AMSE JOURNALS-2016-Series: Advances D; Vol. 21; N°1; pp 54-65
Submitted July 2016; Revised Oct. 11, 2016, Accepted Nov. 15, 2016

Design of Basic Logic Gates using CMOS and Artificial Neural Networks (ANN)

R. K. Mandal

Department of Computer Science & Application, University of North Bengal
Raja Rammohunpur, PO : NB Distt: Darjeeling, West Bengal – 734013, India
(rakesh_it2002@yahoo.com   http://www.nbu.ac.in)

Abstract: **-** This paper shows an approach to simplify the electronic circuits by using Complementary Metal Oxide Semiconductors (CMOS) transistors and develop the equivalent Artificial Neural Networks (ANNs). The development of these types of circuits leads to the simple hardware implementation of ANN models. In this paper the three basic gates already implemented using CMOS are designed by using simple Self Organizing Map (SOM) ANNs, which are unsupervised.

Key-words: - Complementary Metal Oxide Semiconductors (CMOS), Artificial Neural Networks (ANNs), Self Organizing Map (SOM), Gates

I. Introduction

Now-a-days computing is converging towards intelligent computing. Intelligence can be best implemented in computers using Artificial Neural Networks (ANNs) [1, 2]. Research is going on to develop models which can be used in various applications like medical informatics, handwriting recognition, speech recognition, and other applications of pattern recognition [3, 4, 5, 6]. After the development of these models the major challenge is to implement these models in the hardware circuits. If ANNs are developed equivalent to the CMOS circuits, then it becomes simple to map ANNs to hardware circuits using CMOS [7, 8, 9, 10].

Forssell M has worked in the field of hardware implementation of Artificial Neural Networks [11]. Work has already been done in this field where CMOS circuit was designed that accepts synapses inputs and generates pulse width modulated output waveform of constant frequency on the basis of activation level [12]. Logic gates are implemented in single layer and two layers feed forward neural network based supervised learning [13]. In an approach Artificial Neural Network (ANN) is used to demonstrate the way in which the biological system is processed in analog domain by using analog component like Gilbert cell multiplier, Adder, Neuron activation function for implementation [14]. Hui W et al worked on the use of artificial neural networks on segmented arc heather failure prediction [15].

This paper has been divided into three sections. Section-1 discusses the implementation of a simple ANN CMOS NOT gate. Section-2 discusses the implementation of a simple ANN CMOS AND gate. Section-3 explains the implementation of a simple ANN CMOS OR gate.

II. NOT-Gate using CMOS and Equivalent ANN

In Figure 1, an ANN has been developed which can replace a NOT gate. The ANN given in the above figure has been developed using CMOS, where ‘x’ is the input of ANN and ‘y’ is the output. ‘T_1’ and ‘T_2’ are the artificial neurons which replaces two transistors of the CMOS NOT gate. The ‘x’ and ‘V_{dd}’ are the inputs of ‘T_1’ with weights ‘w_1’ and ‘w_2’. The ‘x’ and ‘V_{ss}’ are the inputs of ‘T_2’ with weights ‘w_2’ and ‘w_4’. Another neuron ‘R’ is used as referee neuron. The outputs ‘p’ and ‘q’ of the neuron ‘T_1’ and ‘T_2’ are the inputs to referee neuron with weights ‘w_5’ and ‘w_6’. The output of the referee neuron is ‘y’ which is the output of the NOT gate. Neuron ‘T_1’ uses the following algorithm to generate the output.

Algorithm 1: Output generation of upper transistor of the ANN CMOS NOT gate.

Step 1: The weights 'w_1' and 'w_3' are fixed and assigned values as given below:

'w_1' = -1 and 'w_3' = 1

Step2: Assign 'v_{dd}' = -1

Step3: Calculate 'y_1' = $x*w_1$ and 'y_2' = $v_{dd}*w_3$

Step4: If 'y_1' >= 1 then 'p' = 1 else 'p' = y_2

Step 5: Stop

Neuron 'T_2' uses the following algorithm to generate the output.

