Multilevel Logic Multiplier Using VLSI Neural Network

Huadian Pan[1], Milos Manic[1], Xiangli Li[2], Bogdan Wilamowski[2]

[1]University of Idaho, Graduate Center at Boise, 800 Park Blvd., Suite 200, Boise, Idaho 83712

[2]Auburn University, Department of Electrical and Computer Engineering, 200 Broun Hall, Auburn, Alabama 36849

Abstract – **this paper presents a solution for digital multiplier implemented with cascade connected neural network architecture. Proposed solution uses multilevel logic where information is compressed. To facilitate VLSI implementation, low voltage current mode operation is being used in this paper. Multiplier design is presented as a summator where the result is provided in one clock cycle. The system is fully simulated with SPICE and the chip is fabricated in the AMI 1.5um MOSIS process.**

I. INTRODUCTION

Hardware implementation can be a very difficult and expensive process due to complex learning algorithm computations. Many references provide literature review on this subject. Specific implementations include VLSI and CMOS implementation [1-5], FPGA's [6], probabilistic RAM, and others [7-10].

The problem considered in this paper was VLSI multiplier realized by multilevel logic neural network. Proposed algorithm aims to impose the minimum requirements for hardware implementation by simplifying computational process.

Objective was to implement VLSI multiplier through a summator realized through neural network. The idea was to propose a low voltage current mode operation of neuron that would facilitate VLSI implementation. To develop an algorithm that would provide one clock time result through a single pass through the network. Proposed solution uses multilevel logic where information is compressed. This idea is analogue to modem communication (baud rates), which implies possible further benefits.

A neural unit, which has a functionality that can be understood as AD (analog-to-digital) converter, will be used often in this design; for convenience, just call it PU (Processing Units) for short. PUn means this unit has one resulting bit and n carry bits.

II. NEURAL IMPLEMENTATION OF PROCESSING UNITS

Processing units used in proposed solution for VLSI multiplier are implemented through cascade neural network architecture. This solution imposes minimum requirements with regards to number of neurons at the same time annulling problems of error propagation. For example, for processing 15 positive signals at input, architecture requires only 4 neurons. Other architectures, such as pipeline type of networks might also provide a solution with minimum number of neurons. However, such networks would include possible signal loss, unacceptable in this type of application.

The functionality of these units can be understood as specific AD (analog-to-digital) converter, where the result is obtained through single pass.

Essential characteristics of such implementation are as follows. This solution offers minimum signal loss, i.e. propagation of error with minimum number of neurons. It could be described as an analogue to digital neural architecture, fully connected with one hidden layer (in other words, fully cascade connected).

Neurons are unipolar with hard threshold activation function. Similar architecture would apply for bipolar neurons. The chosen neural network architecture is simpler for hardware implementation and provides faster signal propagation. Fully connected architecture would traditionally have all inputs fed to all neurons, where the architecture proposed in this paper first sums all the inputs and then feeds the resulting signal to other neurons in cascade architecture. First neuron serves as a summator, i.e. one neuron with linear activation function. Other neurons have hard threshold activation function

To facilitate further VLSI implementation, it can be further implemented in low voltage current mode operation. Since each processing unit uses summator, therefore current mode was chosen (currents are easier to sum than voltage).

The functional implementation of processing units PU0 & PU1 units is illustrated by Fig. 1, while PU2 & PU3 units are shown by Fig. 2.

Logic tables for units AND1, PU2, and PU3 are given by Table 1, Table 2, and Table 3. PU0 unit has a simply parity 2 functionality.

0-7803-7852-0/03/$17.00 ©2003 IEEE

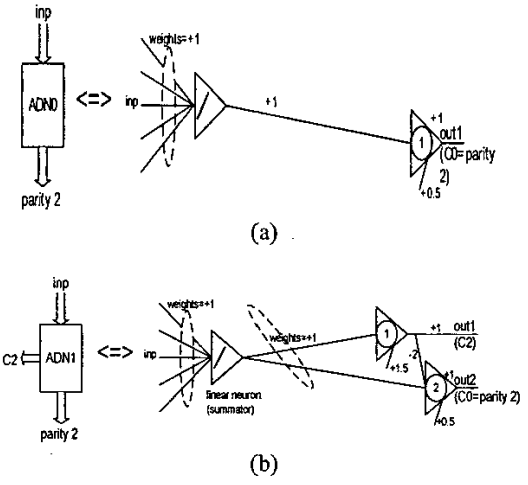

Fig.1 PU0 & PU1 units.
(a) PU0 capable of parity 2 calculation.
(b) PU1 capable of parity 2 calculation and carry 2.

