Asian Journal of Information Technology 15 (5): 955-960, 2016
ISSN: 1682-3915

An Analog Low Power VLSI Implementation of Artificial Neural Network Architecture

[1]S. Arumugam and [2]N. Rajeswaran
[1]Departement of EEE, Sri Shakthi Institute of Engineering and Technology,
[2]Departement of ECE, SNS College of Technology,
Coimbatore, Tamil Nadu, India

Abstract: All the modern technologies in digital systems are slowly converting in to the analog implementation especially, for the fault tolerance and low power consumption. But, the analog implementation of parallel computation with ANN (Artificial Neural Network) in real time implementation is not an easy task in all aspects. This study mainly focuses on the implementation of Neural Network Architecture (NNA) with on chip learning in analog VLSI (Very Large Scale Integration). Back Propagation Neural network (BPN) algorithm is designed and simulated in analog domain by using tanner EDA tool.

Key words: Analog VLSI, ANN, BPN, low power, parallel computation

INTRODUCTION

The brain is also "analog". Understanding information processing in biological systems in addition to the physics of analog signals, even more efficiently signal processing in neural networks can be obtained (Douglas *et al*., 1995). Basically, ANN is made up of input layer, hidden layer and output layer as shown in Fig.1. Here, the input is given to the neurons of the first layer which in turn make a response to the next layer (hidden layer) and so on. This process is continued until a response is received by output layer of the neuron. Several classes of neural network architectures exist. If the training set includes the correct outputs (targets), then it is called as supervised learning or "learning with a teacher" else it is unsupervised learning. In this study, implementation of a neural network with supervised learning is attempted. In the present neural network, model, the input layer consists of six nodes and the output layer consists of one node. After completion of the forward process, the algorithm steps back (back propagation) to one layer before the output layer for recalculates the weights between hidden layer and output layer to minimize the error (Rajeswaran *et al*., 2013).

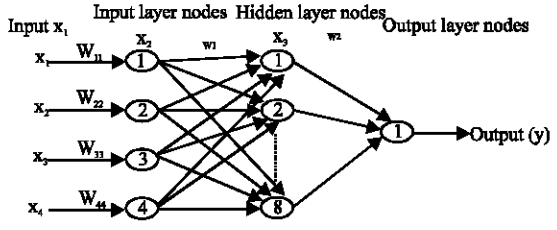

Fig. 1: Multi layer feed-forward neural network with input/output and one hidden layer

MATERIALS AND METHODS

Summary of back propagation algorithm

Calculate the net-input variables to the hidden layer units:

$$\text{net\_hidden} = \sum_{i=1}^{N} w_{ji}^{h} x_{pi} + \theta_{j}^{h}$$
$$\text{net}_{pj}^{h} = \text{Input} \times \text{wt\_hidden} + \text{bias\_hidden} \quad (1)$$

Calculate the outputs from the hidden layer:

$$\text{Op\_hiddenlayer} = \text{Ipj} = f_{j}^{h}\ (\text{net\_hidden}) \quad (2)$$

Calculate the net input values to each unit of the output layer:

$$\text{Net\_op} = (\text{op\_hidden layer} \times \text{wt\_op}) + \text{bias\_op}$$
$$\text{net}_{pk}^{o} = \sum_{j=1}^{L} w_{kj}^{o} i_{pj} + \theta_{k}^{o} \quad (3)$$

Calculate the outputs of the output layer:

Corresponding Author: S. Arumugam, Departement of EEE, Sri Shakthi Institute of Engineering and Technology, Coimbatore, Tamil Nadu, India

$$\text{Op-output layer} = O_{pk} = f_k(\text{net\_op}) \tag{4}$$

Calculate the error terms for the output units:

$$\begin{aligned}\text{Error\_op} &= (\text{net\_op})(\text{desired\_op-op\_outputlayer})\ \ (\text{net\_op})f_j^o \\ \delta_{pk}^o &= (y_{pk} - o_{pk})f_k^o(\text{net}_{pk}^o)\end{aligned} \tag{5}$$

Calculate the error terms for the hidden layer units:

$$\begin{aligned}\text{Error\_hidden} &= \int_j^n (\text{net\_hidden})(\text{error\_op} \times \text{wt\_op}) \\ & f_j^h(\text{net\_hidden})\sum\nolimits_k \delta_{pk}^o w_{kj}^o\end{aligned} \tag{6}$$