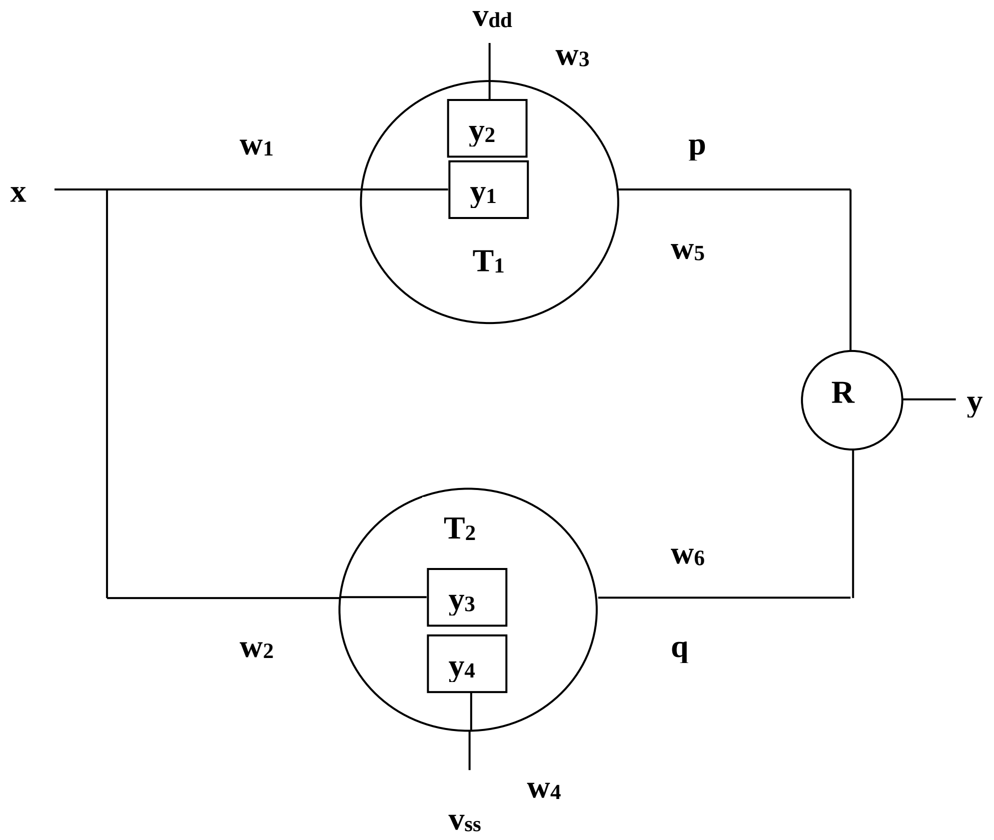

Figure 1: ANN NOT-Gate using CMOS

Algorithm 2: Output generation of lower transistor of the ANN CMOS NOT gate.

Step 1: The weights 'w_2' and 'w_4' are fixed and assigned values as given below:

'w_2' = 1 and 'w_4' = 1

Step2: Assign 'v_{ss}' = 1

Step 3: Calculate 'y_3' = $x*w_2$ and 'y_4' = $v_{ss}*w_4$

Step 4: If ‘y_3’ < 0 the ‘q’ = y_4 else ‘q’ = -1

Step 5: Stop

The referee neuron is a simple perceptron having two inputs without a bias and an output which is the output of the inverter.

Example 1: This example shows the working of an ANN-CMOS inverter.

Let us consider the value of ‘x’ as ‘-1’.

