0	1200
	B1	3.0	1200
	B2	3.0	1200
	bias	-10.0	100
(5)	A2	7.5	1200
	B1	7.5	1200
	Bias	-10.0	100

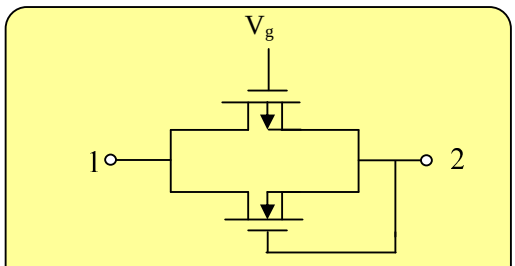

Fig.2. Two MOS transistor as a linear resistor.

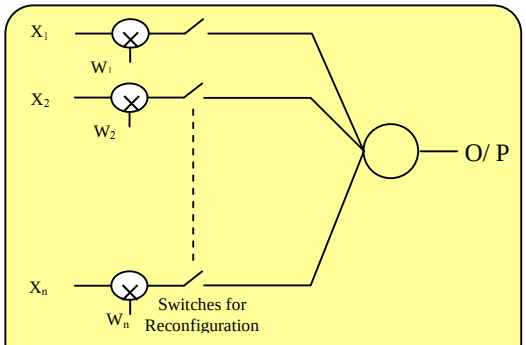

Fig.3. Neuron with reconfigurable switches.

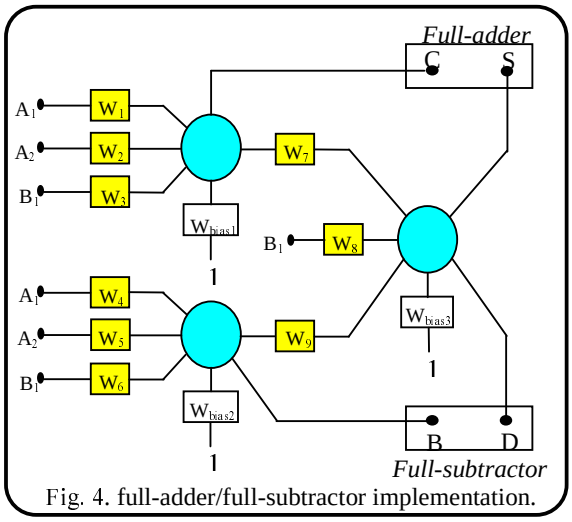

Fig. 4. full-adder/full-subtractor implementation.

Table V
2-bit digital multiplier training set

I/P				O/P			
B_2	B_1	A_2	A_1	O_4	O_3	O_2	O_1
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	1	1	1	1
0	1	0	1	0	0	0	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	1	1	1
1	0	0	1	0	0	1	0
1	0	1	0	0	0	0	1
1	0	1	1	1	0	0	0
1	1	0	0	1	1	1	1
1	1	0	1	0	0	1	1
1	1	1	0	0	1	0	1
1	1	1	1	0	0	0	1

Table VI
Weight values and their corresponding resistance values

Neuron	I/P	W. Val.	Resistor
(1)	A_1	-17.5	56
	A_2	-17.5	56
	B_1	5	2700
	B_2	5	2700
	Bias	5	2700
(2)	A_1	7.5	1200
	A_2	7.5	1200
	B_1	-10	100
	B_2	7.5	1200
	Bias	-17.5	56
(3)	A_1	7.5	1200
	A_2	-10	100
	B_2	7.5	1200
	Bias	-10	100
(4)	A_1	-4.5	220
	A_2	7.5	1200
	B_1	7.5	1200
	B_2	-4.5	220
	Bias	-10	100
(5)	A_1	-20	50
	A_2	-30	33
	B_1	10	1200
	B_2	25	500
	N_3	-25	40
	Bias	17.5	700
(6)	N_1	10	1000
	N_3	10	1000
	Bias	-5	220
(7)	N_1	10	1000
	N_4	10	1000
	Bias	-5	220
(8)	N_1	10	1000
	N_2	10	1000
	Bias	-5	220

Table VII
Practical and Simulation results after the summing circuit of full-adder/full-subtractor

I/p	Neuron(1)		Neuron(2)		Neuron(3)	
X Y Z	Practical	Simulated	Practical	Simulated	Practical	Simulated
0 0 0	-2.79	-3.4157	-2.79	-3.4135	-2.79	-3.4135
0 0 1	-2.73	-2.5968	3.46	3.3741	3.46	3.3741
0 1 0	-2.73	-2.5968	3.46	3.2741	3.46	3.3741
0 1 1	3.46	3.3761	3.46	3.4366	-2.75	-3.3081
1 0 0	-2.73	-2.5968	-2.79	-3.4372	3.46	3.3741
1 0 1	3.46	3.3761	-2.75	-3.3081	-2.75	-3.3081
1 1 0	3.46	3.3761	-2.75	-3.3081	-2.75	-3.3081
1 1 1	3.46	3.4231	3.48	3.4120	3.48	3.4120