Update the weights on the output layer:

$$\begin{aligned}&\text{Wt\_op} = \text{wt\_op} + \text{learn\_para} \times (\text{error\_op} \times \\ &\text{op\_hidden layer}) + \text{momentum} \times \text{learn\_para} \times \\ &\text{error\_op} \times \text{op\_hidden layer} \\ &w_{kj}^o(t+1) = w_{kj}^o(t) + \eta\delta_{kj}^o i_{pk} + \infty\delta_{pk}^o i_{pj}\end{aligned} \tag{7}$$

Update weights on the hidden layer:

$$\begin{aligned}&\text{Wt\_hidden} = \text{wt\_hidden} + \text{learn\_para} \times \\ &\text{error\_hidden} \times \text{input} + \text{momentum} \times \\ &\text{learn\_para} \times \text{error\_hidden} \times \text{input} \\ &w_{ji}^o(t+1) = w_{ji}^h(t) + \eta\delta_{pj}^h x_i + \infty\Delta p w_{ji}^h(t-1) = \\ &w_{ji}^o(t) + \eta\delta_{pj}^h x_i + \infty\eta\delta_{pj}^h x_i\end{aligned} \tag{8}$$

$$\text{Error term} = \frac{1}{2}(\text{error} - \text{op})^2 = \frac{1}{2} = \sum_{k=1}^{M}\delta_{pk}^2 \tag{9}$$

Analog VLSI implementation of BPN: To implement the mathematical neural network equations, we need to implement basic operations as summation, subtraction and multiplication (Rajeswaran *et al*., 2012). If the implementation is a digital system then one has to build an adder, sub tractor and a multiplier. However, as an analog system is to be built, voltage and current are to be considered. Using current signal representation summation (and subtraction) can be easily achieved by just physically connecting signals together by using Kirchhoffs current law. By using the CMOS transistor in weak inversion we obtain an operation range for current signals up to nano Ampheres (nA) theoretically (Linares-Barranco, 2003). Voltage signals have the advantage that they may be distributed to many high-ohmic nodes (as gates on a CMOS transistor). Voltage signals should be applied when a neuron output is assigned too many synapses (Chicca *et al*., 2003).

Multiplication can be performed by various circuits by considering the linear range, offset problems, size and type of input/output signals. Derivation is another non trivial mathematical operation and is to be implemented in analog system. Some circuits need differential input/output representation and other single representation. With one signal connection between each circuit a large amount of routing space is saved. However, a reference signal has to be routed to those circuits having differential signal input. In addition, some of these circuits may have different demands on the reference signal value which means that several reference signals have to be applied on the chip (Heittmann, 2000; Bartolozzi and Indiveri, 2007). With two connections between each circuit, each signal can be split into a positive and a negative component. When routing this type of representation, lot of extra space is required.

Another solution would be to use a combination of both single and differential representation. Linking two or more circuits could use one bidirectional current signal. And for those circuits which need differential inputs a small converter could be applied to convert the one bidirectional signal into two unidirectional signals. There exist several multiplier circuits operating in weak inversion. A major problem for these multipliers is the limited linear range (Wang *et al*., 2006; Rao *et al*., 2009; Tu and Van, 2009).

To overcome this problem, in this paper modified Baugh-Wooley multiplier is applied for the design of BPN. Often they do not satisfy the requirements of accuracy in certain calculation because of transistor mismatch and temperature variations.

It is very important to choose the right multiplier to test various circuits to find multiplier which gives best accuracy and fits with the signal representation (Kuang and Wang, 2007; Namba and Ito, 2005; Sjalander *et al*., 2005; Zouyed and Khoualdia, 2007). Implementation of the neural architecture using analog Blocks is presented in Fig. 2. It contains the multiplier block (Baugh-Wooley multiplier) for processing hidden and input layer with weight values. The input layers are V1 and 2.

