Using Xilinx FPGAs to Implement Neural Networks and Fuzzy Systems

JJ Blake, LP Maguire, TM McGinnity, LJ McDaid

1. Introduction

Over the last thirty years, since Zadeh first introduced fuzzy set theory, there has been widespread interest in the real-time application of fuzzy logic, particularly in the area of control [LEE90]. Recently, there has been considerable interest in the development of dedicated hardware implementations [YAMAKAWA93, WATANABE90] which facilitate high speed processing. However, the main drawback of such an approach is that it is only cost effective for high-volume applications. A more feasible methodology for lower volume problems demands the application of general-purpose or programmable hardware such as the Xilinx FPGAs. There has been a similar trend in the area of neural networks, as initial research employed software simulations but more recent interest has investigated hardware implementations [BOTROS93].

FPGAs are becoming increasingly popular for prototyping and designing complex hardware systems. The structure of a FPGA can be described as an "array of blocks" connected together via programmable interconnections. The main advantage of FPGAs is the flexibility that they afford. An engineer can change and refine a device's design by exploiting the device's reprogrammability. Xilinx introduced the world's first FPGA, the XC2064, in 1985. This contained approximately 1000 logic gates. Since then, the gate density of Xilinx FPGAs has increased 25 times. There has been a lot of recent interest in the FPGA realisation of fuzzy systems [MANZOUL95, SALAPURA95]. Similarly there are a number of FPGA implementations of neural networks reported in the literature [ELDREDGE96, BADE94]. However, this paper provides a report on the implementation of both architectures and also offers a comparison with the hybrid structure.

2. Circuit Elements

The application selected in this work was a non-linear function approximation problem given by equation 1 :

$$y = (1 + x_1^{0.5} + x_2^{-1} + x_3^{-1.5})^2 \qquad \text{....(1)}$$

where x_1, x_2 and x_3 are the three input variables confined to the range 1 to 5. This highly non-linear problem has been employed by a number of other researchers to demonstrate the ability of both fuzzy and neuro-fuzzy systems. The inputs and weights of the two systems are represented using a special 8 bit encoding technique where 10000000 represents 4, 01000000 represents 2 ,..... 00000001 represents 0.03125. Various numbers are represented by simply adding these values together. Table 1 illustrates the various different components that make up the three systems and the following sections describes the methods employed to implement them in hardware :

Neural Network	Fuzzy Logic
Multipliers	Multipliers
Activation Function (Sigmoid)	MIN operators
Summers	Summers

Table 1 *Components used in the two systems*

2.1 Multiplier Design

The multiplier design utilized is a two-input 8 bit multiplier. Fig. 1 shows a block diagram of the multiplier design. The input to the multiplier (the multiplicand) is divided by 16 and then repeated addition is carried out to obtain the output. In such a design there is a trade-off between bit resolution and hardware size.

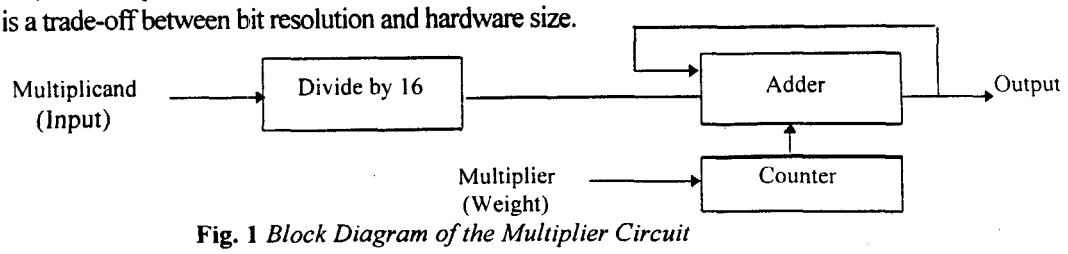

Fig. 1 *Block Diagram of the Multiplier Circuit*

Faculty of Engineering, University of Ulster, Magee College, Northland Rd., Derry

