American Journal of Applied Sciences 8 (10): 1054-1060, 2011
ISSN 1546-9239
© 2011 Science Publications

Design and Implementation of Novel Artificial Neural Network Based Stock Market Forecasting System on Field-Programmable Gate Arrays

[1]Md. Syedul Amin, [2]Md. Mamun, [1]Fazida Hanim Hashim,
[1]Jubayer Jalil and [1]Hafizah Husain
[1]Department of Electrical, Electronic and Systems Engineering,
Universiti Kebangsaan Malaysia, 43600, UKM, Bangi, Selangor, Malaysia
[2]Systems Design Lab, Universiti Kebangsaan Malaysia,
43600, UKM, Bangi, Selangor, Malaysia

Abstract: Problem statement: Multiagent system is very proficient and has rules well-suited for financial forecast with its neural network. In financial forecasting, the approach for rules extractions is less pertinent and involves algorithms which are complex. The unsupervised network method lacks in comprehensibility and leads to ambiguity. **Approach:** The application of neural network technology to real-time processing of financial market analysis demands the development of a new processing structure which allows efficient hardware realization of the neural network mechanism. This study describes the realization of neural network on FPGA device for stock market forecasting system. The stock market forecasting neural network architecture consists of three layers. These are input layer with three neurons, hidden layer with two neurons and output layer with one neuron. For both output layer and hidden layer neurons, Sigmoid transfer function is used. Neuron of each layer is modelled individually using behavioural VHDL. The layers are then connected using structural VHDL. This is followed by timing analysis and circuit synthesis for the validation, functionality and performance of the designated circuit. The designated portfolio is then programmed through download cable into the FPGA chip. **Results:** Kuala Lumpur Stock Exchange (KLSE) index has been utilized for validating the usefulness of the completed prototype. Test on the sample of 100 data demonstrated an accuracy of 99.16% in predicting closing price of the KLSE index 10 days in advance. **Conclusion:** The test results are anticipated to be a higher rate of prediction for stock market analysis, thereby maintaining the high quality of supplying information in stock market business.

Key words: Neural network, backpropagation, synthesis, VHDL, stock market

INTRODUCTION

In recent years, financial markets have become more interrelated. The fundamental factors are becoming more critical for the analysis of financial market. The research in recent past shows that the nonlinear domain with artificial intelligence technologies can be modeled more precisely compared to single-market and linear statistical methods which have been the mainstay for technical analysis for past decade.

The synergistic market analysis which merges technical and fundamental analysis using artificial intelligence tools and synthesizes fundamental, technical and intermarket data within an analytical framework, results in better forecasting capabilities. Neural networks among various artificial intelligence tools are increasingly used to the financial forecasting as neural nets are found to be technologically flexible and powerful, ideally suited to perform financial market analysis.

Multiagent system (Zimmermann *et al*., 2001; Garliauskas, 1999; Disorntetiwat and Dagli, 2000; Mogaki *et al*., 2007; Seridi-Bouchelaghem *et al*., 2005; Obe and Shangodoyin, 2010) is very proficient and has rules well-suited for financial forecast with its neural

Corresponding Author: Md. Syedul Amin, Department of Electrical, Electronic and Systems Engineering, University Kebangsaan Malaysia, 43600, UKM, Bangi, Selangor, Malaysia
Tel: +603-89216316 Fax: +603-89216146

network. But a drawback is complexity of the system. In financial forecasting, the approach for rules extractions (Kane and Milgram, 1994) is less pertinent and involves algorithms which are complex. The unsupervised network method (Corchado *et al*., 1998) lacks in comprehensibility and leads to ambiguity. It can be applied on standard iterated mapping functions only. Backpropagation is better for prediction of stock market. As it has got a set of defined rules and algorithms for training, backpropagation is the most comprehensive and straightforward method.

