VHDL Modeling of an Artificial Neural Network for Classification of Power Quality Disturbance

FLORENCE CHOONG, F. MOHD-YASIN, M.S. SULAIMAN, M.I. REAZ

Faculty of Engineering

Multimedia University

63100 Cyberjaya Selangor

MALAYSIA

Abstract: - This paper describes the design and modeling of an artificial neural network (ANN) classifier using VHDL. This classifier is targeted primarily to classify the six different types of power quality disturbance. The high level architecture comprises of a control unit and a neural network datapath. The control unit is further divided into five interconnected sub modules: bus master, ram, pseudo random number generator, error calculator and trainer. *Univariate randomly optimized Neural Network (uronn)* algorithm is employed to model the neural network. Proper simulation is carried out to verify the functionality of the individual modules and the system. In addition, the algorithm was also implemented in Matlab and C as comparison with the hardware implementation in VHDL. Comparisons, verification and analysis made validate the advantage of this approach. Currently, the classification average accuracy is 77.53%. The classifier also has the potential of being extended to classify other kinds of power quality disturbances.

Key-Words: - Artificial neural network, power quality, VHDL, FPGA, classification, modeling

1. Introduction

Power quality is defined as any deviation from a perfect sinusoidal waveform that can result in failure or misoperation of equipment [1]. In a power system, faults, dynamic operations or non-linear loads often cause various types of power quality disturbances such as voltage sags, voltage swells, switching transients, interruption and fluctuation [2]. Power quality disturbance waveform recognition is often troublesome because it involves a broad range of disturbance categories or classes, and therefore the decision boundaries of disturbance features may overlap [3].

Artificial Neural Networks attempt to recreate the behavior of a biological brain in logic to solve complex problems [4]. They consist of nodes (neurons) in which inputs are multiplied by pre-trained weights and are summed together to produce overall outputs. Artificial Neural Networks can be used to solve complex problems with a comparatively small amount of effort. It has been effectively applied in power system applications with great appreciation [2,5,6,7] and has been used in various segments of industry to solve several problems including power quality [8,9]. For a partial list of ANN applications in electric power industry, the reader is referred to [10].

There are a few works that had been done in this area. In one of the work, the neural network is designed for high-speed processing which can provide selective real-time detection and classification of faults. The method involves utilizing both supervised and unsupervised neural network training. The training involved two different training samples, one with a reduced number and the other a large number of samples simulated using the Electromagnetic Transient Program (EMTP) [11]. The second work proposes a recognition scheme performed in the wavelet domain using a set of multiple neural networks. Wavelet transform coefficients are used as inputs to the neural network. Since multiple neural networks are utilized, the outcome of the network is then integrated using a decision making scheme [3]. Another work employs an ANN-based methodology using a time-delay neural network (TDNN) for the classification of disturbance waveforms. The TDNN exhibits a translation-shift invariance property. The standard back-propagation is applied to each time-shifted network to obtain the error derivatives for each unit in the network. The TDNN extracts temporal relationships from the input data in the classification process [7].

However, the classification methods employed in the works mentioned above are based mainly on software implementation. Software simulations are useful for investigating the capabilities of neural network models and creating new algorithms; but hardware implementations remain essential for taking full advantage of the inherent parallelism of neural network and provide an increase in speed and performance. For example, Field Programmable Gate Array (FPGA), an application-specific integrated circuit does provide a speed-up of several orders of magnitude compared to software simulation [12]. This will allow all of the neural network calculations to be done directly and in parallel. This will contribute a significant increase in performance compared to a neural network implemented in software where the many internal neural network calculations must be compiled into multiple individual commands for a processor to complete sequentially.

ANN developed in software has several drawbacks. A limiting factor is that the size of neural network is limited by the size of the system that runs the compiled neural network code on. A processor that has a small cache or that is slow may not be able to handle a large amount of data to be used to model a system or be able to train on a data from a system in a reasonable amount of time.

With hardware, the memory storage for the neural network is large and dedicated to the neural network and not dependent on a processor. FPGA implementation greatly reduces the size, offers a higher reliability, improved security, higher performance, higher accuracy and also the ability to add on features easily. Instead of using conventional programming languages such as C, Matlab or Java, the new method models the design of a neural network in Very High Speed Integrated Circuits Hardware Description Language or VHDL. This is a hardware independent language, which supports a design to be modified easily to suit research requirement.

