Neural Network Implementation Using FPGA: Issues and Application

A. Muthuramalingam, S. Himavathi, E. Srinivasan

International Science Index, Electrical and Computer Engineering Vol:2, No:12, 2008 waset.org/Publication/15106

***Abstract*—**Hardware realization of a Neural Network (NN), to a large extent depends on the efficient implementation of a single neuron. FPGA-based reconfigurable computing architectures are suitable for hardware implementation of neural networks. FPGA realization of ANNs with a large number of neurons is still a challenging task. This paper discusses the issues involved in implementation of a multi-input neuron with linear/nonlinear excitation functions using FPGA. Implementation method with resource/speed tradeoff is proposed to handle signed decimal numbers. The VHDL coding developed is tested using Xilinx XC V50hq240 Chip. To improve the speed of operation a lookup table method is used. The problems involved in using a lookup table (LUT) for a nonlinear function is discussed. The percentage saving in resource and the improvement in speed with an LUT for a neuron is reported. An attempt is also made to derive a generalized formula for a multi-input neuron that facilitates to estimate approximately the total resource requirement and speed achievable for a given multilayer neural network. This facilitates the designer to choose the FPGA capacity for a given application. Using the proposed method of implementation a neural network based application, namely, a Space vector modulator for a vector-controlled drive is presented

***Keywords*—** FPGA Implementation, Multi-input Neuron, Neural Network, NN based Space Vector Modulator

I. Introduction

THE aspiration to build intelligent systems complemented with the advances in high speed computing has proved through simulation the capability of Artificial Neural Networks (ANN) to map, model and classify nonlinear systems. The learning capability of the network has opened its application to various fields of engineering, science, economics etc. [1-4]. Real time applications are possible only if low cost high-speed neural computation is made realizable. Towards this goal numerous works on implementation of Neural Networks (NN) have been proposed [5].

Neural networks can be implemented using analog or digital systems. The digital implementation is more popular as it has the advantage of higher accuracy, better repeatability, lower noise sensitivity, better testability, higher flexibility, and compatibility with other types of preprocessors. The digital NN hardware implementations are further classified as (i) FPGA-based implementations (ii) DSP-based implementations (iii) ASIC-based implementations [6-7]. DSP based implementation is sequential and hence does not preserve the parallel architecture of the neurons in a layer. ASIC implementations do not offer re-configurablity by the user. FPGA is a suitable hardware for neural network implementation as it preserves the parallel architecture of the neurons in a layer and offers flexibility in reconfiguration.

Parallelism, modularity and dynamic adaptation are three computational characteristics typically associated with ANNs. FPGA-based reconfigurable computing architectures are well suited to implement ANNs as one can exploit concurrency and rapidly reconfigure to adapt the weights and topologies of an ANN. FPGA realization of ANNs with a large number of neurons is still a challenging task because ANN algorithms are "multiplication-rich" and it is relatively expensive to implement. Various works reported in this area includes new multiplication algorithms for NN [8], NNs with some constraints to achieve higher speed of operation at lower cost [9] and multi-chip realization [10].

In this paper, issues related to the FPGA implementation of a multi-input neuron are discussed. Both the linear and nonlinear excitation functions are considered. The related issues such as resource requirement, speed of execution and accuracy are addressed through computational architectural solutions using FPGA. Section II presents the design and implementation of computational blocks of a multi-input neuron. Also identifies the various conflicting requirements involved in its implementation and proposes a solution other than tradeoff. Section III and IV deal with a real time application of neural networks and their FPGA implementation. Section V concludes the paper.

II. Computational Blocks for a Neuron

The basic structure of a neuron with '*n*' inputs is shown in Fig. 1. The function of a neuron is described by the following equations.

$$y = f(x), \tag{1}$$

Manuscript received in April, 2007. The research is supported in part by the grants from All India Council for Technical Education (AICTE), a statutory body of Government of India. File Number: ***8020/RID/TAPTEC-32/2001-02.***

A.Muthuramalingam is an Assistant Professor in the Electrical and Electronics Engineering Department of Pondicherry Engineering College, Puducherry, India (email: amrlingam@hotmail.com).

S.Himavathi is an Assistant Professor in the Electrical and Electronics Engineering Department of Pondicherry Engineering College, Puducherry, India (phone: 91-413-2655281; fax: 91-413-2655101; e-mail: hima_pondy@yahoo.co.in).