The Field-Programmable Gate Arrays (FPGA) provides a potential substitute to speed up hardware implementation (Coussy *et al*., 2009; Marufuzzaman *et al*., 2010; Reaz *et al*., 2007a). FPGA comes with the merits of shorter design cycle, lower cost and higher density from computer-aided design perspective (Choong *et al*., 2005; Akter *et al*., 2008). It contains various building blocks. Each block comprises of programmable storage registers and look-up table. The interconnections are programmed by Hardware Description Language (HDL) among these blocks (Reaz *et al*., 2003; 2004a; 2005a). For prototyping digital system, the simplicity and programmability of FPGA made it appealing. FPGA allows users to easily and inexpensively realize their own logic networks in hardware. FPGA also allows modifying the algorithm easily and the design time frame for the hardware becomes shorter by using FPGA (Choong *et al*., 2006; Mohd-Yasin *et al*., 2004).

This study aims to investigate the hardware feasibility and performance of ANN based stock market forecasting system using FPGA by standard hardware description language VHDL. A formal description of the system and utilization of specific description styles for covering various abstraction levels (logic, architectural and register transfer level) made VHDL attractive (Reaz *et al*., 2006; 2007b). In the computation method, first the problem is separated into small pieces. In VHDL, each of them is a submodule. The synthesis is then activated following software verification of each submodule. It performs the translations of HDL code into a similar digital cells' netlist. To explore a far wider range of architectural alternative, the synthesis helps to integrate the design work and offers higher feasibility (Reaz *et al*., 2005b; 2004b). To validate the effectiveness of the method, Kuala Lumpur Stock Exchange (KLSE) index has been used in this study. For hardware implementation, this method offers a systematic approach, facilitating quick prototyping of neural network for financial market analysis.

MATERIALS AND METHODS

Design overview: FPGA realization of neural network in stock market prediction is introduced to ease the development of financial market analysis. The system is capable of predicting the closing price of the KLSE index 10 days in advance from the current day. The input for the neural network will consist of the daily high, daily low and closing price of the KLSE index.

The neural network architecture for stock market forecasting is a three layer with three neurons in input layer, one neuron in output layer and two neurons in hidden layer (Tan *et al*., 1993). The system is capable of minimizing the output error of the backpropagation neural network to forecast the closing price of the KLSE index 10 days in advance from the current day. The neural network input consists of the daily high, daily low and closing price of the KLSE index. For a stable network with minimum error, we choose a learning rate of 0.3 that is scaled to 307 since it is implemented as integer data type in VHDL. The scaling is done by multiplying the learning rate with 1024, gives a constant value thus synthesizable. The network is trained by using the historical data of the KLSE index from January 2004 to July 2011. This set of epochs is feed into the network three times.

For enabling any combination of number of output neurons, input neurons and hidden neurons, the network is configurable. The output and hidden layer neurons utilize sigmoid transfer function. Every neuron in the output, hidden and input layer is programmed as a single entity. Using structural approach, these neurons combinations are coded in a high level design. To process the input data and initializing the neural network weights to a suitable range, normalization circuit and random number generator are added. The developed neural network is divided in three modes of operation. These are testing, random weight generation and training mode.

System level design: The VHDL code is divided in five entities. Out of these, four are linked together by one entity. FINANCIAL is the top-level entity that contains the BEHAVIOUR architecture for linking the components utilizing VHDL structural approach. INPUT_N, HIDDEN, OUTPUT and BACKPROP are the other four entities. Structural view of the system is illustrated in Fig. 1.

Number format design: Since finite storage devices, also referred to as sequential elements, are being used in the neural network chip, the internal signals must have a numbering format. In this study, number-scaling method is utilized to minimize the complexity and to facilitate synthesis process.

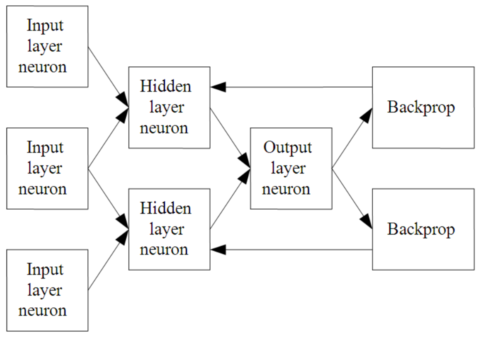


Fig. 1: Structural view of the system

This method multiplies integer type of VHDL with 1024 or 1024^2 depending on the number of multiplication and later divides the number with the same constant to revert to the original scale.