Some useful features of the new approach are listed below:

1) Artificial Neural Network is used for its pattern recognition capabilities. However, its ability to perform well is greatly influenced by the weight adaptation algorithm. Thus a unique algorithm, *Univariate randomly optimized Neural Network (uronn)* was chosen for implementation. This will provide significant increase in the accuracy of the classification.

2) The use of Carry-save-adder (CSA) as the multiplier component in each neuron provides a speed-up in the overall computation and thus reduces training time.

3) The VHDL model provides a systematic approach for hardware realization, facilitating the rapid prototyping of neural network for power system applications.

4) The method owns the potential of being extended to classify other kinds of disturbances.

This paper is organized as follows: the algorithm is described in section 2, section 3 describes the individual modules that make up the system, section 4 is dedicated to presenting the results and discussions, and finally, section 5 presents the conclusion and future works.

2. Design Overview

The algorithm chosen for implementation was the *Univariate randomly optimized Neural Network (uronn)* [4]. This algorithm is robust for comparing training speeds between software and hardware.

Besides that, this algorithm is a very advantageous approach in the implementation of a neural network trainer because less calculation needs to be performed in each epoch because this algorithm searches for the weights that best fit the neural network by randomly changing them in order to minimize error. Thus the calculation to determine the rate of change of error to rate of change of weights as in equation (1) can be eliminated.

$$Slope = \frac{\Delta Error}{\Delta W} = \frac{dE}{dW} \tag{1}$$

In addition, this algorithm utilizes 8-bit signed integers as values for weights and inputs instead of 32-bit floating points values. Arithmetic operation using floating-point operands is significantly more complex than signed integer operands [4].

The network is trained using a training data set that consists of input/output pair. The inputs are the sample values of power quality disturbances that are generated using PSCAD and MATLAB software. The classes include transients, sag, swell, interruption, fluctuation and perfect wave. Simulated samples of each class are preprocessed using Matlab wavelet transform toolbox. This will generate approximate and detailed coefficients of the original signal as represented in equations (2) and (3) respectively:

$$cA_i(n) = \Sigma_k\ f(n)*h_d(-k+2n) \qquad (2)$$
$$cD_i(n) = \Sigma_k\ f(n)*g_d(-k+2n) \qquad (3)$$

The 8-bit output classifies the type of power quality disturbance.

3. The Developed Approach

The high-level design consists of two key components: the control unit and the neural network datapath, that is interconnected and interact with one another as shown in Figure 1.

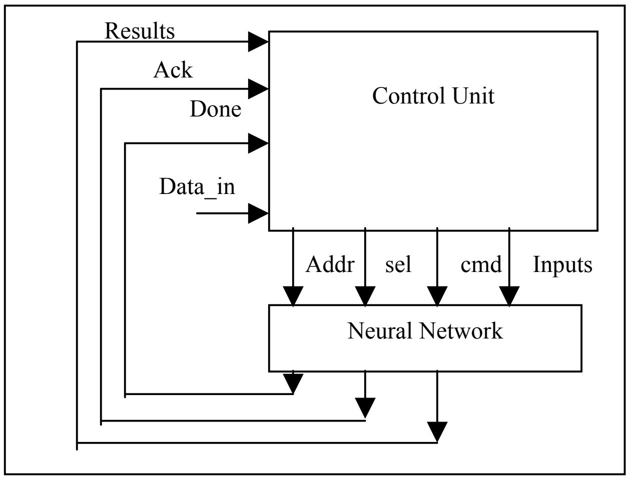

Fig. 1. High-level system design

3.1 Neural Network

This module is one of the important modules as it performs the classification task. It consists of neurons and their connections into a network. A neural network consists of simple processing elements known as neurons. The input of a neuron is usually an input column-vector $x = [x\ (1),\ldots..,x(p)]^T$ of a preprocessed signal. The *n*th element of the input vector, x(n), is connected to a neuron k by a weight factor m(k,n). The weight factor then forms a weight vector for neuron k, $m_k = [m(k,1),\ldots.,m(k,p)]$. The output of the neuron is simply a linear combination of the input vector x with the weight vector m_k, as shown in equation (4):

$$u_k = mx^T = \Sigma\ m(k,i)x(i) \text{ for i from 1 to P} \qquad (4)$$

The Neuron architecture is modeled using the Carry-Save-Adder prototype as its multiplier component for its high speed and regular structure. The datapath contains three neurons, two for the hidden layer and one for the output layer, and are connected in a traditional feed-forward architecture. For this implementation input neurons were not considered, as their purpose is to merely distribute the inputs among the neurons at the hidden layer. Thus, in the design the inputs come directly from the data bus into the neurons at the hidden layer.