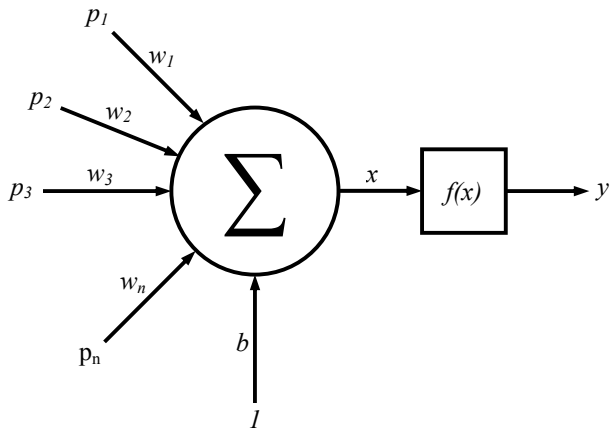
E.Srinivasan is an Assistant Professor in the Electronics and Communication Engineering Department of Pondicherry Engineering College, Puducherry, India (email: esrinivasan2004@yahoo.co.in).

International Science Index, Electrical and Computer Engineering Vol:2, No:12, 2008 waset.org/Publication/15106

and $$x = \sum_{i=1}^{n} p_i w_i + b$$

where p_i be the i^{th} inputs of the system w_i is the weight in the i^{th} connection and 'b' is the bias.

Fig. 1 Structure of a Neuron



The function *f(x)* is the excitation function used in the neuron. Generally Linear, Log-sigmoid and Tan-sigmoid excitation functions are used. They are defined as

(i) Linear

$$f(x) = x \tag{2}$$

(ii) Log-sigmoid function

$$f(x) = \frac{1}{1+e^{-x}} \tag{3}$$

(iii) Tan-sigmoid function

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4}$$

To realize a function of a single neuron, the above expression (1) and (2) or (3) or (4) are to be computed. So, the different computational blocks are adder, multiplier and complex evaluator of the nonlinear excitation function.

A. Implementation Issues of Computational Blocks

The inputs to the neural network are generally normalized to lie within the range of –1 to +1. Hence the signed floating-point computations are required. The implementation of signed floating-point multipliers and computation of nonlinear excitation function is complex and requires large resource. The major issues in realization of the computational blocks using FPGA are

i. Parallel/Sequential implementation
ii. Bit precision
iii. Use of look up table for nonlinear function

Parallel computations require larger resource and are therefore costly. To reduce cost, the computations are carried out sequentially which in turn reduces the speed of computation. Selecting bit precision is another important choice when implementing ANNs on FPGAs. A higher bit precision means fewer quantization errors in the final implementations, while a lower precision leads to simpler designs, greater speed and reductions in area requirements and power consumption. Lookup tables improve speed of operation, but higher precision demands larger memory. For a given application the speed and minimum accuracy is dictated by the system under consideration. Hence the solution is to minimize the cost for a given speed of operation and required accuracy.

B. FPGA Implementation of a Multi-Input Neuron

The structure of a neuron is split into various sub blocks and these blocks are implemented individually first and then they are integrated to form the entire neuron. The hierarchy of the different blocks is as shown in Fig. 2. The two major blocks are SIGMA block and LOGSIG block. As this paper aims to build multi-layer neural networks with minimum resource 9-bit word length with one sign bit and 8 data bits has been chosen for this implementation. In typical application hardware matching the precision to the computational accuracy of the neural network can further optimize resource.

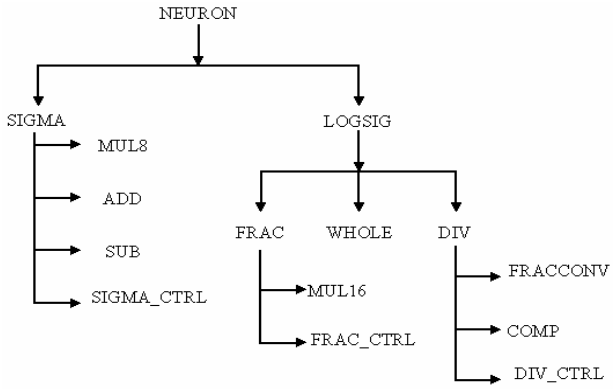


Fig. 2 Computational Blocks of a Neuron

The result of the excitation function *f(x)* is obtained as a 9-bit signed number (1-bit for sign and 8-bit for data). The precision and word length is chosen so that a single neuron can handle a maximum of 16 inputs without the problem of overflow. However modification to the word length could accommodate higher number of inputs to a neuron. Generally in real time applications (Such as vector control of motor drives) of neural networks the number of inputs to a neuron rarely exceeds 16. The complete structure of the neuron in FPGA is shown in Fig. 3.