$w_1 = -1$ and $w_3 = 1$

$v_{dd} = -1$

$y_1 = (-1) \times (-1)$

$y_1 = 1$

$y_2 = (-1) \times 1$

$y_2 = -1$

$y_1 = 1$ therefore $p = 1$

Now, $w_2 = 1$ and $w_4 = 1$

$v_{ss} = 1$

$y_3 = (-1) \times 1$

$y_3 = -1$

$y_4 = 1 \times 1$

$y_4 = 1$

$y_3 < 0$ therefore $q = 1$

Now $p = 1$ and $q = 1$

Table 1: Truth Table for ANN-CMOS NOT gate

x	w_1	w_2	w_3	w_4	w_5	w_6	v_{dd}	v_{ss}	y_1	y_2	y_3	y_4	p	q	y_out	y
-1	-1	1	1	1	1	1	-1	1	1	-1	-1	1	1	1	2	1
1	-1	1	1	1	1	1	-1	1	-1	-1	1	1	-1	-1	-2	-1

$w_5 = 1$ and $w_6 = 1$

Applying perceptron learning rule:

$y\_out = p*w_5 + q*w_6$

$y\_out = (1 \times 1) + (1 \times 1)$

$y\_out = 2$ which is > 0 therefore $y = 1$

[Hence, for x = -1 we get y = 1]

Let us consider the value of 'x' as '1'.

$w_1 = -1$ and $w_3 = 1$

$v_{dd} = -1$

$y_1 = 1 \times (-1)$

$y_1 = -1$

$y_2 = (-1) \times 1$

$y_2 = -1$

$y_1 = -1$ which is less than '0' therefore $p = -1$

Now, $w_2 = 1$ and $w_4 = 1$

$v_{ss} = 1$

$y_3 = 1 \times 1$

$y_3 = 1$

$y_4 = 1 \times 1$

$y_4 = 1$

$y_3 = 1$ therefore $q = -1$

Now $p = -1$ and $q = -1$

$w_5 = 1$ and $w_6 = 1$

Applying perceptron learning rule:

$y\_out = p*w_5 + q*w_6$

$y\_out = (-1 \times 1) + (-1 \times 1)$

$y\_out = -2$ which is < 0 therefore $y = -1$

[Hence, for x = 1 we get y = -1]

III. AND-Gate using CMOS and Equivalent ANN

In Figure 2, an ANN has been developed which can replace an AND gate. The ANN given in the figure has been developed using CMOS, where 'x_1' and 'x_2' are the inputs of ANN and 'y' is the output. 'T_1', 'T_2', 'T_3' and 'T_4' are the artificial neurons which replace four transistors of the CMOS AND gate. 'x_1' and 'V_{dd}' are the inputs of 'T_1' with weights 'w_1' and 'w_2' and output of the neuron is 't_1'. 'x_2' and 'V_{dd}' are the inputs of 'T_2' with weights 'w_8' and 'w_7' and output of the neuron is 't_2'. 'x_1' and 't_1' are the inputs of 'T_3' with weights 'w_3' and 'w_4' and output of the neuron is 't_3'. 'x_2' and 't_3' are the inputs of 'T_4' with weights

‘w_5’ and ‘w_6’ and output of the neuron is ‘t_4’. Another neuron ‘R’ is used which is also called the referee neuron. ‘t_1’ and ‘t_2’ are the inputs of ‘R’ with weights ‘w_9’ and ‘w_{10}’ and output of the neuron is y’, which is presented to an ANN CMOS NOT gate to generate the output of the ANN CMOS AND gate.

The transistors ‘T_1’ and ‘T_2’ are reverse biased so the weights of the ‘x_i’ inputs ‘w_1’ and ‘w_8’ are set to ‘-1’ and weights of ‘v_{dd}’ inputs ‘w_2’ and ‘w_7’ are also set to ‘-1’. The transistors ‘T_3’ and ‘T_4’ are forward biased so the weights of the ‘x_i’ inputs ‘w_3’ and ‘w_5’ are set to ‘1’ and other weights ‘w_4’, ‘w_6’, ‘w9’ and ‘w10’ are also set to ‘1’.