3.2 Control Unit

The control unit is used to control the neural network. It consists of five sub modules modeled in a top-down design approach [14] as described below:

3.2.1 Trainer

This module basically utilizes state machines governing control signals for the neural network.

3.2.2 Bus Master

This component is designed to get and set the weights and to activate the neural network during operational mode. The data bus controller controls the transfer of data through the neural network, and supplies the network with new random weights as well as the inputs. The datapath entity is instantiated for each neuron. Subsequently, each neuron will have a unique identifier corresponding to the number in which it was instantiated. A comparator uses this identifier to ensure that the value being passed in on the address bus lines matches the identifier of the neuron. If a match is detected, then it sets the equal signal to high, and the logic gates for the neuron is used. Along with the address bus and the data bus, there exists a command bus and select bus. The command bus provides the neuron with one of four commands utilizing 2 bits, which also act as inputs to the combinational logic. “00” is to read the weights of the neurons, i.e. send to the datapath, “01” is to writes weights of the neurons, i.e. take from datapath and store, “10” is idle and “11” is to begin forward calculation The select bus tells the neuron which weight it is dealing with. The combinational logic determines from the bus inputs which weight is being used and what operation is to perform. The bi-directional 8-bit data bus exists to provide the weight storage units with their values during storage and to take the values from the storage units when reading.

The address bus is used to inform the neural network which neuron it should be performing a read or write operation.

3.2.3 Error calculator

Error calculator keeps a count of how many times the network misclassifies input data and uses this to tell the control unit whether a new weight should be kept or discarded. This module compares the output evaluated by the neural network from the inputs provided with the current weights, and compares it to the expected target output. If the evaluated output is closer to the target value, the current random weight being tested replaces the previous weight value.

Let t_k be the k-th target or desired output and Z_k be the k-th computed output with k=1,…….K and w represents all the weight in the network. The training error is calculated using equation (5):

$$Error = \frac{\sum_{n=1}^{N}\sum_{k=1}^{K}(Z_{kn}-t_{kn})^2}{NK} \tag{5}$$

where N = number of pattern and K = no of outputs

3.2.4 Pseudo random number generator

The random number generator generates the random weights used by the neural network. It employs the linear feedback shift register (LFSR) that consists of the input sequence (initialization vector), the feedback (tap sequence), and the output. An LFSR is a mechanism for generating a sequence of binary bits. The register consists of a series of cells that are set by an initialization vector that is, most often, the secret key. A clock regulates the behavior of the register and at each clocking instant, the contents of the cells of the register are shifted right by one position, and the exclusive-or of a subset of the cell contents is placed in the leftmost cell [15]. The feedback gives a linear relationship between the input and the output. Let the input sequence of length n be (s_0, s_1, …, s_{n-1}). The feedback is thus a linear function f (s_0, s_1, …, s_{n-1}) defined by equation (6):

$$f(s) = \sum_{i=0}^{n-1} c_i s_i \tag{6}$$

where c_o, c_1, …, c_n are constant coefficients. The output of the LFSR is determined by the initial values s_0, s_1, …, s_{n-1} and the linear recursion relationship as shown in equation (7) and (8):

$$s_{k+n} = \left(\sum_{i=0}^{n-1} c_i s_{i+k}\right),\ k \geq 0 \tag{7}$$

or equivalently,

$$\sum_{i=0}^{n} c_i s_{i+k} = 0,\ k \geq 0 \tag{8}$$

where c_n = 1 by definition [15].
Figure 2 [15] shows a sample LFSR of a generic length n.