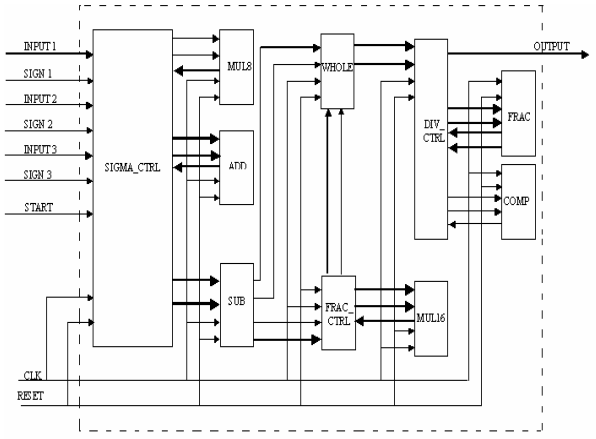


Fig. 3 Complete Structure of a Neuron Implementation using Computation Method in FPGA

Sigma Block: This bock computes the value for

$$x = \sum_{i=1}^{n} p_i w_i + b$$

The basic functions of the block is multiplication, addition, subtraction and a control block to coordinate and sequence the flow of data to the various blocks The input data $p_1, p_2 \ldots p_n$ are signed numbers in the range [-1, +1]. This is represented using a signed 9-bit number (1-bit for sign and 8-bit for data). The weights of the network are represented with 17 bits (one bit for sign, 8-bit for whole part and 8-bit for fraction part) assuming that the weights lie between –256 to +256. The bias is represented by a signed 25-bit number (one bit for sign, 8-bit for whole part and 16-bit for fractional part). The product of the weights and inputs are stored as 24-bit number along with a sign bit. The output $x = \sum_{i=1}^{n} p_i w_i + b$ is obtained with 29-bits (1-bit for sign + 12-bit for whole part + 16-bit for fractional part). The different sub blocks are as follows.

MUL8: Performs 8-bit multiplication.
ADD: Performs 24-bit addition.
SUB: Performs 24-bit subtraction.

SIGMA_CTRL: It is finite state machine, which controls the operation of ADD, MUL8, and SUB blocks.

LOGSIG Block: The computation of $f(x) = 1/(1+e^{-x})$ is done in this block. The logic used to determine e^{-x} is to obtain 2^{-x} as detailed in [11]. Then e^{-x} is obtained using the relation $e^{-x} = 1.4426 \times 2^{-x}$. The determination of $f(x)$ is done as follows. The value of x is split into $x_1 + x_2$ where x_1 is the whole number and x_2 is the fractional part. The value 2^{-x2} is obtained using the FRAC block; the value 2^{-x} is then obtained by WHOLE block by shifting 2^{-x2} right x_1 times and then converts 2^{-x} to e^{-x}. The DIV block obtains the value of the function $f(x) = 1/(1+e^{-x})$.

C. Hardware Implementation

The architecture of the complete neuron is shown in Fig. 3. The same is implemented using Xilinx XC V50hq240 Chip. The results obtained for a three input neuron with different excitation functions is shown in Table I. From Table I the LOGSIG block requires huge resource for implementation and time for execution. To reduce the execution time and the cost of implementation, a lookup table replaces the LOGSIG block.

TABLE I
RESOURCE AND TIMING OF THREE INPUT NEURON WITH DIFFERENT EXCITATION FUNCTIONS

Excitation Function	Resource required in Slices			Timing Required in cycles		
	SIGMA Block	LOGSIG Block	Total	SIGMA Block	LOGSIG Block	Total
Linear	258	-	258	23	-	23
Log sigmoid	258	705	963	23	98	121
Tan sigmoid	258	811	1069	23	126	149

The architecture of a 3-input neuron with LUT is shown in Fig. 4. The LUT is implemented using the inbuilt RAM available in FPGA IC. The use of LUTs reduces the resource requirement and improves the speed. Also the implementation of LUT needs no external RAM since the inbuilt memory is sufficient to implement the excitation function. As the excitation function is highly nonlinear a general procedure adopted to obtain a LUT of minimum size for a given resolution is detailed [12].