Decimal value	Number of 1.s on inputs	Binary value	O1O2 (C2C0)
0	0	0 0	0 0
1	1	0 1	0 1
2	1	1 0	1 0
3	2	1 1	1 1

Table 1 Logic table for PU1 unit.

Decimal value	Number of 1.s on inputs	Binary value	O1O2O3 (C4C2C0)
0	0	0 0 0	000
1	1	0 0 1	001
2	1	0 1 0	001
3	2	0 1 1	010
4	1	1 0 0	001
5	2	1 0 1	010
6	2	1 1 0	010
7	3	1 1 1	011

Table 2 Logic table for PU2 unit

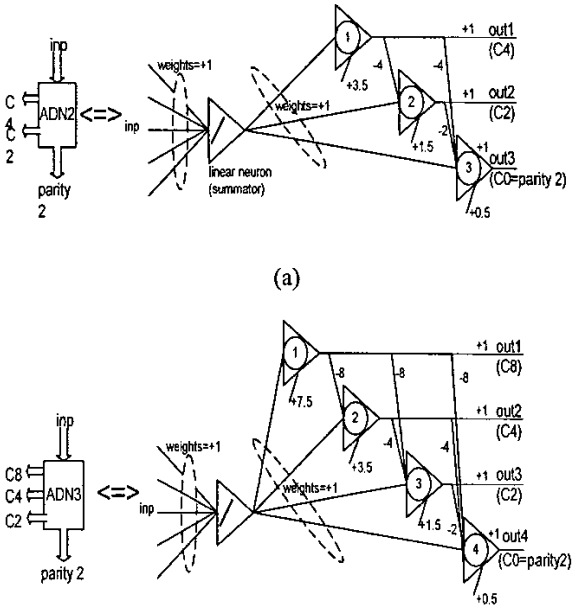

Fig.2 PU2 & PU3 units.
(a) PU2 capable of parity 2 calculation and carry 2 and 4.
(b) PU3 capable of parity 2 calculation and carry 2, 4, and 8.

Number of 1.s on inputs	O1O2O3O4 (C8C4C2C0)
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
10	1 0 1 0
11	1 0 1 1
12	1 1 0 0
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Table 3 Logic table for PU3 unit

III. FUNCTIONAL DESIGN

A. 4x4 multiplier

This was the simplest case considered. For this type of multiplier, the "worst" case with regards to number of ones is the one given by table 4. Implementation proposed in this paper is using neural units (described in section II). Possible implementation using proposed processing units is illustrated in Fig. 3.

Summarized requirements on fields and bits required for the implementation presented by Fig. 3 are given in Table 5. Table 6 illustrates the functionality of 4x4 multiplier realized as summator.

Worst case:

$$\begin{array}{r} 1111 \\ \times\quad 1111 \\ \hline 1110\ 0001 \end{array}$$

Table 4 Worst-case scenario for the 4x4 multiplier

7	6	5	4	3	2	1	0	Bit number
2	3	4	5	5	4	2	1	Fields to add
1	2	2	2	2	2	2	1	A->D (digital bit output)

Table 5 Summarized requirements for realization of 4x4 multiplier

4,3	4,2	4,1	4,0		
5,4	5,3	5,2	5,1	0	
6,5	6,4	6,3	6,2	1c	
7,6	7,5	7,4	7,3	1c	2
5c	5c	4c	2c	2c	5
6c	6c	3c	3c		
7	6	5	4		

Table 6 4x4 multiplier functionality as a summator, table representation.