2.2 Activation Function (Sigmoid)

The sigmoid function is the traditional nonlinear activation function used in neural networks. The sigmoid function is not suitable for direct digital implementation as it consists of an infinite exponential series. Many implementations use a lookup table to approximate the sigmoid function. However the amount of hardware required for these tables can be quite large especially if one required a reasonable approximation. A simple second order nonlinear function exists which can be used as an approximation to a sigmoid function [Kwan 92]. This nonlinear function can be implemented directly using digital techniques. The following equation is a second order nonlinear function which has a tanh-like transition between the upper and lower saturation regions :

$$F(z)=\begin{cases} z(\beta - \theta\ z) & for \quad 0 \le z \le L \\ z(\beta + \theta\ z) & for \quad -L \le z < L \end{cases} \qquad \text{....(2)}$$

where β and θ represent the slope and the gain of the nonlinear function F(z) between the saturation regions - L and L. Fig. 2 shows a block diagram of the activation function implemented using this process.

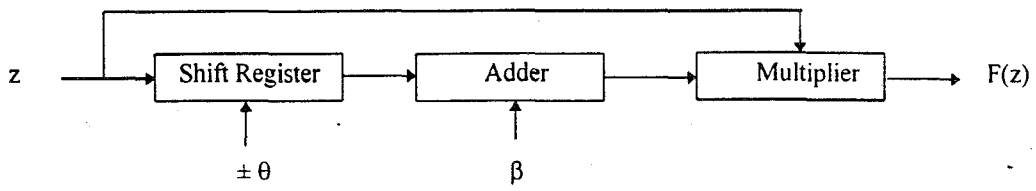

Fig. 2 *Block Diagram of the Sigmoid Function*

F(z) is implemented by one binary shift , one binary addition and one multiplication. Fig. 3 shows the comparison between the bipolar sigmoid and the hardware approximation. This clearly shows that the hardware realization of the sigmoid function provides a reasonable approximation to this function.

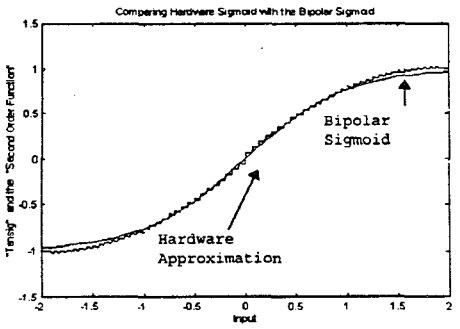

Fig. 3 *Bipolar Sigmoid Function and Hardware Approximation*

2.3 MIN Operators

The intersection of the fuzzy rules in the fuzzy logic circuit was implemented using the MIN operator. This operator was employed in preference to a PRODUCT operator because its hardware realization reduced the amount of circuitry necessary and therefore minimized the CLB utilization of the FPGA. The MIN operator was implemented using two subtractors as illustrated in Fig. 4 for the problem of determining the minimum of three inputs.

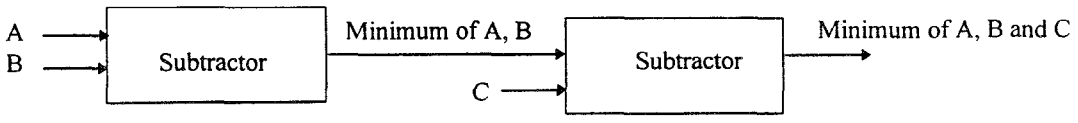

Fig. 4 *Block Diagram of MIN Operator*

2.4 Summers

The maximum number obtained from the non-linear approximation problem is approximately 27.42, when the input variables are confined to the range 1 to 5. This number can be represented by the encoding technique as 0000 0011 0110 1110. Therefore to accommodate this number in the coding scheme the summers utilized are 16 bit adders. These adders are the only 16 bit devices in the two architectures.

3. The fuzzy and neural network architectures

The following sections provide an overview of the representation of the fuzzy system and neural network employed in this work. All the architectures were first implemented using the MATLAB neural network toolbox. The weights obtained from the implementation were then used in the hardware realizations. Both of the architectures were evaluated using an arbitrary set of twenty input-output pairs.