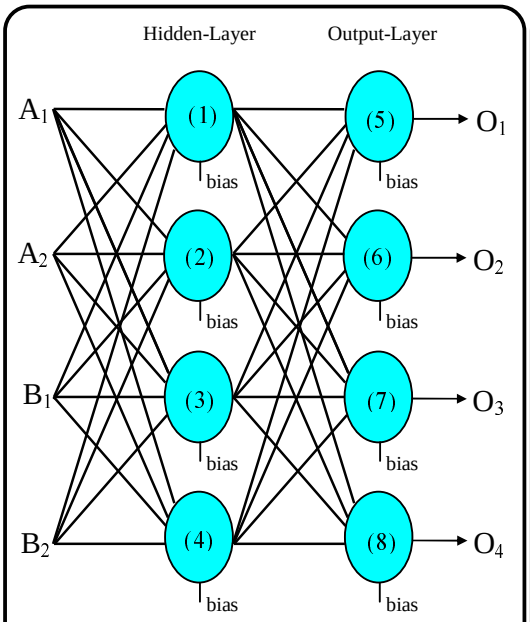

Fig. 5. 2-Bit digital multiplier using traditional feed-forward neural network

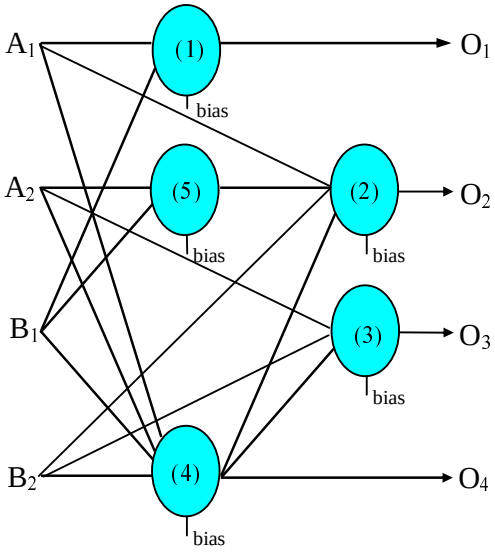

Fig. 6. A novel design for2-Bit multiplier using neural network

Table VIII
Practical and Simulation results after the summing circuit of 2-bit digital multiplier

Neuron (1)		Neuron (2)		Neuron (3)		Neuron (4)		Neuron (5)	
Pract.	Sim.	Pract.	Sim.	Pract.	Sim.	Pract.	Sim.	Pract.	Sim.
-2.79	-3.415	-2.79	-3.409	-2.79	-3.413	-2.79	-3.447	-2.79	-3.415
-2.34	-2.068	-2.72	-2.498	-2.79	-3.314	-2.78	-3.438	-2.79	-3.415
-2.79	-3.415	-2.79	-3.409	-1.63	-1.355	-2.78	-3.438	-2.34	-2.068
-2.34	-2.068	-2.72	-2.498	-1.63	-1.355	-2.78	-3.423	-2.34	-2.068
-2.34	-2.068	-2.79	-3.409	-2.79	-3.413	-2.78	-3.438	-2.34	-2.068
3.46	3.390	-2.72	-2.498	-2.79	-3.413	-2.78	-3.423	-2.34	-2.068
-2.34	-2.068	3.45	3.397	-1.63	-1.355	-2.78	-3.423	3.46	3.390
3.46	3.390	3.45	3.424	-1.63	-1.355	-2.74	-3.384	3.46	3.390
-2.79	-3.415	-2.72	-2.498	-1.63	-1.355	-2.78	-3.438	-2.79	-3.415
-2.34	-2.068	3.45	3.373	-1.63	-1.355	-2.78	-3.423	-2.79	-3.415
-2.79	-3.415	-2.72	-2.498	3.45	3.399	-2.78	-3.423	-2.34	-2.068
-2.34	-2.068	3.45	3.373	3.45	3.399	-2.74	-3.384	-2.34	-2.068
-2.34	-2.068	-2.72	-2.498	-1.63	-1.355	-2.78	-3.423	-2.34	-2.068
3.46	3.390	3.45	3.373	-1.63	-1.355	-2.74	-3.384	-2.34	-2.068
-2.34	-2.068	3.45	3.373	3.45	3.399	-2.74	-3.384	3.46	3.390
3.46	3.390	-2.73	-3.398	-2.70	-2.710	1.86	2.519	3.46	3.390