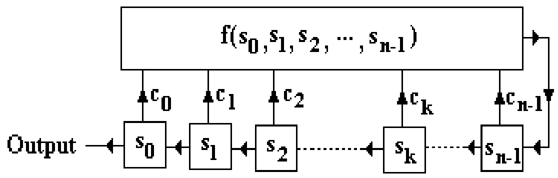

Fig. 2. Linear feedback shift register

3.2.5 RAM interface

This module handles the storage of the training data into RAM. It also recalls the training data set for the control unit for subsequent training epochs. The values are stored as 32-bit binary.

4. Results and Discussion

4.1 Software Implementations

To see the effectiveness of this algorithm, it was implemented in Matlab and C. Table 1 summarizes the results obtained.

Both methods showed to be slower and have lower average classification result. However this experiment was performed for statistical use only, and to have a comparison point for the neural training that is being implemented. This is an ideal study case to compare speed on software-developed systems and hardware description based systems.

Table 1 - Comparison on the speed of software developed algorithms for experimentation

	Matlab	**C**
Number of iterations	**Correct Classification (%)**	**Correct Classification (%)**
1000	65.1%	60.4
2000	71.4%	69.15

4.2 VHDL Simulation and Results

The classification experiments were done with power quality disturbance signals simulated using MatLab and PSCAD. The results of a 6-class classification are shown in Table 2.

Table 2: Neural network performance on the test data (C-class, 1-transient, 2-voltage sag, 3-voltage swell, 4-interruption,5-fluctuation, 6-normal, Correct-percentage of correct identification, mi-percentage of mistake identification to class i)

C	Correct	M1	M2	M3	M4	M5	M6
1	65	-	4.6	7.5	-	20.1	2.8
2	77.6	4.4	-	-	4.7	13.3	-
3	72.5	9.4	-	-	5.4	12.7	3.7
4	89.5	3.9	-	-	-	6.6	4.2
5	70.1	11.5	7.9	10.5	-	-	-

Wavelet transform using Matlab were then performed on each sample disturbance to generate the detail and approximate coefficients. The coefficients will be the inputs to the neural network. A total of 300 examples (50 examples per class) were used to test the classification method. The starting time, duration and distortion magnitude are generated randomly. This makes the testing results more reliable because none of these are fixed for real power system disturbance events. Testbenches were used to perform extensive testing on each individual component. Performance on the test set reached an average of 77.53% agreement.

Figure 3 shows a sample simulation result from classifying an input/output pair. At the rising edge of the clock signal, In1 (approximate coefficient) and In2 (detail coefficient) will be taken in as the inputs and will be multiplied by the weights, We1 and We2 respectively. These values will be processed by the datapath to perform the classification. The active high Done signal indicates that the classification task is complete and the final result on the result register that reflects the type of power disturbance. For classification purpose, each disturbance type is assigned a fixed output. In the sample simulation in Figure 3, an output of "00" on the result register represents a sag disturbance and "FF" represents transient.

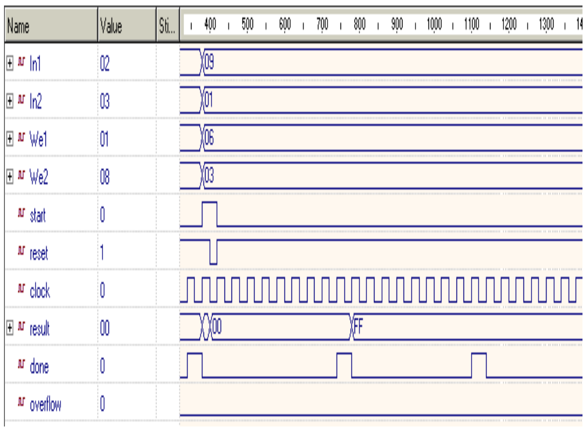

Fig. 3 Sample simulation result

The average percentage of accuracy achieved is 77.53%. When comparing with similar works in this area, the percentage of this method is lower. However, an increase in the accuracy is expected by increasing the number of training samples for the future works. In [7], a feedforward, time-delay, and modified time-delay neural network are three different implementations that were investigated: Classification accuracy obtained is 72%, 57% and 93% respectively. As for [11], classification accuracy within the range of 52% and 92% is obtained. These works were based on software implementations. Fast digital computers with mega/giga potential may be used but there still exist problems and limitations for which they will never be fast enough compared to hardware platforms. This method offers the flexibility in terms of speed and design cycle time. The use of VHDL helps in speeding the execution process and designs can be developed and tested efficiently and in a shorter design cycle time. These important characteristics are needed in most of the digital components today.