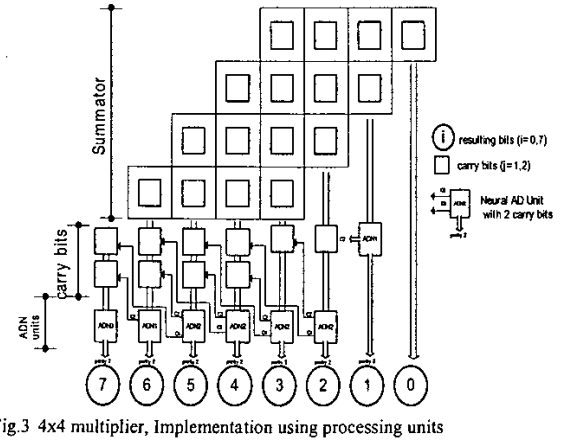

Fig.3 4x4 multiplier, Implementation using processing units

B. 8x8 multiplier

Functionality of 8x8 multiplier realized as summator is given by Table 9. Bits are denoted by colors, i.e. green – original bits (row, number of final bit), blue – final bits, and yellow – carry bits.

Summarized requirements for 8x8 multiplier realized as a summator are given by table 8, while the worst case scenario is illustrated by table 7. Possible implementation using processing units is illustrated by Fig.4.

General case NxN multiplier would require *N*-number of bits in one register and *2N* bits in resulting register.

Worst case:

$$\begin{array}{r} 1111\ 1111 \qquad 255\text{x}255=65025 \\ \times\quad 1111\ 1111 \\ \hline 1111\ 1110\ 0001\ 0001 \end{array}$$

Table 7 Worst case scenario for 8x8 multiplier

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Bit number
2	3	5	6	7	8	9	10	10	9	8	7	5	4	2		Fields to add
1	2	3	3	3	4	4	4	4	4	4	3	3	3	2	1	A->D

Table 8 Summarized requirements for realization of 8x8 multiplier

7,7	7,6	7,5	7,4	7,3	7,2	7,1	7,0		
8,8	8,7	8,6	8,5	8,4	8,3	8,2	8,1	0	
9,9	9,8	9,7	9,6	9,5	9,4	9,3	9,2	1	
10,10	10,9	10,8	10,7	10,6	10,5	10,4	10,3	1c	2
11,11	11,10	11,9	11,8	11,7	11,6	11,5	11,4	2c	3
12,12	12,11	12,10	12,9	12,8	12,7	12,6	12,5	2c	3c	4
13,13	13,12	13,11	13,10	13,9	13,8	13,7	13,6	4c	3c	5
14,14	14,13	14,12	14,11	14,10	14,9	14,8	14,7	4c	5c	6
	12c	12c	9c	9c	9c	6c	6c	6c	5c	7
	13c	10c	10c	10c	7c	7c	7c			
	14	11c	11c	8c	8c	8c	5c			
		13	12	11	10	9	8			
14c										
13c			15c							
15										