3.1 Fuzzy Logic

Fuzzy inference systems have been successfully applied in a diverse range of areas and as a result of this multi-disciplinary approach there has evolved a range of different approaches and nomenclatures. The interpretation of fuzzy reasoning used in this work is commonly referred to as a zero-order Sugeno fuzzy model [SUGENO88]. In this model the consequent of each fuzzy *IF THEN* rule is represented by a fuzzy singleton. As a result, the inferred output from each rule is defined as a crisp value induced by the rule's firing strength. The overall output is taken as the weighted average of each rule's output which avoids the time consuming process of defuzzification. Thus for an n-input single output problem where each input domain is partitioned into p fuzzy sets, there will be p^n rules and a total of np distinct fuzzy sets. Fig. 5(a) illustrates the layout of the circuit and the four stages of the fuzzy reasoning process: fuzzification, intersection, implication and defuzzification.

The partitioning strategy selected used two triangular fuzzy sets on each input domain, which can be termed Small and Big. Note the Small set has a maximum membership value for a 1 input and a minimum membership value for 5. The set Big is the converse of the Small set. The membership values of these inputs was realized in hardware using ordinary shift registers and adders. The MIN operator was used for the intersection of the fuzzy rules. Multipliers were used in the implication process while the defuzzification process uses a summer.

The results (see Section 4) demonstrated that the fuzzy approach is relatively poor at approximating this function except at those cases where the inputs are 1 and 5 corresponding to the fuzzy set definitions. This is as expected as in the interval between 1 and 5 the fuzzy system is effectively performing linear interpolation to obtain an approximate value of the output. However it was found that the hardware results offer a similar degree of accuracy as the results achieved in software.

3.2 Neural Network

The neural network architecture employed was the conventional three layer feedforward architecture (see Fig. 5(b)). The nodes in the first layer simply broadcast the inputs and perform no processing, and an n-input problem requires n nodes in this layer. The determination of the number of nodes in the hidden layer is more arbitrary and application dependent. In this work, the number of nodes was selected by choosing a network which was of similar dimension to the fuzzy system and yet of appropriate size to perform the approximation accurately. A sigmoid non-linear squashing function was utilized in each of these nodes in the hidden layer. As this is a single output application there is only one node in the output layer which is essentially a summer. The results obtained are more accurate indicating that the neural network is more effective at approximating the function (see Section 4).

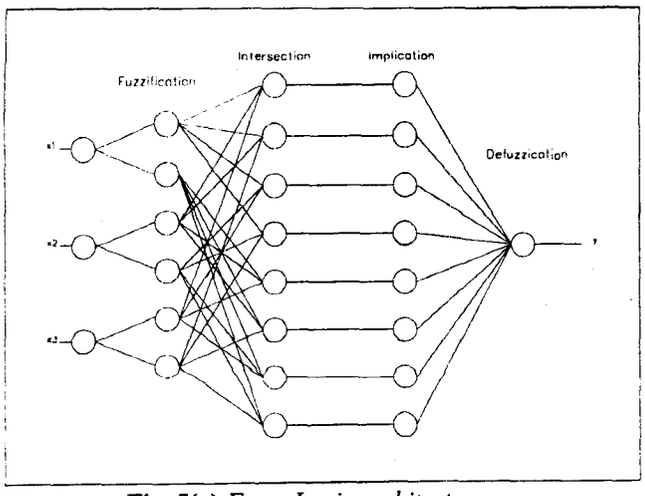

Fig. 5(a) *Fuzzy Logic architecture*

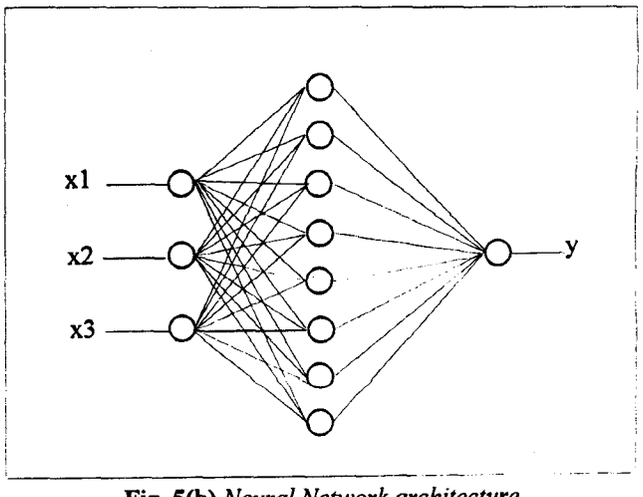

Fig. 5(b) *Neural Network architecture*

4. Discussion

The accuracy of the results were evaluated using the following performance index :

$$J = \frac{1}{T}\sum_{i=1}^{T}\frac{d_i - y_i}{d_i} \times 100\,\%$$

where d_i and y_i are the target output and the output inferred from the two systems respectively for the i^{th} data pair for a total of T data sets. The performance index results, expressed as a percentage, are presented in Table 2 using the twenty training data.