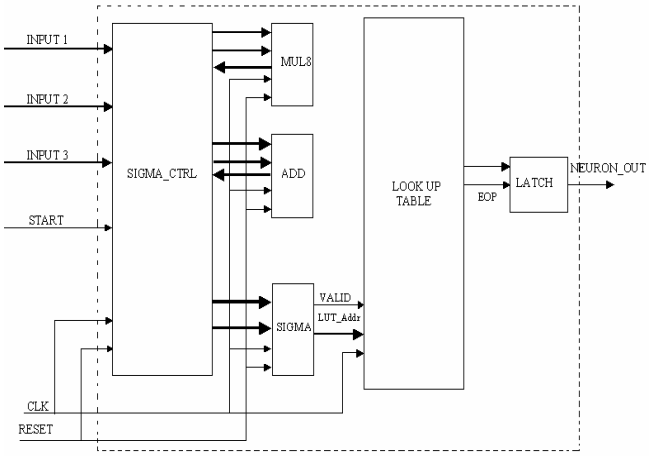


Fig. 4 Complete Structure of a Neuron using LUT in FPGA

A single neuron with three inputs and various excitation functions was implemented with and without LUT in using 'XCV400hq240'. The results are summarized in Table II.

International Science Index, Electrical and Computer Engineering Vol:2, No:12, 2008 waset.org/Publication/15106

TABLE II
RESOURCE AND TIMING REQUIREMENT OF A NEURON WITH AND WITHOUT LUT

Excitation Function	Resource required in Slices			Timing Required in cycles		
	LUT	COMPUTATION	% Saving	LUT	COMPUTATION	% Saving
Log Sigmoid	281	953	70.5	25	121	79.33
Tan Sigmoid	282	1096	74.27	25	149	83.22

It was observed that there is 70 to 74% reduction in resource requirement and 79 to 83% improvement in speed is obtained by using a LUT.

Estimation of Resource and Time for a NN: The single neuron with varying number of inputs and excitation function is implemented in hardware. As the number of input to the neuron increases, the resource and timing also increases. From the results generalized formulas are derived in this section to approximately compute the total resource and timing requirement for any given NN.

Let n be the number of layers and let S^0, S^1, S^2, $S^3 \ldots S^n$ be the number of neurons in each layer. If the i^{th} layer is a linear function $a_{1i}=1$, $a_{2i}=0$, $a_{3i}=0$ for a log-sigmoid function $a_{1i}=0$, $a_{2i}=1$, $a_{3i}=0$ and for a tan sigmoid function $a_{1i}=0$, $a_{2i}=0$, $a_{3i}=1$. Then the total number of slices (S), and total number of clock cycles (T) for a neural network using computation method and LUT method is given below.

For Computation Method:

$$S \approx \sum_{i=1}^{n} a_{1i} S^i (250 + 6S^{i-1}) + \sum_{i=1}^{n} a_{2i} S^i (950 + 6S^{i-1}) + \sum_{i=1}^{n} a_{3i} S^i (1050 + 6S^{i-1}) \quad (5)$$

$$T \approx 5 \times \{\sum_{i=1}^{n} a_{1i} (1.6 + S^{i-1}) + \sum_{i=1}^{n} a_{2i} (21.2 + S^{i-1}) + \sum_{i=1}^{n} a_{1i} (26.8 + S^{i-1})\} \quad (6)$$

For LUT Method:

$$S \approx \sum_{i=1}^{n} S^i (255 + 6S^{i-1}) \quad (7)$$

$$T \approx 10 + \{\sum_{i=1}^{n} a_{1i} (5 + 5S^{i-1} + 3S^i) + \sum_{i=1}^{n} a_{2i} (7 + 5S^{i-1} + 3S^i) + \sum_{i=1}^{n} a_{3i} (7 + 5S^{i-1} + 3S^i)\} \quad (8)$$

Choice of Optimal Architecture: The choice of architecture of NN is more an art than a science. The results obtained will aid the designer to choose an optimal architecture of NN for the given application. The architecture of NN comprises of determining the following.

i. Number of input neurons
ii. Number of layers
iii. Number of neurons in each layer
iv. Type of activation function for each layer
v. Number of output neurons

i and v are dictated by the problem. The type of excitation for the output layer depends on the range of output. The maximum sampling interval t_s for a given system can be obtained from the time constant of the system. From t_s the maximum number of layers (ii) can be determined using any one of these equations (5)&(6) or (7)&(8). Let the number of layers be 'L'. Using more number of layers will help the network learn faster. Place 2 neurons in each layer and train the network and test for the performance. Increase the number of neurons in a layer and train the network again till satisfactory performance. This systematic procedure helps to obtain an optimal NN architecture.

III. SVM DRIVEN VOLTAGE SOURCE INVERTER

Space Vector Modulation (SVM) is an optimum Pulse Width Modulation (PWM) technique for an inverter used in a variable frequency drive applications. It is computationally rigorous and requires high computation time and hence limits the inverter switching frequency. Increase in switching frequency can be achieved using Neural Network (NN) based SVM. This section discusses a neural network based SVM technique for a Voltage Source Inverter (VSI). The three phase two level inverter with an active load is shown in Fig. 5.