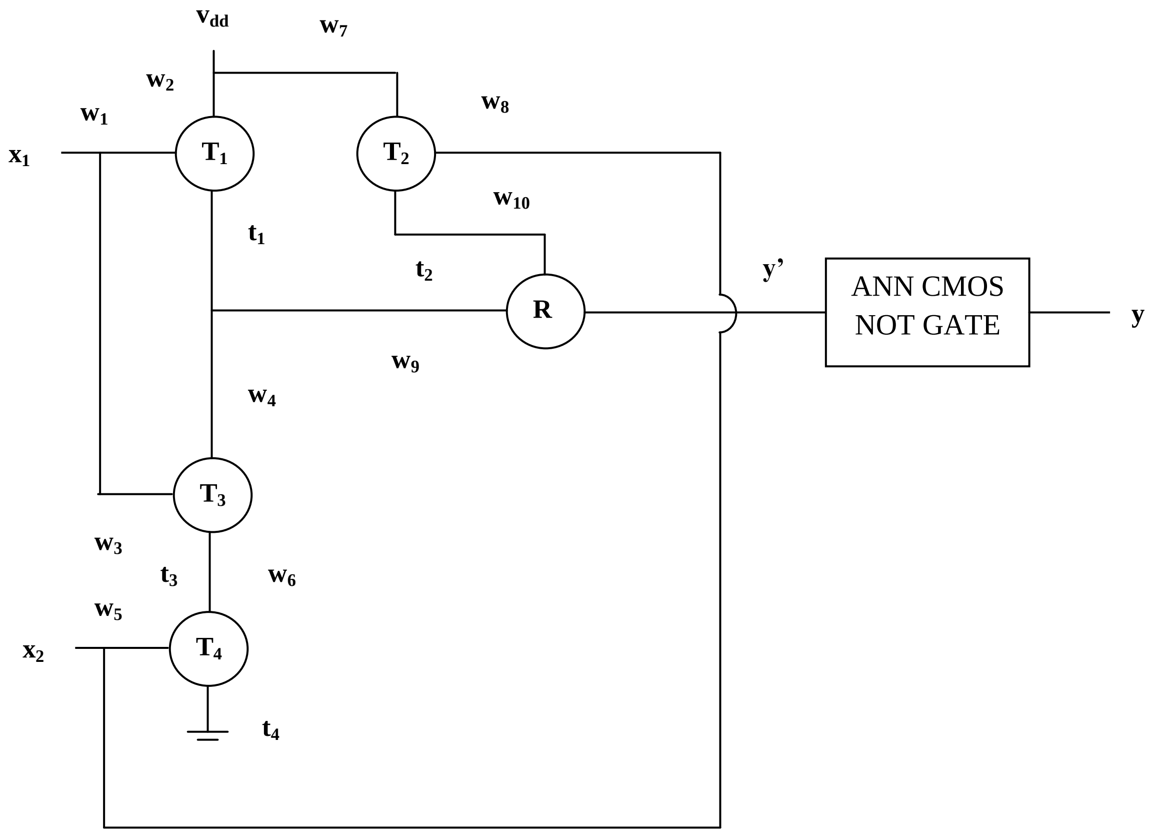

Figure 2: ANN AND-Gate using CMOS

For the transistors T_1, T_2, T_3, T_4 and R, the outputs are calculated as given in the following equations:

$t_{out1} = x_1 * w_1 + v_{dd} * w_2$ ….Equation 1

$t_1 = f(t_{out1})$ ….Equation 2

where, $f = \{ 1 \text{ if } t_{out1} >= 0 \text{ else } -1\}$

Similarly,

$t_{out2} = x_2*w_8 + v_{dd}*w_7$ ….Equation 3

$t_2 = f(t_{out2})$ ….Equation 4

$t_{out3} = x_1*w_3 + t_1*w_4$ ….Equation 5

$t_3 = f(t_{out3})$ ….Equation 6

$t_{out4} = x_2*w_5 + t_3*w_6$ ….Equation 7

$t_4 = f(t_{out4})$ ….Equation 8

Table 2: Weight Matrix for ANN-CMOS AND gate

x_1	x_2	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
-1	-1	-1	-1	1	1	1	1	-1	-1	1	1
-1	1	-1	-1	1	1	1	1	-1	-1	1	1
1	-1	-1	-1	1	1	1	1	-1	-1	1	1
1	1	-1	-1	1	1	1	1	-1	-1	1	1

Table 3: Truth Table for ANN-CMOS AND gate

x_1	x_2	t_1	t_2	t_3	t_4	y’	y
-1	-1	1	1	1	1	1	-1
-1	1	1	-1	1	1	1	-1
1	-1	-1	1	1	1	1	-1
1	1	-1	-1	1	1	-1	1

$y'_{out} = t_2*w_{10} + t_1*w_9$ ….Equation 9

$y' = f(y'_{out})$ ….Equation 10

Table 2 displays the weight matrix for ANN-CMOS AND gate for different possible inputs of an AND gate. Table 3 displays the truth table of ANN-CMOS AND gate.