System	J(%) Software	J(%) Hardware	CLB Utilization	Suitable FPGA
Fuzzy Logic	55.40%	51.97%	286 CLBs	XC4008
Neural Network	13.05%	20.29%	1290 CLBs	XC4025+XC4005

Table 2 *Performance and CLB Utilization of the three systems*

The fuzzy system approach demonstrates a very poor approximation ability of this non-linear function. This, however can be explained as the fuzzy reasoning adapted a very crude partitioning strategy with only two sets on the input domains. Clearly this is not sufficient for this problem. Similiarly a neural network with more neurons in the hidden layer would probably have given a better performance index. However a neural network with more hidden neurons would demand a larger circuit and more dense interconnections.

The CLB utilization of the two architectures is also given in Table 2. The fuzzy logic implementation is the smallest circuit in terms of CLB usage. The neural network consumed the most CLBs, a total of 1290 CLBs which may be implemented on two FPGAs, an XC4025 and an XC4005.

5. Conclusion

The results of this work successfully demonstrate the hardware implementation of fuzzy systems and neural networks using Xilinx FPGAs. This allows comparisons to be made between the hardware realisations of these technologies. The paper used a modular approach to identify elements of each architecture. A three input non-linear function approximation problem was employed to demonstrate the validity of the different architectures. A software simulation was used to benchmark the accuracy of the results and to enable comparisons with actual results.

Although the fuzzy logic used the least amount of CLBs it provided a poor approximation of the problem. In contrast the neural network gave a reasonable approximation but consumed a larger number of CLBs. The continual developments in FPGA technology and their associated cost, and reprogrammability make this approach a viable alternative to the development of custom hardware for real-time applications.

References

[BADE94] Bade, S; Hutchings, BL; "FPGA based stochastic neural network implementation", Proc. IEEE workshop on FPGAs for Custom Computing Machines, pp.189-198,1994.

[BOTROS93] NM Botros; M. Abdul-Aziz: "Hardware implementation of an artificial neural network" Proc. IEEE Int. Conf. on Neural Networks, pp. 1252-1257, 1993.

[ELDREDGE96] Eldredge, JG; Hutchings, BL: "Run-time configuration : A method for enhancing functional density of SRAM based FPGAs", Journal of VLSI Signal Processing, Vol. 12, No. 1, pp. 67-86, 1996.

[KWAN92] Kwan, HK : "Simple Sigmoid-Like Activation Function Suitable for Digital Hardware Implementation", Electronics Letters, pp. 1379-1380, Vol. 28, No. 15, 1992.

[LEE90] Lee, CC : "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - part I and II", IEEE Trans SMC, pp 404-435, Vol.20, No.2,1990.

[MANZOUL95] Manzoul A; Jayabharathi, D : "FPGA for fuzzy controllers", IEEE Trans. SMC, pp. 213-216, Vol. 25 No 1, Jan.1995.

[SALAPURA95] Salapura, V; Hamann, V : "Prototyping fuzzy controllers using VHDL and FPGA technology", International ICSC Symposium on Fuzzy Logic, Zurich, Switzerland, May 1995.

[SUGENO88] Sugeno, M; Kang, GT: "Structure identification of fuzzy model", Fuzzy Sets and Systems, Vol. 28, pp. 15-33, 1988.

[WATANABE90] Watanabe, H; Detloff, WD; Yount, KE "A VLSI Fuzzy Logic Controller with Reconfigurable, Cascadable Architecture", IEEE Journal of Solid State Circuits, pp. 376-382, Vol.25, No.2, April 1990.

[YAMAKAWA93] Yamakawa, T : "A Fuzzy Inference Engine in non-linear analog mode and its application to a Fuzzy Logic Control", IEEE Trans. on Neural Networks, pp. 496-522, Vol. 4, No. 3, May 1993.