Table 9 8x8 multiplier functionality as a summator, table representation

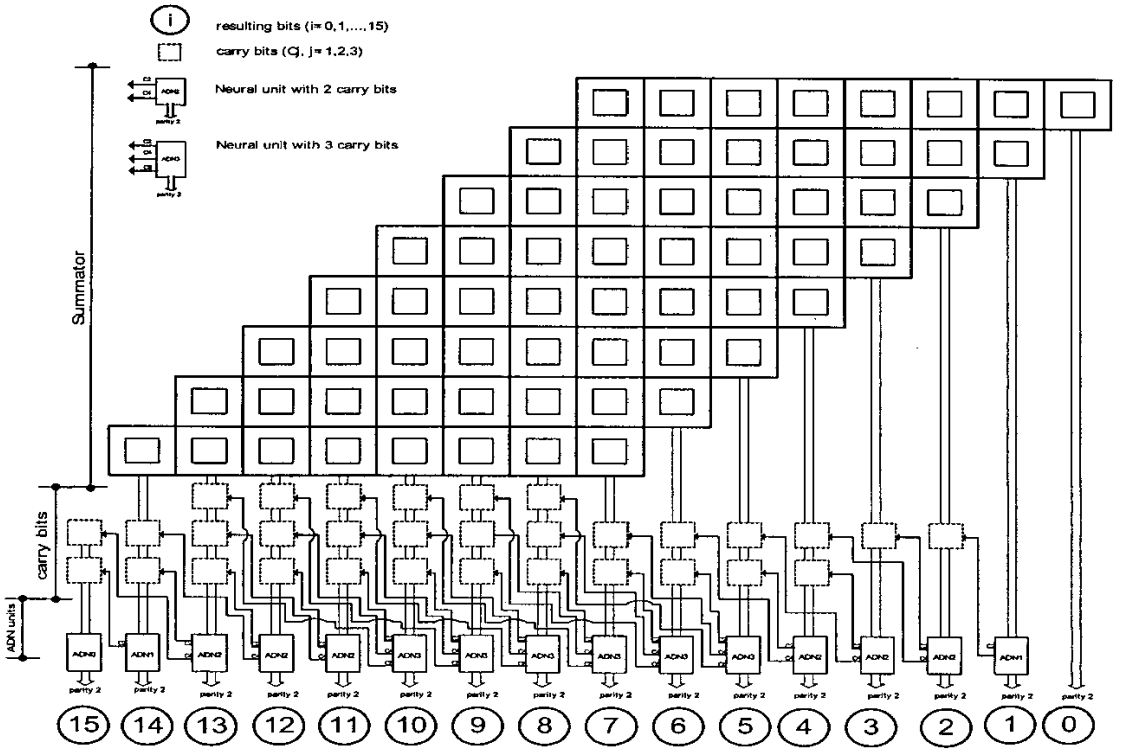

Fig.4 8x8 multiplier, functionality and implementation using PU units

IV. VLSI IMPLEMENTATION

The VLSI implementation of Unipolar neuron with hard threshold activation function is shown as Fig 5.

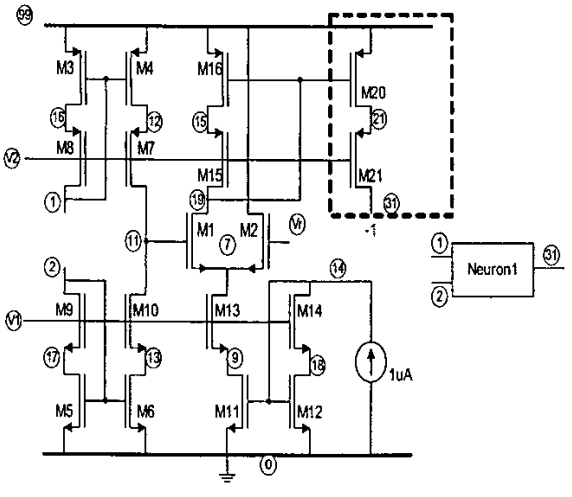

Fig.5 The schematic circuit of the neuron1

In the circuit on Fig 5, node 1 is the positive input, and node 2 is the negative input, which would be used for the threshold setting, node 31 is the output node. With the input current changing, the voltage of the node 11 will change between 0 to 4.8V roughly, so the transistor M1 will turn on and off with the input current changing. When the input current is smaller than the threshold current, no current is at the output node, otherwise, there will a 1 uA current detected from the output node.

The design was verified by SPICE simulation [12] with AMI 1.5um SPICE transistor model, the W/L ratios of the transistor M7, M8 and M21 are set as 15/5; the others are set as 8/1.6. The simulation results are shown in Fig. 6. In this simulation, the threshold current is set as 0.5uA. The input DC current is sweeping from 0 to 1uA as the X axis showing. From figure 6(a), it can be seen that the voltage of node 11 varies between 0 and 5 voltages with the input current sweeping. Fig. 6(b) shows the output current changing with the input current sweeping. From the Fig 6(b) it can be seen that when the input current is smaller than the 0.5uA, the output current is zero, and when the input current larger than the threshold current, there will be a 1uA output current detected from the output node. So with this circuit, the unipolar neuron can be implemented.

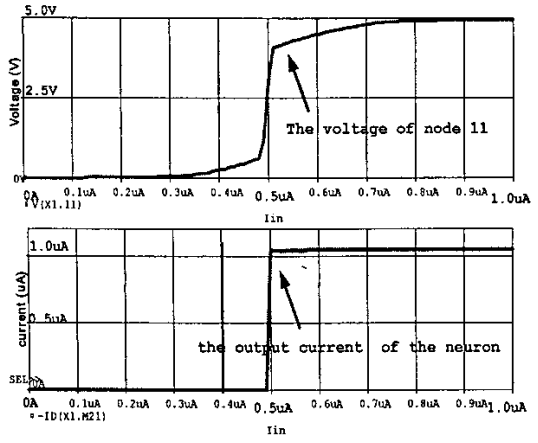

Fig.6 The simulation of the neuron (a: the voltage of node 11, b: the output current of the neuron)

Replacing the two output transistors in the neuron1 as the dash line box showing with the circuit shown in the Fig 7A, the neuron2 circuit which has one 1x output and one 2x output can be implemented. Similarly, the neuron3 circuit with one 1x output and two 4x outputs can be implemented with replacing the circuit block shown in Fig. 7B.