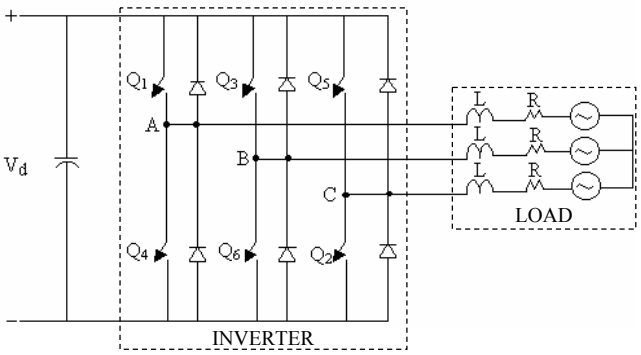


Fig. 5 Three Phase Voltage Source Inverter

Its switching operation is controlled using Space Vector Modulation (SVM). The SVM is characterized by eight switch states S_i = (SW_a, SW_b, SW_c), i= 0, 1,…., 7. Where, SW_a represents the switching status of inverter Leg-A. It is "1", when switch Q_1 is ON & Q_4 is OFF and ZERO, when switch Q_1 is OFF & Q_4 is ON. Similarly SW_b & SW_c is defined for inverter Leg-B and Leg-C. The output voltages of the inverter are controlled by these eight switching states. Let the inverter voltage vectors $V_0(000)$, …, …, $V_7(000)$, correspond to the eight switching states[13]. These vectors form the voltage vector space as shown in the Fig. 6. The three-phase reference voltage decides the inverter switching and is represented as space vector V^* with the magnitude V^* and phase angle θ^* as shown in the Fig. 6.

In a sampling/switching interval, the inverter output voltage vector V is expressed in terms of space vectors and switching on time.

$$V = \frac{t_0}{T_s} V_0 + \frac{t_1}{T_s} V_1 + \cdots + \frac{t_7}{T_s} V_7 \quad (9)$$

International Science Index, Electrical and Computer Engineering Vol:2, No:12, 2008 waset.org/Publication/15106

International Science Index, Electrical and Computer Engineering Vol:2, No:12, 2008 waset.org/Publication/15106

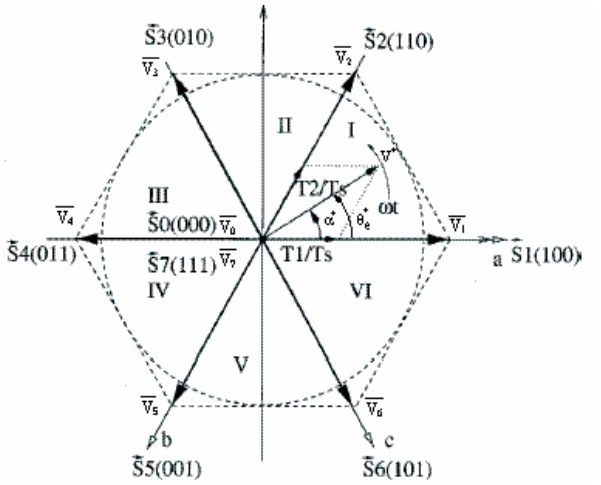


Fig. 6 Voltage Vector Space

where t_0, t_1,…t_7 are the turn on time of the vectors V_0, V_1,…..V_7 respectively and T_s is the sampling/switching time period. From the above equation the vector V^* can be decomposed into two nearest adjacent vectors (V_a , V_b) and zero vectors in an arbitrary sector. The equations of the effective time of the inverter switching states is given as

$$t_a = 2.K.V^* \sin\left(\pi/3 - \alpha^*\right) \tag{10}$$

$$t_b = 2.K.V^* \sin\alpha^* \tag{11}$$

$$t_0 = \left(T_s/2\right) - \left(t_a + t_b\right) \tag{12}$$

where

V^* - Magnitude of command or reference voltage vector

t_a - Time period of switching vector ($V_a = V_1$) that lags V^*

t_b - Time period of switching vector($V_b = V_2$) that leads V^*

t_c - Time period of switching the zero voltage vector

$T_s = (1/f_s)$ - Sampling/Switching time period

α^* - Angle of V^* in a 60° sector

f_s - Switching frequency

V_d - DC link voltage and $K = \left(\sqrt{3}T_s/4V_d\right)$

The switching time (t_a, t_b, and t_c) need to be distributed such that symmetrical PWM pulses are produced. To produce such pulses, the instant of switching on for each phase and each sector is calculated. The generalized equation for turn on time ($T_{A\text{-}ON}$), turn off time ($T_{A\text{-}OFF}$) and pulse width function $g_a(\alpha^*)$ are given below and shown for Phase A. For phases B and C, the switching instants are same but phase shifted appropriately by 120°.