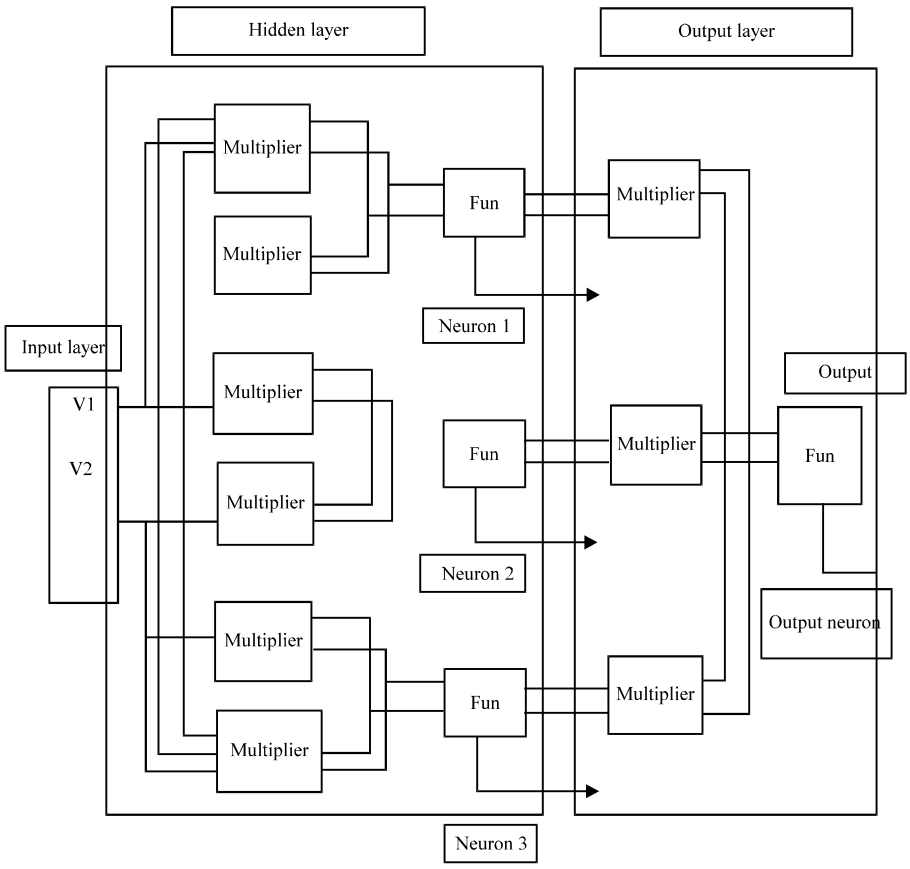

Fig. 2: Implementation of the neural architecture using analog blocks

RESULTS AND DISCUSSION

Tanner t-spice circuit simulator puts in control of simulation jobs with an easy to use graphical interface and a faster intuitive design environment (Hommounda *et al*., 2008; Varghese *et al*., 2008). Figure 3 the inputs to the neuron are multiplied by the weight matrix and the resultant output is summed up and is passed through a Neuron Activation Function (NAF).

The output of the activation function then passes to the next layer for further processing. Blocks to be used are multiplier block, adders, NAF block with derivative. The designed activation function is tan sigmoid. The proposed design is actually a differential amplifier modified for differential output. Same circuit is capable of producing the output of activation function and its differentiation. This block is named as tan in the final schematics for neural architecture for compression and achieving a tangential output. Differential amplifier when designed to work in the sub-threshold region acts as a neuron activation function. The Baugh-Wooley multiplier is used to achieve low power and less area utilization on chip. Schematic of the design is used for the tan sigmoid function generator with modification for the differential output. Input applied at V2 is 0101 and at V3 is 0011. The NAF function can be derived from the same differential pair configuration. The structure has to be modified for the differential output (Fig. 4).

One important complication in the development of the efficient multiplier implementations is the multiplication of two's complement signed numbers. The modified Baugh-Wooley two's complement signed multiplier is the best known algorithm for signed multiplication because it maximizes the regularity of the multiplier logic and allows all the partial products to have positive sign bits. It will helpful for the implementation of ANN in analog. The baugh-wooley multiplier is implemented with the help of two blocks known as white cell and grey cell. The white cell consists of a AND gate to which two inputs are fed from a and b respectively. The output of the AND gate is fed to a full adder circuit along with the sum bit and carry bit from the previous stages.