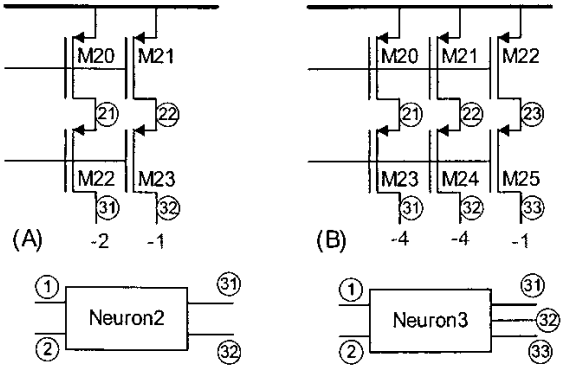

Fig.7 Block diagram of Neuron2 and Neuron3.

Then PU2 unit can be built with these three neurons. Fig.8 shows the block diagram of the PU2 unit. The threshold current of the neuron1, neuron2 and neuron3 were set as 0.5uA, 1.5uA, and 3.5uA respectively. One of the 4x outputs of neuron3 is connected to the negative input of neuron1, the other is connected to the negative input of neuron2, also, and the 2x output of neuron2 is connected to the negative input of neuron1. With this configuration, the PU2 behaves like a 3 bit AD converter.

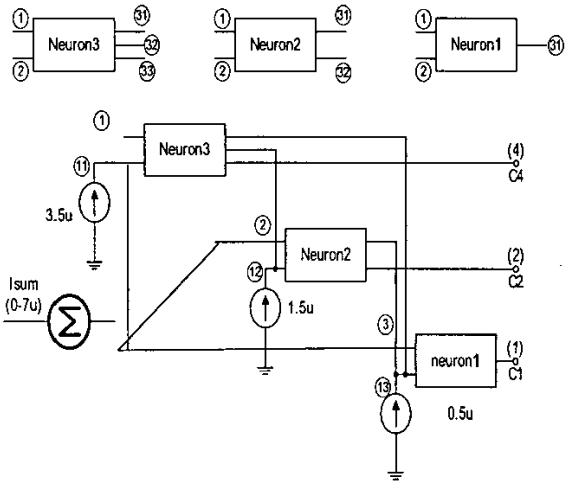

Fig.8 The block diagram of the PU2 unit

The SPICE simulation results of the PU2 are shown in the Fig. 9. From the simulation result, it can be seen that the PU2 unit works very well. For example, when the input from the current summator is 5uA, the output of c4, c2 and c1 will be 1, 0, 1, and so on.

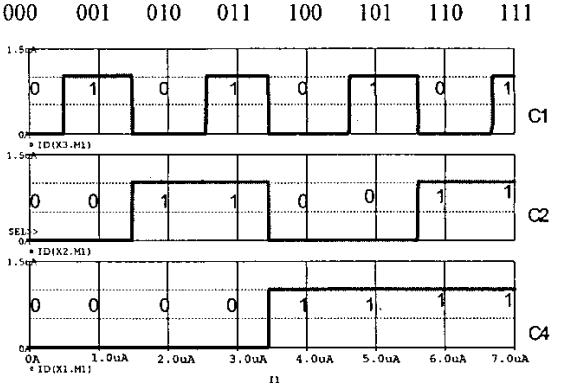

Fig.9 The PSPICE simulation of the PU2 (Isum sweep from 0 to 7uA)

LASI is being used as the CAD tool to layout the chip [11]. The layout of this PU2 is showed in Fig.10. The layout is based on with AMI 1.5um MOSIS technology.

SPICE netlists of the PU2 unit were extracted with LASI automatically. The simulation results from the extracted netlists and the pre-layout simulation show reasonable agreement.