$$T_{A-ON} = \left(T_s/4\right) + V^* \cdot T_s \cdot g_a\left(\alpha^*\right) \tag{13}$$

$$g_a(\alpha^*) = \begin{cases} \frac{\sqrt{3}}{4 \cdot V_d}\left[-\sin(\pi/3-\alpha^*) - \sin\alpha^*\right], S = 1,6 \\ \frac{\sqrt{3}}{4 \cdot V_d}\left[-\sin(\pi/3-\alpha^*) + \sin\alpha^*\right], S = 2 \\ \frac{\sqrt{3}}{4 \cdot V_d}\left[+\sin(\pi/3-\alpha^*) + \sin\alpha^*\right], S = 3,4 \\ \frac{\sqrt{3}}{4 \cdot V_d}\left[+\sin(\pi/3-\alpha^*) - \sin\alpha^*\right], S = 5 \end{cases} \tag{14}$$

$$T_{A\_OFF} = T_s - T_{A-ON} \tag{15}$$

IV. Neural Based SVM

The inverter output voltage is controlled by a reference vector V^*, which used to compute the switching function such as turn on time ($T_{A\text{-}ON}$), turn off time ($T_{A\text{-}OFF}$) and pulse width function $g_a(\alpha^*)$. The equation (14) is nonlinear and complex, hence requires high computation time. This limits the sampling frequency, switching frequency and performance of the inverter. To increase switching frequency, Multilayer Feed Forward NN is proposed to reduce the time of evaluation the pulse width function $g_a(\alpha^*)$and increase the inverter switching frequency.

SVM technique is implemented using multilayer NN. The input to the neural network is the phase angle (θ^*) of the reference voltage vector, the outputs are the turn-on pulse width functions $g_a(\alpha^*)$, $g_b(\alpha^*)$, $g_c(\alpha^*)$ for the phases A, B, and C. Using the procedure proposed in this paper, the NN architecture for this application is identified as 1-6-6-6-3 structure and shown in Fig. 7.

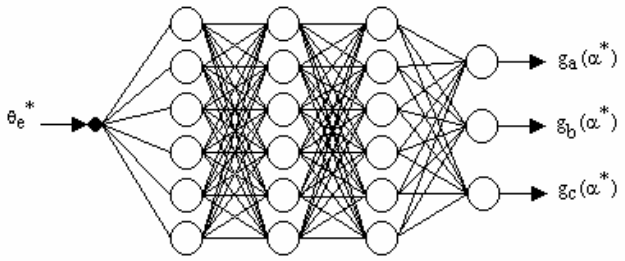


Fig. 7 Proposed NN Architecture for SVM (1-6-6-6-3)

A Performance of NN Based SVM

The neural network shown above is trained using back propagation algorithm with 360 input –output target pairs. The intermediate layer use log-sigmoid activation function and the output layer uses linear activation function. The choice of activation function is based on the non-linearity and output range of the system under consideration. The mean square error obtained for the proposed networks after one-lakh epochs is 3.48e-6. The performance of the inverter with proposed architectures is evaluated using MATLAB-Tools. The schematic of the ANN based space vector modulated voltage source inverter is shown in the Fig. 8. The inverter and load parameters are given in Table III.

International Science Index, Electrical and Computer Engineering Vol:2, No:12, 2008 waset.org/Publication/15106

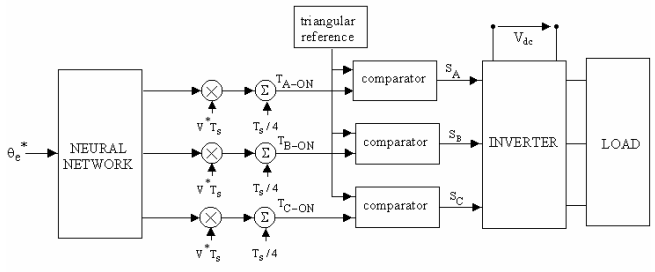


Fig. 8 Schematic of ANN based SVM Inverter

TABLE III
PARAMETERS OF THE LOAD AND INVERTER DRIVEN BY NN BASED SVM

DC-link voltage:	300 V
Frequency:	50 Hz
Load resistance/phase:	0.817 Ω
Load inductance/phase:	2.38 mH
Back emf/phase:	100 V

The load is an active load with a back emf of 100V/phase. Total Harmonic Distortion (THD) is chosen as the index to evaluate the inverter performance. The low THD/Index indicates the higher inverter performance. To make a general purpose NN based SVM, the proposed NN architectures is trained independent of switching frequency. The inverter is operated NN based SVM for typical switching frequencies; THD of line current is given in Table IV. The results demonstrate that the neural network based SVM, which is trained independent of frequency, operates well for various switching frequencies of the inverter. The inverter performance improves with switching frequencies. Higher switching frequency is made possible with proposed SVM.