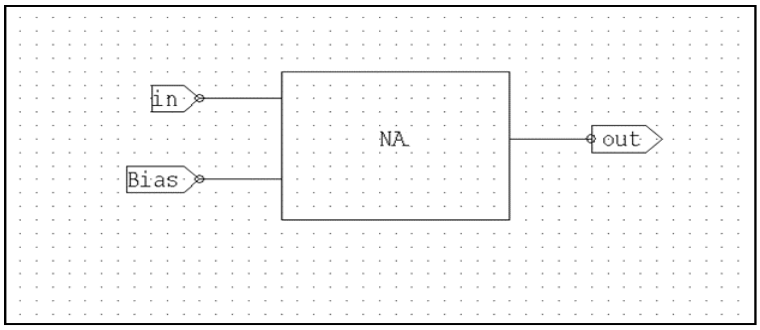

Fig. 3: Block diagram of NAF

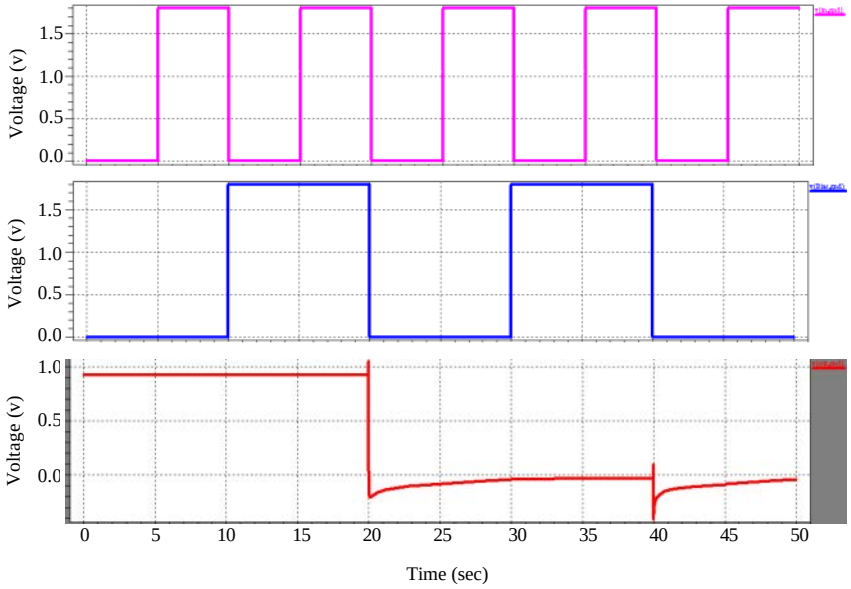

Fig. 4: Simulation output of NAF

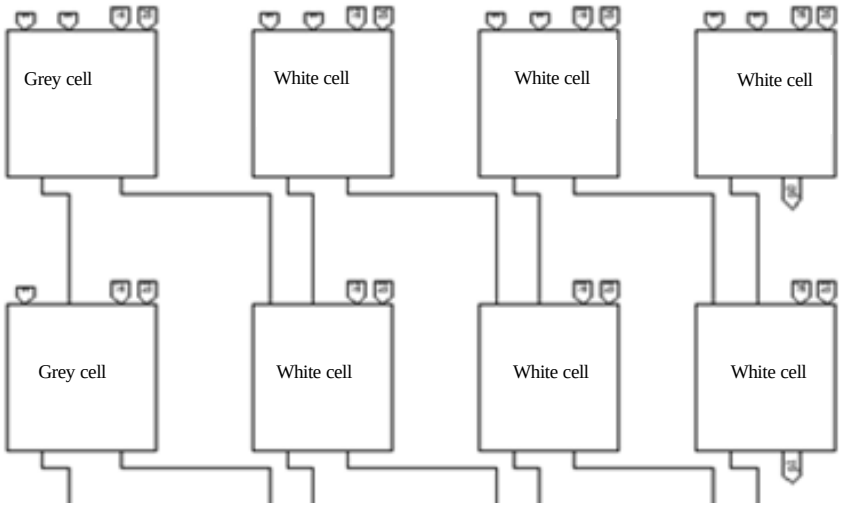

Fig. 5: Baugh-Wooley multiplier design

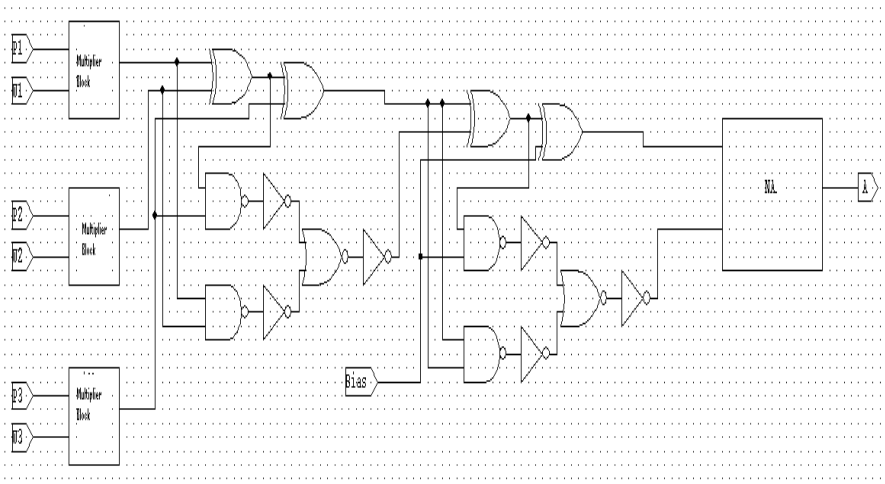

Fig. 6: Schematic view of back propagation neural network implementation

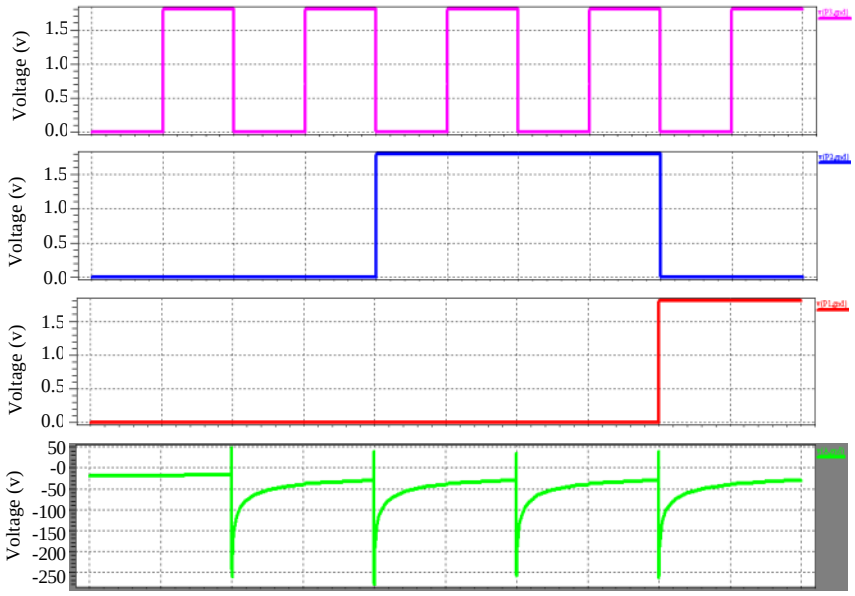

Fig. 7: Simulation output of BPN

Table 1: Design parameters

Parameters	Value assigned
P1 input	0000000011111111
P2 input	00001100011111100
P3 input	0110000011111111
W1 weight	1100000011111110
W2 weight	0001110010011111
W3 weight	0001100011111110
Bias	1
DC	1.8 V
GND	-
V2	0000110011111111
V3	0000110011111100
V4	0110000000111111
V5	0001100011111110
V6	0001100011111001
V7	0001100011111000
V8	0000011011111100

A grey cell is similar to the white cell with the exception that a NAND gate is used in place of the AND gate. The Baugh-Wooley multiplier that is designed with white and grey cell is shown in Fig. 5. The schematic view of BPN is shown in Fig. 6 and the design parameters are tabulated in Table 1.

The proposed design output is shown in Fig. 7. The obtained results are clearly proved that the proposed analog VLSI implementation of artificial neural network occupied the less area and computing the functions only involves 7 transistors. Using analog CMOS, low power consumption is achieved, especially when operating the CMOS transistor in the subthreshold region.