IV. OR-Gate using CMOS and Equivalent ANN

In Figure 3, an ANN has been developed which can replace an OR gate. The ANN given in the figure has been developed using CMOS, where ‘x_1’ and ‘x_2’ are the inputs of ANN and ‘y’ is the output. ‘T_1’, ‘T_2’, ‘T_3’ and ‘T_4’ are the artificial neurons which replace four transistors of the CMOS OR gate. ‘x_1’ and ‘v_{dd}’ are the inputs of ‘T_1’ with weights ‘w_1’ and ‘w_2’ and output of the neuron is ‘t_1’. ‘x_2’ and ‘v_{dd}’ are the inputs of ‘T_2’ with weights ‘w_3’ and ‘w_4’ and output of the neuron is ‘t_2’. ‘x_1’ and ‘v_{dd}’ are the inputs of ‘T_3’ with weights ‘w_5’ and ‘w_6’ and output of the neuron is ‘t_3’. ‘x_2’ and ‘v_{dd}’ are the inputs of ‘T_4’ with weights 'w_7' and 'w_8' and output of the neuron is 't_4'. Another neuron 'R' is used which is also called the referee neuron. 't_3' and 't_4' are the inputs of 'R' with weights 'w_9' and 'w_{10}' and output of the neuron is y, which generates the output of the ANN CMOS OR gate. The transistors 'T_1' and 'T_2' are reverse biased so the weights of the 'x_i' inputs 'w_1' and 'w_3' are set to '-1' and weights of 'v_{dd}' inputs 'w_2' and 'w_4' are also set to '-1'. The transistors 'T_3' and 'T_4' are forward biased so the weights of the 'x_i' inputs 'w_5' and 'w_7' are set to '1' and other weights 'w_6', 'w_8', 'w_9' and 'w_{10}' are also set to '1'.