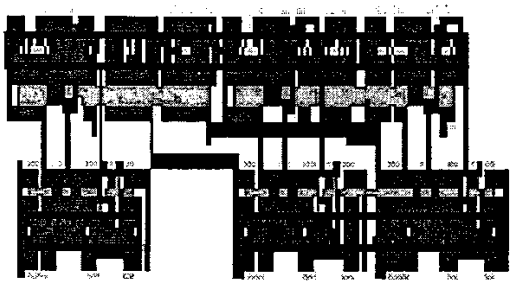

Fig.10 The layout of PU2 with LASI

The microphotograph of a 6x6 multiplier is shown in Fig.11. The chip size is 1.5mmx1.5mm, the chip was fabricated in the AMI 1.5um ABN process.

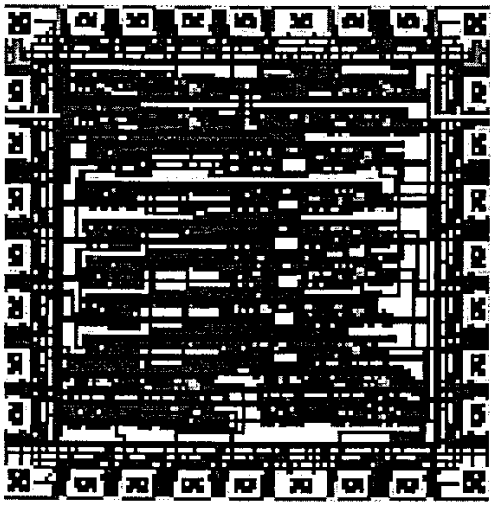

Fig.11 Microphotograph of the 6x6 multiplier layout, the chip size is 1.5mmx1.5mm

V. CONCLUSION

Ideas on VLSI multiplier realized by multilevel logic neural network were presented in this paper. Low voltage current mode neuron operation facilitates VLSI implementation. Multiplier design was presented as a summator where the result is provided in one clock cycle, i.e. by single pass through the network. Future benefits of this multilevel-logic implementation include information compression which is similar to the idea with data compression in modem communication.

REFERENCES

[1] U. Ramacher, U. Ruckert, VLSI design of neural networks, Boston: Kluwer Academic Publishers, 1991.

[2] B M Wilamowski, J. Binfet, M O Kaynak, "VLSI Implementation of Neural Networks" International journal of neural systems. 10, no. 3, pp191-199, 2000.

[3] Y. Ota, B. M. Wilamowski, "CMOS Implementation of a Pulse-Coded Neural Network with a Current Controlled Oscillator", IEEE International Symposium on Circuits and Systems. No. 3, 1996.

[4] M. Carver, M. Ismail, Analog VLSI implementation of neural systems, Boston: Kluwer Academic Publishers, 1989.

[5] F. Gregoretti, R Passerone, L.M. Reyneri, et.al, "A High Speed VLSI Architecture for Handwriting Recognition", The Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology 28, no. 3 pp259-278, 2001.

[6] N. M. Botros, M. Abdul-Aziz, "Hardware Implementation of an Artificial Neural Network Using Field Programmable Gate Arrays (FPGA's)", IEEE transactions on industrial electronics: Vol. 41, no. 6, pp. 665-972, 1994.

[7] Bernbard E. Boser, Eduard Sackinger, Jane Bromley, Yann, leCun, Lawrence D. Jackel, "Hardware Requirements for Neural Network Pattern Classifiers: A Case Study and Implementation", IEEE micro. 12, no. 1, pp32-41, 1992.

[8] R. J. Haycock, T. A. York, "Hardware implementation of a pulse-stream neural network", IEE proceedings. Circuits, devices, and systems. 145, no. 3, pp.141-148, 1998.

[9] H. Hikawa, "An Efficient Three-Valued Multilayer Neural Network with On-Chip Learning Suitable for Hardware Implementation", Systems and computers in Japan. 31, No. 4, pp.43-52, 2000.

[10] Kenneth R. Crounse, Eula L. Fung, Leon O. Chua, "Hardware implementation of cellular automata via the Cellular Neural Network Universal Machine", Berkeley Electronics Research Laboratory, College of Engineering, University of California, 1995.

[11] R. Jacob Baker, Harry W.Li, David E.Boyce, "CMOS circuit design, layout and simulation", publisher: Wiley-IEEE Press, August 1997.

[12] Bogdan M. Wilamowski, Richard C.Jaeger, "Computerized circuit analysis using SPICE program", The McGraw-Hill Companies, Inc., August 1997.