Neuron design of input layer: The input integer layer comprises of three neurons for the neural network design. Acting as a buffer in between hidden layer neurons and data is the input layer's purpose. INPUT is the entity for the input layer neuron and BEHAVIOUR is its architecture.

Neuron design of hidden layer: The hidden layer is consisted of 2 neurons. It uses a sigmoid transfer function. As input, each hidden layer neurons possesses 3 system data of current exemplar. HIDDEN is the entity for hidden layer neuron and BEHAVIOUR is its architecture. The code is written in a way that each neuron is able to perform three modes of operation.

The random weight generator is the first mode of operation. For each three inputs, it generates random weights. If the "mode" input pin is put to "01" of std_logic_vector type, the operation is triggered.

Forward mode is the second mode of operation. The network propagate forward its input to produce output in this mode. All the 3 input data attributes are multiplied by a unique internal weight for every hidden layer neuron produced by random weights generator. Then these 3 results are added and applied to internal transfer squashing function which is the sigmoid function. The transfer function output is the activation output of neuron. Each hidden layer neurons output is fed to the output layer neuron. For testing the network or once the network really acts as stock price predictor, the second mode is used. If "mode" input pin is put to "10" value of std_logic_vector type, the second mode operation is triggered.

Backward propagate mode is the third mode. The network error is backpropagated in this mode. To reduce network error, the threshold and weights are adjusted if the same input is applied to the network. In this process, backpropagation learning algorithm is utilized. For monitoring purposes, hidden layer neuron error is transferred to port hidden_error1 and hidden_error2 of the two hidden layer neuron. The network runs in training mode when neural network chip runs in the third mode. The "mode" input pin is set to "11" of std_logic_vector type if third mode is triggered.

BACKPROP, another entity, is created for the third mode for an efficient operation. Output error is required to correct the hidden layer neuron weights. As such, after the output error is obtained, the weights are corrected. BACKPROP is after the output layer neuron for obtaining the output error. Another purpose is to correct the weights as per the backpropagation-learning algorithm. Corrected weights are fed back to hidden layer neuron. Design flowchart of hidden layer neuron is shown in Fig. 2.

Random weights generator: Random weights generator generates a random number with uniform distribution in the interval between -1024 and 1024. The random weights generator uses a random number between 0-1024 to generate the desired number derived from the following mathematical function Eq. 1:

$$X(n) = A + (B - A) \times R(n) \quad (1)$$

Where:

A and B	= The limits of the interval
X	= A random number between A and B
R	= A random number between 0 and 1

Given interval A = -1024 and B = 1024, therefore,

$$X = -1024 + (2048 \times R) \quad (2)$$

Since we fed random number of range between 0 and 1024, the following formula is applied to the network instead of Eq. 2 and 3:

$$X = -1024 + (2048 \times R / 1024) = -1024 + (2 \times R) \quad (3)$$

The value of 1024 is chosen because it is a power factor of 2. This is important since only a power factor of 2 can be utilized in arithmetic division operation to synthesize. The value of the weights generated is stored in signal class of data object because an object of this class holds not only the current value of a type but also the past value and the set of scheduled future values that are to appear on that signal. A signal is assigned to a value using a signal assignment statement.