5. Conclusions and Future Work

The objective of this project was to develop a VHDL model of an artificial neural network aimed at classifying power quality disturbances. The initial work involved determining the functions of the system and designing modules to perform these functions. The functionality and timing behavior of the modules were successfully verified. The model was also implemented with Matlab and C programs for comparison with VHDL implementation.

At present, the amount of training data set used for testing are small and the data were generated from a program in PSCAD and MATLAB. The next step is to use the real training data set obtained from power utility companies to increase the algorithm generalization capability and produce more accurate result. Further research will then be carried out for hardware implementation using FPGA. This could take advantage of the high speeds achievable using hardware, and as a result would be a beneficial and economic investment for designs requiring artificial Neural Networks.

References:

[1] Min Wang: "Automatic Recognition of Power Quality Disturbances", *APT Centre, MSEE Thesis Presentation*, August 9, 2001.

[2] Mladen Kezunovic, Yuan Liao: "A Novel Software Implementation Concept for Power Quality Study", *IEEE Transactions on Power Delivery*, Vol. 17, No. 2, April 2002, pp. 544-549.

[3] Surya Santoso: "Power Quality Disturbance Waveform Recognition Using Wavelet-Based Neural Classifier – Part 1: Theoretical Foundation', *IEEE Transaction on Power Delivery*, Vol. 15, No. 1, January 2000, pp. 222-228.

[4] Looney, Carl: "*Pattern Recognition Using Neural Networks: Theory and Algorithms for Engineers and Scientists*", Oxford University press, New York, 1997, Chapters 3, 4 and 6.

[5] David L. Lubkeman, Chris D. Fallon, A. A. Girgis. "Unsupervised Learning Strategies For The Detection and Classification of Transient Phenomena on Electric Power Distribution Systems", *Proceedings of the First International Forum on Applications of Neural Networks to Power Systems*, pp. 107-111.

[6] C. J. Kim, B. D. Russell, "Classification of Faults and Switching Events by Inductive Reasoning and Expert System Methodology", *IEEE Transactions on Power Delivery*, Vol. 4, No. 3, July 1999, pp. 1631-1637.

[7] A.K. Ghosh, David L. Lubkeman. "The Classification of Power System Disturbance Waveforms Using A Neural Network Approach", *IEEE Transactions on Power Delivery*, Vol. 10, No. 1, January 1995, pp. 109-115.

[8] S. Haykin, "*Neural Networks – A Comprehensive Foundation"*, New York: IEEE Press and Macmillian, 1995, Chapters 2 and 3.

[9] L. Fausett, *Fundamentals of Neural Networks*, New Jersey: Prentice-Hall, 1994, Chapters 4 and 6.

[10] M. Kezunovic and I. Rikalo, "Detect and Classify Faults Using Neural Nets," *IEEE Computer Applications in Power*, Vol. 9, No. 4, 1996, pp. 42-47.

[11] M. Kezunovic, Igor Rikalo, D.J. Sobajic. "High-Speed Fault Detection and Classification with Neural Nets", *Electric Power Systems Research*, No 34, February 1995, pp. 109-116.

[12] S. Hauck, "The Roles of FPGAs in Reprogrammable Systems" *Proceedings of the IEEE*, 86(4), April 1998, pp. 615-638.

[13] Taewhan Kirn, William Jao, Steve Tijiang: "Arithmetic Optimization using Carry-Save-Adders', *World Wide Web URL* at http://www.sigda.org/Archives/ProceedingArchives/Dac/Dac98/papers/1998/dac98/pdffiles/26_2.pdf

[14] Stefan Sjoholm, Lennart Lindh: "*VHDL for Designers"*, Prentice Hall, 1997, pp. 261-263.

[15] "Linear Feedback Shift Registers", *World Wide World Wide Web URL* at http://www.math.cudenver.edu/~wcherowi/courses/m5410/m5410fsr.html, pp 1 – 8, 09 January 2002.