TABLE IV
INVERTER PERFORMANCE WITH NN BASED SVM

Switching Frequency	Line Current (i_a) THD
20 kHz	0.1340%
10 kHz	0.2673%
2 kHz	1.347%

B FPGA Implementation of NN based SVM.

NN based SVM is implemented in FPGA using LUT described in this paper. The NN with 8-bit precision has been implemented using Xilinx 6.1i, simulated with ModelSim XE II 5.7c, and downloaded and tested in the IC 'XCV400hq240' using Mechatronics test equipment-model: MXUK-SMD-001. The resource requirement in terms of slices is found to be 6132. The approximate total number obtained using the proposed generalized formula is 5931. The approximate and actual slices for this application validate the proposed/derived formula. The outputs of the network are tested with the outputs simulated using the same precision and shown in Fig. 9.

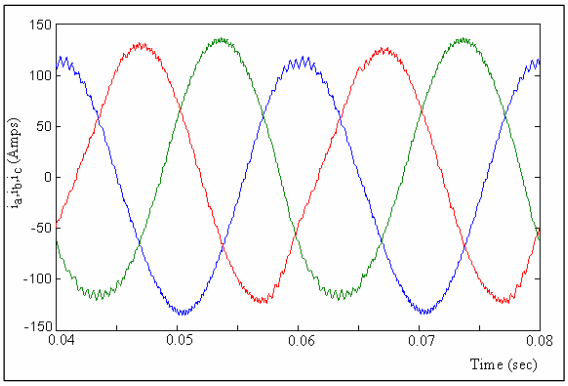


Fig. 9 Line currents of the inverter with the ANN having 8-bit precision for f_s=2kHz

The effect of the bit precision is also demonstrated with this application. Bit precision is used to trade-off the capabilities of the realized ANNs against the implementation cost. The minimum precision is found through simulation before implementation [13]. The MSE of the network with different input–output (I/O) bit precisions are tabulated in Table V. The MSE is the sum of squared error between the actual targets of the network and the outputs obtained practically due to bit truncation for the given inputs divided by the total number of patterns. From the Table V, it is seen that as the bit precision is increased, the mean square error is reduced.

TABLE V
MSE ACHIEVED WITH VARIOUS BIT PRECISIONS

Bit Precision	MSE
8-bit	78.23 e-2
10-bit	4.89 e-2
12-bit	2.50 e-3
16-bit	3.3312 e-5

In order to study the effect of the bit truncation and consequently the MSE, on the inverter performance, the THD of the inverter line currents with the ANN implemented with different bit precisions is determined and is tabulated in Table VI.

TABLE VI
THD OF THE LINE CURRENTS WITH DIFFERENT BIT PRECISIONS

Bit Precision	% THD for f_s=2kHz			%THD for f_s=20kHz		
	i_a	i_b	i_c	i_a	i_b	i_c
8	9.634	6.946	7.797	8.607	5.88	5.821
10	1.549	1.867	2.093	0.8038	0.8547	1.553
12	1.363	1.383	1.406	0.3129	0.199	0.441
16	1.347	1.348	1.348	0.1336	0.1312	0.1312
32	1.347	1.347	1.348	0.1296	0.13	0.1299

International Science Index, Electrical and Computer Engineering Vol:2, No:12, 2008 waset.org/Publication/15106

As the network is independent of switching frequency, the performance of the inverter is studied with the switching frequencies of 2kHz and 20kHz. It can be seen from Table VI that the inverter performance improves with bit precision and satisfactory performance is obtained with 16-bit precision and shown in Fig. 10.

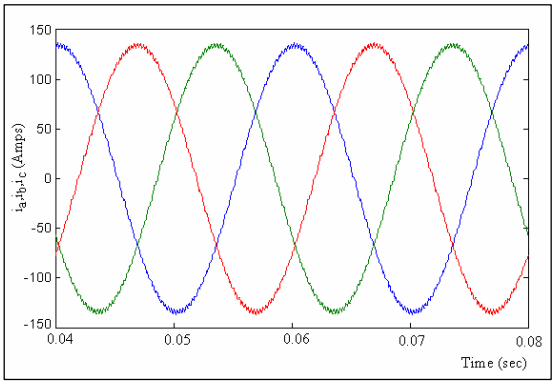


Fig. 10 Line Currents of the Inverter with the ANN having 16-Bit Precision for f_s=2kHz

In this section NN based SVM is designed to be independent of inverter switching frequency. The inverter performance for the architecture 1-6-6-3, the NN based SVM with different bit precision is studied and reported. The hardware implementation results validate the proposed generalized formula. To decide the optimal bit precision of the NN based SVM the performance of inverter is studied for different bit precisions. The NN based SVM with 16-bit precision is concluded to be optimum in terms of implementation and inverter performance.