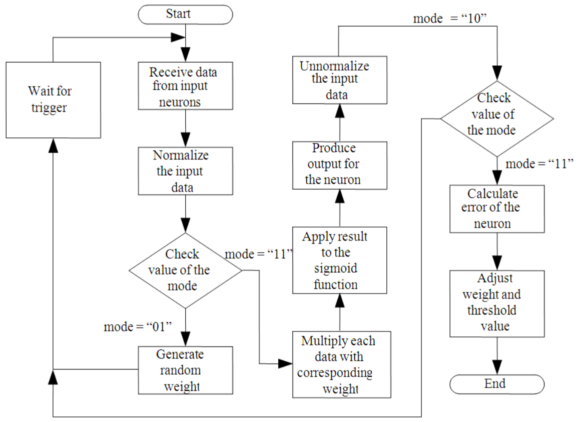

Start
Wait for trigger
Receive data from input neurons
Normalize the input data
Check value of the mode
mode = "01"
Generate random weight
mode = "11"
Multiply each data with corresponding weight
Apply result to the sigmoid function
Produce output for the neuron
Unnormalize the input data
mode = "10"
Check value of the mode
mode = "11"
Calculate error of the neuron
Adjust weight and threshold value
End

Fig. 2: Flowchart of hidden layer neurons

Normalization: Normalization is utilized for reducing the data range set to appropriate value for input to the activation function. To normalize the input data, the floating-point value of the input data, which usually represents the cents value in the data, is first removed. Then, the input data is normalized by using the following function Eq. 4:

$$\text{New value} = (\text{Old value} - \text{Mean}) / (\text{Maximum Range}) \qquad (4)$$

Some modification is applied since the above function produces a new value in the range between -1 and 1; therefore, the new value is multiplied by 2048 to increase the range and thus facilitate synthesize. However, if the maximum range is not a power factor of two then the division of 2048 from the maximum range produces a result that is close to 1. Therefore, the division and multiplication terms are removed and the following function is used Eq. 5:

$$\text{New value} = (\text{Old value} - \text{Mean}) \qquad (5)$$

However, this function produces a suitable range of values for input to the sigmoid activation function. The function to re-scale the output back to its original value is Eq. 6:

$$\text{New value} = (\text{Old value} + \text{Mean}) \qquad (6)$$

Sigmoid Function and threshold value: Sigmoid transfer function works for squashing the neuron output into one and zero. In the network, the sigmoid function piecewise linear approximation is utilized. The output range must be in between 0-1024 as the input range is in between -1024 and 1024, after sigmoid function is applied to the network. As such, equation illustrated in Table 1 is utilized to the design of the VHDL Eq. 7.

Table 1: The set of equation

0	P<-400	y = 150p + 600	1000<p<2000
y=31p + 140	-4000<p<-3000	y = 67p + 750	2000<p<
y=67p + 250	-3000<p<-2000	y = 31p + 860	3000<p<4000
y=150p + 400	-2000<p<-1000	y = 1023	p<4000
y=230p + 500	-1000<p<1000		

Where:

Activation output,

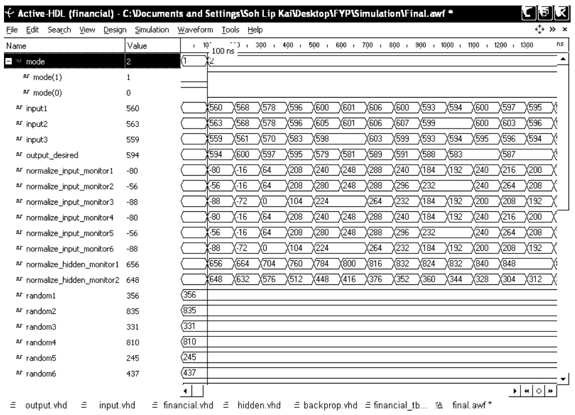
$$p = \left[\left(\sum_{i=1}^{3} \text{weight} \times \text{input}\right) \div 1024\right] + \text{threshold value} \qquad (7)$$

The network bias or threshold value is set to 697 arbitrarily.