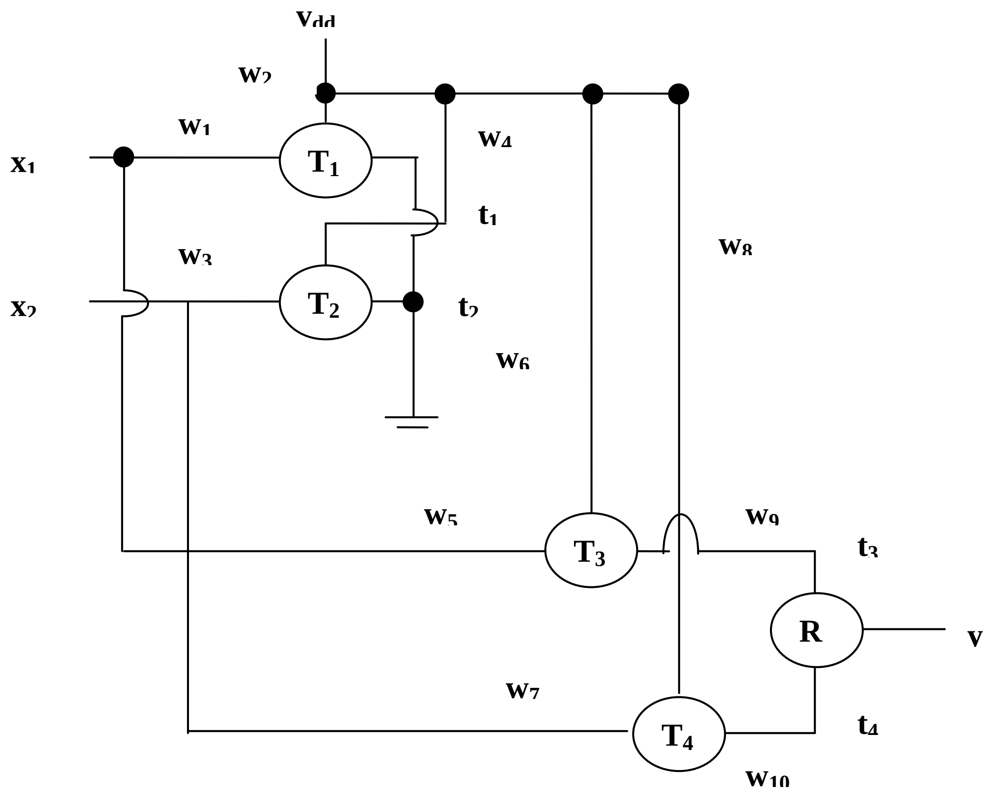

Figure 3: ANN OR-Gate using CMOS

For the transistors T_1, T_2, T_3, T_4 and R, the outputs are calculated as given in the following equations:

$t_{out1} = x_1*w_1 + v_{dd}*w_2$ ….Equation 1

$t_1 = f(t_{out1})$ ….Equation 2

Where, f = { 1 if $t_{out1} > 0$ else -1}

Similarly,

$t_{out2} = x_2*w_3 + v_{dd}*w_4$ ….Equation 3

$t_2 = f(t_{out2})$ ….Equation 4

$t_{out3} = x_1*w_5 + v_{dd} *w_6$ ….Equation 5

$t_3 = f(t_{out3})$ ….Equation 6

$t_{out4} = x_2*w_7 + v_{dd}*w_8$ ….Equation 7

$t_4 = f(t_{out4})$ ….Equation 8

$y_{out} = t_4*w_{10} + t_3*w_9$ ….Equation 9

$y = f'(y_{out})$ ….Equation 10

Where, $f' = \{1 \text{ if } t_{out1} >= 0 \text{ else } -1\}$

Table 4 displays the weight matrix for ANN-CMOS OR gate for different possible inputs

Table 4: Weight Matrix for ANN-CMOS OR gate

x_1	x_2	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
-1	-1	-1	-1	-1	-1	1	1	1	1	1	1
-1	1	-1	-1	-1	-1	1	1	1	1	1	1
1	-1	-1	-1	-1	-1	1	1	1	1	1	1
1	1	-1	-1	-1	-1	1	1	1	1	1	1

Table 5: Truth Table for ANN-CMOS OR gate

x_1	x_2	t_1	t_2	t_3	t_4	y
-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	1	1
1	-1	-1	-1	1	-1	1
1	1	-1	-1	1	1	1

of an OR gate. Table 5 displays the truth table of ANN-CMOS OR gate.

V. Discussion

This paper is an approach to design the ANN models for basic logic gates. These basic logic gates designed using ANN models are based on CMOS circuits. These ANNs are simple Self Organizing Maps (SOMs) which uses unsupervised learning. So the synaptic weighs are fixed here. Some work has already been done in this field as discussed in the introduction but the work in this paper very simple circuits are developed. Keeping the circuit simple makes the circuits easy to implement. This approach can be used to simplify complex circuits in future. This approach is slightly deviated from the overall functioning and design of CMOS gates. A similar approach has been done in another paper which is more similar to CMOS gate.

Now-a-days complex electronic circuits are simplified using VLSI design. This can be done using CMOS circuits. Lot of work has already be done on ANNs. ANNs try to model human brain using artificial neurons. These neurons take binary inputs and produce binary outputs. If the CMOS circuits can be designed using ANNs, the circuits can be further simplified by software implementation. Complex circuits are expensive. Simple circuits can be designed which are less expensive. Complex functions can be carried out by ANN models and these models can be mapped to simple circuits.