CONCLUSION

Neural network architecture implemented in analog VLSI provides a high degree of fault tolerance as it may not be critical if a few transistors do not function. This is not true with digital systems. The analog circuit computing the functions only involves 7 transistors. Such a high computational density is impossible for digital systems to achieve. Using analog CMOS, low power consumption is achieved, especially when operating the CMOS transistor in the sub threshold region. However, analog systems are not robust to noise but that is not a requirement for neural networks. Besides, it has been shown that noise assists neural networks to learn. The neural architecture can be extended for the application of image compression, speech recognition and testing and fault diagnosis of the system.

REFERENCES

Bartolozzi, C. and G. Indiveri, 2007. Synaptic dynamics in analog VLSI. Neural Comput., 19: 2581-2603.

Chicca, E., D. Badoni, V. Dante, M.D. Andreagiovanni and G. Salina *et al*., 2003. A VLSI recurrent network of integrate-and-fire neurons connected by plastic synapses with long-term memory. IEEE Trans. Neural Networks, 14: 1297-1307.

Douglas, R., M. Mahowald and C. Mead, 1995. Neuromorphic analogue VLSI. Annu. Rev. Neurosci., 18: 255-281.

Hammouda, H.B., M. Mhiri, Z. Gafsi and K. Besbes, 2008. Neural-based models of semiconductor devices for SPICE simulator. Am. J. Applied Sci., 5: 385-391.

Heittmann, A., 2000. An analog VLSI pulsed neural network for image segmentation using adaptive connection weights. Dresden University of Technology, Department of Electrical Engineering and Information Technology, Dresden, Germany.

Kuang, S.R. and J.P. Wang, 2007. Design of power efficient pipelined truncated multipliers with various output precision. IET Comput. Digital Tech., 1: 129-136.

Linares-Barranco, B., T. Serrano-Gotarredona and R. Serrano-Gotarredona, 2003. Compact low-power calibration mini-DACs for neural arrays with programmable weights. IEEE Trans. Neural Networks, 14: 1207-1216.

Namba, K. and H. Ito, 2005. Design of defect tolerant Wallace multiplier. Proceedings of the 11th Pacific Rim International Symposium on Dependable Computing, December 12-14, 2005, Hunan, China, pp: 300-304.

Rajeswaran, N., T. Madhu and M. Suryakalavathi, 2013. VHDL synthesizable hardware architecture design of back propagation neural networks. Proceedings of the IEEE Conference on Information and Communication Technologies, April 11-12, 2013, JeJu Island, South Korea, pp: 445-450.

Rajeswaran, N., T. Madhu and M.S. Kalavathi, 2012. Fault diagnosis and testing of induction machine using Back Propagation Neural Network. Proceedings of the IEEE International Power Modulator and High Voltage Conference, June 3-7, 2012, San Diego, CA., USA., pp: 492-495.

Rao, P.V., C.P. Raj and S. Ravi, 2009. VLSI design and analysis of multipliers for low power. Proceedings of the 5th International Conference on Intelligent Information Hiding and Multimedia Signal Processing, September 12-14, 2009, Kyoto, Japan, pp: 1354-1357.

Sjalander, M., M. Drazdziulis, P. Larsson-Edefors and H. Eriksson, 2005. A low-leakage twin-precision multiplier using reconfigurable power gating. Proceedings of the IEEE International Symposium on Circuits and Systems, Volume 2, May 23-26, 2005, Kobe, Japan, pp: 1654-1657.

Tu, J.H. and L.D. Van, 2009. Power-efficient pipelined reconfigurable fixed-width Baugh-Wooley multipliers. IEEE Trans. Comput., 58: 1346-1355.

Varghese, A., C.N. Marimuthu and P. Thangaraj, 2008. Low power high performance multiplier. Int. J. Soft Comput., 3: 412-420.

Wang, A., B. Calhoun and A.P. Chandrakasan, 2006. Sub-Threshold Design for Ultra Low-Power Systems. Springer, New York, USA., ISBN-13: 978-0387335155, Pages: 209.

Zouyed, A. and A. Khoualdia, 2007. Contribution to definition of a structured design methodology of mixed circuit. Asia J. Inform. Technol., 6: 1218-1223.