Neuron design of output layer: The neural network's output layer comprises of one neuron which utilizes a sigmoid transfer function. OUTPUT is the entity for output layer neuron and BEHAVIOUR is the architecture. As inputs, the output layer neuron has the activation output of each two hidden layer neurons. It has the target value as input also. From hidden layer neurons, each two input data values is multiplied by a unique internal weight for output layer neuron created by the random weights generator. These two results are added collectively. Thereafter, it is applied to internal sigmoid transfer function. The neuron activation is the output. The output layer neuron activation is the activation of entire model. From the data exemplar, the activation is compared with the expected target input to get the current neural network exemplar error. As hidden layer neurons, the same modes of operation are applied to the output layer neurons. The same random normalization method, weights generator and piecewise linear approximation of sigmoid function as hidden layer are utilized in output layer.

RESULTS AND DISCUSSION

By inserting test bench into VHDL code, simulation was accomplished to verify its functionality and performance. Training was done by feeding the historical data of the KLSE Index into the network iteratively for three times. With a period of 100 ns, each integer input signal is fed into the model. The training process was done in between 0 ns and 30000 ns. The process was accomplished in mode "10". For initializing the weights of output layer and hidden layer neurons, first 100 ns were used. In mode "01", this process was accomplished. Testing was accomplished in between 30100 ns and 39900 ns. In mode "11", this process was done. From the KLSE index historical data, only first 100 data was used to show the VHDL code functionality.



Active-HDL (financial) - C:\Documents and Settings\Soh Lip Kai\Desktop\FYP\Simulation\Final.awf *

Name	Value	Waveform (100 ns to 1300 ns)
mode	2	1, 2
mode(1)	1	
mode(0)	0	
input1	560	560, 568, 578, 596, 600, 601, 606, 600, 593, 594, 600, 597, 595
input2	563	563, 568, 578, 596, 605, 601, 606, 607, 599, 600, 603, 596
input3	559	559, 561, 570, 583, 598, 603, 599, 593, 594, 595, 596, 594
output_desired	594	594, 600, 597, 595, 579, 581, 589, 591, 588, 583, 587
normalize_input_monitor1	-80	-80, -16, 64, 208, 240, 248, 288, 240, 184, 192, 240, 216, 200
normalize_input_monitor2	-56	-56, -16, 64, 208, 280, 248, 288, 296, 232, 240, 264, 208
normalize_input_monitor3	-88	-88, -72, 0, 104, 224, 264, 232, 184, 192, 200, 208, 192
normalize_input_monitor4	-80	-80, -16, 64, 208, 240, 248, 288, 240, 184, 192, 240, 216, 200
normalize_input_monitor5	-56	-56, -16, 64, 208, 280, 248, 288, 296, 232, 240, 264, 208
normalize_input_monitor6	-88	-88, -72, 0, 104, 224, 264, 232, 184, 192, 200, 208, 192
normalize_hidden_monitor1	656	656, 664, 704, 760, 784, 800, 816, 832, 824, 832, 840, 848
normalize_hidden_monitor2	648	648, 632, 576, 512, 448, 416, 376, 352, 360, 344, 328, 304, 312
random1	356	356
random2	835	835
random3	331	331
random4	810	810
random5	245	245
random6	437	437

Fig. 3: Sample timing diagram of training simulation

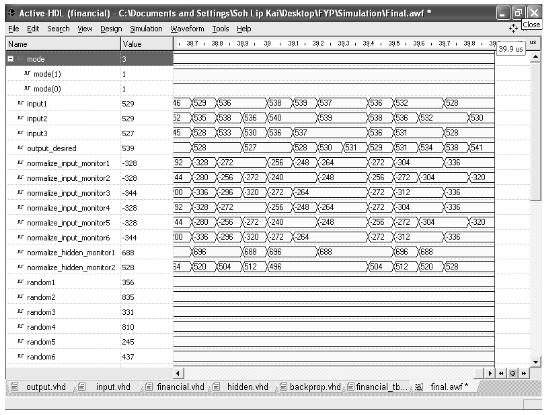
The successive 100 data was utilized to test. The normalization function was adjusted accordingly so that it performs normalization with respect to the specified data only.