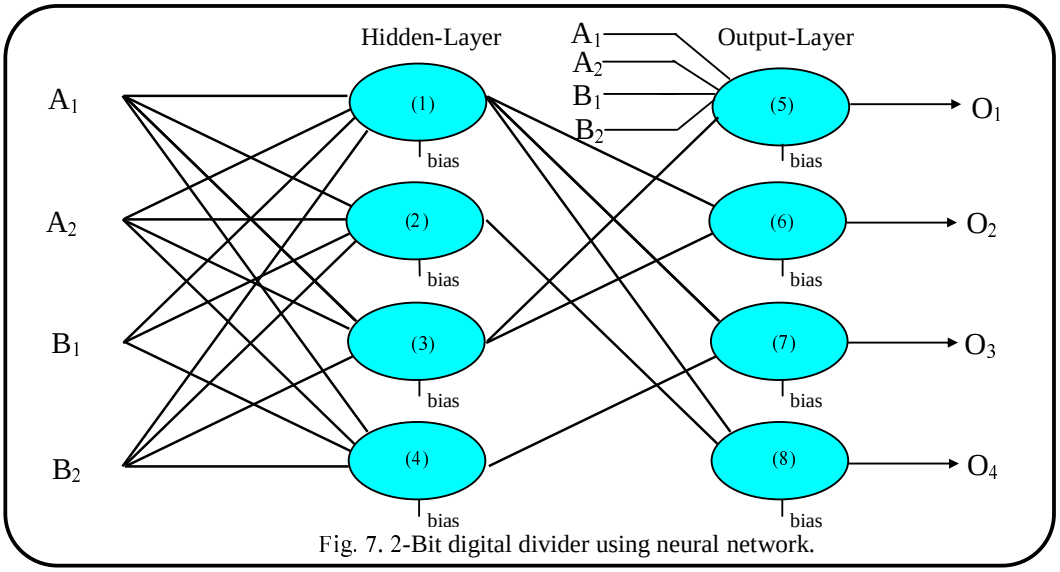

Fig. 7. 2-Bit digital divider using neural network.

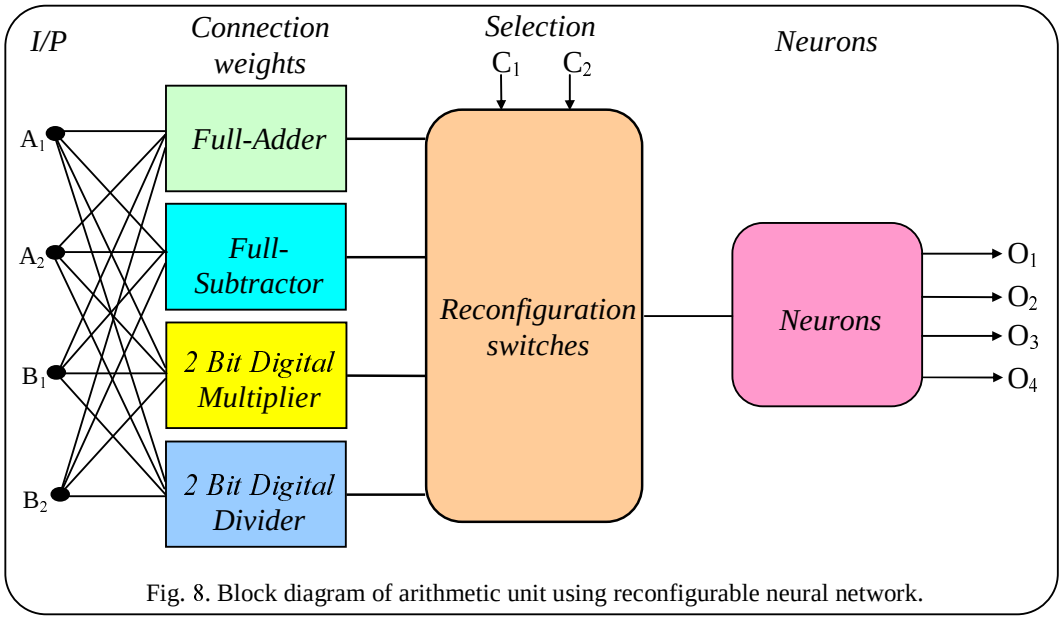

Fig. 8. Block diagram of arithmetic unit using reconfigurable neural network.