V. Conclusion

The issues involved with the implementation of a single neuron in FPGA are discussed in this paper. The best possible implementation with resource /speed trade off involves aspects like Serial/parallel implementation of computational blocks, Bit precision and use of LUT. These aspects are discussed in detail and the procedure to arrive at an optimal solution for a given problem is presented. A single neuron is implemented using the computational method and LUT method and the hardware results are presented. The hardware implementation is carried out for a neuron with various inputs and excitation functions. From the results obtained generalized formulae to approximately determine the resource in slices and speed in cycles of a neural network is derived.

To demonstrate and validate the hardware implementation issues proposed in this paper, an application, namely, a NN based SVM technique for a Voltage Source Inverter is presented. A 1-6-6-6-3 architecture is chosen. The NN based SVM is designed to be independent of inverter switching frequency. The performance of the NN based SVM for different bit-precisions is investigated and the results are reported. The hardware implementation is carried out using IC 'XCV400hq240' and verified. The generalized formula obtained is also validated. The performance of the Inverter driven by NN based SVM with different bit precisions is discussed. It is identified that 16-bit precision is optimum in terms of implementation and inverter performance. The methodology proposed and results presented in this paper will aid neural network design and implementation.

Acknowledgment

The research is supported by the grants from the All India Council for Technical Education (AICTE), a statutory body of Government of India. File Number: *8020/RID/TAPTEC-32/2001-02.*

References

[1] B.Widrow and R.Winter, "Neural nets for adaptive filtering and adaptive pattern recognition ", IEEE Computer magazine, pp. 25-39, March 1988.
[2] K.Fukushima, S.Miyake and T.Ito, "Neocognitron: A neural network model for a mechanism of visual pattern recognition", IEEE transactions on systems, Man and Cybernetics, vol.13, no.5, pp. 826-834, 1983.
[3] M.Cristea, A.Dinu, "A New Neural Network Approach to Induction Motor Speed Control", IEEE power electronics specialist conference, vol. 2, pp. 784-788, 2001
[4] S.Grossberg, E.Mingolla and D.Todorovic, "A neural network architecture for pre-attentive vision", IEEE Transactions on Biomedical Engineering, vol.36, no.1, pp. 65-84, Jan 1989.
[5] Leonardo Maria Reyneri "Implementation Issues of Neuro-Fuzzy Hardware: Going Towards HW/SW Codesign" IEEE Transactions on Neural Networks, vol.14, no.1, pp. 176-194, 2003.
[6] Y.J.Chen, Du Plessis, "Neural Network Implementation on a FPGA ", Proceedings of IEEE Africon, vol.1, pp. 337-342, 2002.
[7] Sund Su Kim, Seul Jung, "Hardware Implementation of Real Time Neural Network Controller with a DSP and an FPGA ", IEEE International Conference on Robotics and Automation, vol. 5, pp. 3161-3165, April 2004.
[8] Turner.R.H, Woods.R.F, "Highly Efficient Limited Range Multipliers For LUT-based FPGA Architectures", IEEE Transactions on Very Large Scale Integration Systems, Vol.15, no.10, pp. 1113-1117, Oct 2004.
[9] Marchesi.M, Orlandi.G, Piazza.F, Uncini.A, "Fast Neural Networks Without Multipliers", IEEE Transactions on Neural Networks, vol. 4, no.1, Jan 1993.
[10] Babak Noory, Voicu Groza, "A Reconfigurable Approach to Hardware Implementation Of Neural Networks", Canadian Conference on Electrical and Computer Engineering, IEEE CCGEI 2003, pp. 1861-1863, 2003.
[11] S.Himavathi, B.Umamaheswari "New Membership functions for effective Design and Implementation of Fuzzy Systems", IEEE Transactions on Systems, Man, Cybernetics, Part A, vol. 31, no.6, Nov 2001.
[12] Anitha " FPGA Implementation of Estimators for sensorless control of DTC Drives", M.Tech Thesis, Pondicherry Engg College, India, June 2005.
[13] B.K.Bose, *Modern Power Electronics and ac drives*, Pearson Education (Singapore) Pvt. Ltd., India, 2003.