Training simulation*:* At weight initializing mode, the neural network chip operated at 0 ns. The random number was created and transformed to weight by using the random weights generator. The neural network chip was in training mode at 100 ns where weights were tuned for reducing output error.

Several sets of weights were tested randomly before getting the simulation result shown in this report. It has been observed that the value of output error is reduced as more input is fed into the network. The weights were adjusted so that the probability of the next input to get its desired output was higher. The input to the network was first normalized before the calculation was performed in the training mode. Figure 3 shows a timing diagram of training simulation.

Testing the simulation: The neural network chip was in testing mode at 30100 ns. A neural network passes by weight initialization process and training before testing was done. Input data was only required during testing. At this period, it was considered that the neural network's weights reached to its local or global minimum. No adjustment of weights during training which implies no output error. As such, the variables stop changing was observed at 30100 ns. Changes in weights were less frequent as the network come close to 30100 ns. Because, after executing the backpropagation learning algorithm, it almost has arrived its local or global minimum.

The simulation halted at 39900 ns after the neural network model was fed with three iterative inputs for training and subsequent 100 inputs for testing.



Active-HDL (financial) - C:\Documents and Settings\Soh Lip Kai\Desktop\FYP\Simulation\Final.awf *

Name	Value	Waveform (38.7 us to 39.9 us)
mode	3	
mode(1)	1	
mode(0)	1	
input1	529	46, 529, 536, 538, 539, 537, 536, 532, 528
input2	529	52, 535, 538, 536, 540, 539, 538, 536, 532, 530
input3	527	45, 528, 533, 530, 536, 537, 536, 531, 528
output_desired	539	528, 527, 528, 530, 531, 529, 531, 534, 538, 541
normalize_input_monitor1	-328	92, -328, -272, -256, -248, -264, -272, -304, -336
normalize_input_monitor2	-328	44, -280, -256, -272, -240, -248, -256, -272, -304, -320
normalize_input_monitor3	-344	00, -336, -296, -320, -272, -264, -272, -312, -336
normalize_input_monitor4	-328	92, -328, -272, -256, -248, -264, -272, -304, -336
normalize_input_monitor5	-328	44, -280, -256, -272, -240, -248, -256, -272, -304, -320
normalize_input_monitor6	-344	00, -336, -296, -320, -272, -264, -272, -312, -336
normalize_hidden_monitor1	688	696, 688, 696, 688, 696, 688
normalize_hidden_monitor2	528	54, 520, 504, 512, 496, 504, 512, 520, 528
random1	356	
random2	835	
random3	331	
random4	810	
random5	245	
random6	437	

Fig. 4: Testing simulation sample timing diagram

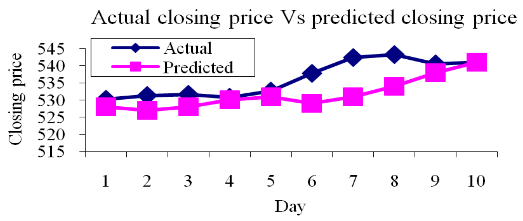


Fig. 5: Actual closing price Vs predicted closing price

Table 2: The chip utilization rate

Device	EPF10K100EQC208-1
f_{max} (MHz)	33.76 MHz
Dedicated input pins	6/6
I/O Pins	57/141
Logic cells	2853/4992
Embedded cells	6/192
EABs	1/12

At 39900 ns, desired_output was obtained for the KLSE Index closing price 10 days in advance of the input price. The reduction in input number to 100 inputs was done to reduce the system complexity. Figure 4 shows the timing diagram of testing simulation. A comparative study between actual closing price and predicted closing price is illustrated in Fig. 5. By comparing, it is concluded that the model prediction is 99.16% precise Eq. 8:

Percentage of accuracy

$$= 100 - \left[\left(\sum_{i=1}^{N} \frac{X - Y}{X} \times 100\right) \div N\right] = 99.16\% \quad (8)$$

Where:

N = No. of days
X = Actual closing price
Y = Predicted closing price

Synthesis: Regarding hardware implementation, the VHDL code is synthesized keeping Altera FLEX10KE: EPF10K100EQC208-1 FPGA chip in consideration, resulting in maximum clock frequency of 33.76 MHz. The FLEX 10KE family offers speed, density and features integrating the full systems, which includes multiple 32-bit buses in a single chip. Table 2 lists utilization rate of the chip.