VI. Conclusion

The approach in this paper is to develop simple circuits to design basic logic gates using CMOS. The use of CMOS circuits is used in many devices now days. ANNs are also becoming popular. This approach will lead to the development of simple circuits and also the hardware implementation of the ANN models will become simple with the advancement of this technology. Approach here is to initiate the work to map simple CMOS hardware circuits with ANN models which can solve complex problems. The work can be started by simply modeling logic gates with ANN. This paper is an approach to design the basic gates applying this logic.

References

[1.] L. Fausett, “Fundamentals of Neural Networks, Architectures, Algorithms and Applications”, Pearson Education, India, 2009.

[2.] G.N. Swamy, G. Vijay Kumar, “Neural Networks”, Scitech, India, 2007.

[3.] Ashutosh Aggarwal, Rajneesh Rani, RenuDhir, “Handwritten Devanagri Character Recognition using Gradient Features”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 5, May 2012, pp. 85-90.

[4.] Sandeep, Saha, Nabarag Paul, Sayam Kumar Das, Sandip Kundu, “Optical Character Recognition using 40-point Feature Extraction and Artificial Neural Network”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 4, April 2013, ISSN 2277 128X, pp. 495-502.

[5.] Ali Borji, Mandana Hamidi, Fariborz Mahmoudi, “Robust Handwritten Character Recognition with Features Inspired by Visual Ventral Stream”, © Springer Science+Business Media, LLC. 2008, published online (31 August 2008), pp. 97-111.

[6.] Y Perwej and A Chaturvedi, “Neural Networks for Handwritten English Alphabet Recognition”, International Journal of Computer Applications, Volume 20, No. 7, pp. 1-5, 2011.

[7.] Frye R C, Rietman E A, and Wong C C, “Back-propagation learning and non idealities in analog neural network hardware,” Neural Networks, IEEE Transactions on, vol. 2, no. 1, pp. 110–117, 1991.

[8.] Jung S and Kim S S, “Hardware implementation of a real-time neural network controller with a dsp and an fpga for nonlinear systems,” Industrial Electronics, IEEE Transactions on, vol. 54, no. 1, pp. 265–271, 2007.

[9.] Hikawa H, “{FPGA} implementation of self organizing map with digital phase locked loops”, Neural Networks, vol. 18, no. 56, pp. 514 – 522, 2005, {IJCNN} 2005. Available Online: http://www.sciencedirect.com/science/article/pii/S0893608005001103

[10.] Merolla P A, Arthur J V, Alvarez-Icaza R, Cassidy A S, Sawada J, Akopyan F, Jackson B L, Imam N, Guo C, Nakamura Y, Brezzo B, Vo I, Esser S K, Appuswamy R, Taba B, Amir A, Flickner M D, Risk W P, Manohar R, and Modha D S, “A million spiking-neuron integrated circuit with a scalable communication network and interface”, Science, Vol. 345, No. 6197, pp 668–673, 2014.
Available Online: http://www.sciencemag.org/content/345/6197/668.abstract

[11.] Forssell M, “Hardware Implementation of Artificial Neural Networks”, 18-859E Information Flow in Networks, pp 1-4.
“Available:http://users.ece.cmu.edu/~pggrover/teaching/files/NeuromorphicComputing.pdf”, (Accessed : 2016)

[12.] Yellamraju S, Kumari Swati, Girolkar S, Chourasia S and Tete A D, “Design of Various Logic Gates in Neural Networks”, Annual IEEE India Conference (INDICON), 2013, Mumbai, India.

[13.] Hawas N M, Rekaby B K A, “ANN Based On Learning Rule of Neuron Activation Function Using Electronic Devices”, International Journal of Advanced Computer Technology (IJACT), Vol 4, No. 3, pp 19-22, 2015.

[14.] Kale N B, Padole V B, “Compression and Decompression of Signal Using CMOS Technology...A Review”, International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, Issue 3, pp 53-55, 2014.

[15.] Hui W, Dejang C, Wei Z, Ping Z, Yongsheng L, “Application of artificial neural networks to segmented arc heather failure prediction”, AMSE Journals, Advances in Modeling, Series B, Vol. 54-1, Issue 1, 2011, pp17-29.