CONCLUSION

By simulating with KLSE index data the proposed stock market forecasting system based on neural network is successfully designed, implemented and tested on FLEX 10KE FPGA chip. This study test results are anticipated to be a higher rate of prediction for stock market analysis, thereby maintaining the high quality of supplying information in stock market business.

REFERENCES

Akter, M., M.B.I. Reaz, F. Mohd-Yasin and F. Choong, 2008. Hardware implementations of an image compressor for mobile communications. J. Commun. Technol. Elect., 53: 899-910. DOI: 10.1134/S106422690808007X

Choong, F., M.B.I. Reaz and F. Mohd-Yasin, 2005. Power quality disturbance detection using artificial intelligence: A hardware approach. Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Apr. 4-8, IEEE Xplore Press, Denver, USA., pp: 146a-146a. DOI: 10.1109/IPDPS.2005.348

Choong, F., M.B.I. Reaz, T.C. Chin and F. Mohd-Yasin, 2006. Design and implementation of a data compression scheme: A partial matching approach. Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation, Jul. 26-28, IEEE Xplore Press, Sydney, Australia, pp: 150-155. DOI: 10.1109/CGIV.2006.94

Corchado, J., C. Fyfe and B. Lees, 1998. Unsupervised learning for financial forecasting. Proceedings of the IEEE/IAFE/INFORMS 1998 Conference on Computational Intelligence for Financial Engineering (CIFEr), Mar. 29-31, IEEE Xplore Press, New York, USA., pp: 259-263. DOI: 10.1109/CIFER.1998.690316

Coussy, P., D.D. Gajski, M. Meredith and A. Takach, 2009. An introduction to high-level synthesis. IEEE Design Test Comput., 26: 8-17. DOI: 10.1109/MDT.2009.69

Disorntetiwat, P. and C.H. Dagli, 2000. Simple ensemble-averaging model based on generalized regression neural network in financial forecasting problems. Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications and Control Symposium, Oct. 01-04, IEEE Xplore Press, Lake Louise, Canada, pp: 477-480. DOI: 10.1109/ASSPCC.2000.882522

Garliauskas, A., 1999. Neural network chaos and computational algorithms of forecast in Finance. Proceedings of IEEE International Conference on Systems, Man, and Cybernetics, Oct. 12-15, IEEE Xplore Press, Tokyo, Japan, pp: 638-643. DOI: 10.1109/ICSMC.1999.825335

Kane, R. and M. Milgram, 1994. Financial Forecasting and Rules Extraction from Trained Networks. Proceedings of the IEEE International Conference on Neural Networks IEEE World Congress on Computational Intelligence, Jun. 27-Jul. 02, IEEE Xplore Press, Orlando, USA., pp: 3190-3195. DOI: 10.1109/ICNN.1994.374745

Marufuzzaman, M., M.B.I. Reaz, M.S. Rahman and M.A.M. Ali, 2010. Hardware prototyping of an intelligent current dq PI controller for FOC PMSM drive. Proceedings of the IEEE International Conference on Electrical and Computer Engineering, Dec. 18-20, IEEE Xplore Press, Dhaka, Bangladesh, pp: 86-88. DOI: 10.1109/ICELCE.2010.5700559

Mogaki, S., M. Kamada, T. Yonekura, S. Okamoto and Y. Ohtaki *et al*., 2007. Time-stamp service makes real-time gaming cheat-free. Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games, Sep. 19-20, ACM, New York, USA., pp: 135-138. DOI: 10.1145/1326257.1326281

Mohd-Yasin, F., A.L. Tan and M.I. Reaz, 2004. The FPGA prototyping of iris recognition for biometric identification employing neural network. Proceedings of the 16th International Conference on Microelectronics, Dec. 6-8, IEEE Xplore Press, Tunis, Tunesia, pp: 458-461. DOI: 10.1109/ICM.2004.1434697

Obe, O.O. and D.K. Shangodoyin, 2010. Artificial neural network based model for forecasting sugar cane production. J. Comput. Sci., 6: 439-445. DOI: 10.3844/jcssp.2010.439.445

Reaz, M.B.I., M.T. Islam, M.S. Sulaiman, M.A.M. Ali and H. Sarwar, 2003. FPGA realization of multipurpose FIR filter. Proceedings of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies, Aug. 27-29, IEEE Xplore Press, Chengdu, China, pp: 912-915. DOI: 10.1109/PDCAT.2003.1236448

Reaz, M.B.I., F. Mohd-Yasin, M.S. Sulaiman, K.T. Tho and K.H. Yeow, 2004a. Hardware prototyping of boolean function classification schemes for lossless data compression. Proceedings of the 2nd IEEE International Conference on Computational Cybernetics, Aug. 30- Sep. 01, IEEE Xplore Press, Vienna, Austria, pp: 47-51. DOI: 10.1109/ICCCYB.2004.1437664

Reaz, M.B.I., M.S. Sulaiman, F.M. Yasin and T.A. Leng, 2004b. IRIS recognition using neural network based on VHDL prototyping. Proceedings of the IEEE International Conference on Information and Communication Technologies: From Theory to Applications, Apr. 19-23, IEEE Xplore Press, Damascus, Syria, pp: 463-464. DOI: 10.1109/ICTTA.2004.1307832

Reaz, M.B.I., F. Mohd-Yasin, S.L. Tan, H.Y. Tan and M.I. Ibrahimy, 2005a. Partial encryption of compressed images employing FPGA. Proceedings of the IEEE International Symposium on Circuits and Systems, May 23-26, IEEE Xplore Press, Kobe, Japan, pp: 2385-2388. DOI: 10.1109/ISCAS.2005.1465105

Reaz, M.B.I., P.W. Leong, F. Mohd-Yasin and T.C. Chin, 2005b. Modeling of data compression using partial matching: A VHDL approach. Proceedings of the 6th World Wireless Congress (WWC), May 25-27, Palo Alto, USA, pp: 411-416.

Reaz, M.B.I., F. Choong and F. Mohd-Yasin, 2006. VHDL modeling for classification of power quality disturbance employing wavelet transform, artificial neural network and fuzzy logic. Simulation, 82: 867-881. DOI: 10.1177/0037549707077782

Reaz, M.B.I., F. Choong, M.S. Sulaiman and F. Mohd-Yasin, 2007a. Prototyping of wavelet transform, artificial neural network and fuzzy logic for power quality disturbance classifier. J. Elect. Power Components Syst., 35: 1-17. DOI: 10.1080/15325000600815431

Reaz, M.B.I., M.I. Ibrahimy, F. Mohd-Yasin, C.S. Wei and M. Kamada, 2007b. Single core hardware module to implement encryption in TECB mode. Inform. Midem Ljubljana, 37: 165-171.

Seridi-Bouchelaghem, H., T. Sari and M. Sellami, 2005. A neural network for generating adaptive lessons. J. Comput. Sci., 1: 232-243. DOI: 10.3844/jcssp.2005.232.243

Tan, C.N.W. and G.E.A, Wittig, 1993. A study of the parameters of a backpropagation stock price prediction model. Proceedings of the 1st New Zealand International Two Stream Conference on Artificial Neural Networks and Expert Systems, Nov. 24-26, IEEE Xplore Press, Dunedin, New Zealand, pp: 288-291. DOI: 10.1109/ANNES.1993.323023

Zimmermann, H.G., R. Neuneier and R. Grothmann, 2001. Multi-agent Modeling of Multiple FX-Markets by Neural Networks. IEEE Trans. Neural Netw., 12: 735-743. DOI: 10.1109/72.935087