

# A Reconfigurable Online BIST for Combinational Hardware Using Digital Neural Networks

S. Behdad Hosseini, Ali Shahabi, Hassan Sohofi and Zainalabedin Navabi

CAD Laboratory, Electrical and Computer Engineering, School of Engineering Colleges, Campus 2  
University of Tehran, 1450 North Amirabad, 14395-515 Tehran, Iran  
{behdadh, shahabi, h\_sohofi}@cad.ece.ut.ac.ir, navabi@ece.wpi.edu

**Abstract**— Online testing, one of the most challenging issues in design for test domain, is intended for inspection of digital systems behavior during their working period. This paper presents a novel approach for simultaneous online testing of several combinational circuits using a reconfigurable neural network implemented along the original hardware. Automatic generation of the neural network to model the behavior of each design is proposed as well as required techniques to obtain optimum configuration for its hardware realization. Advantages and shortcomings of this approach in terms of area overhead, fault latency and reliability are discussed as well.

**Keywords**— On-line Testing, Self-Checking, Digital Logic Modeling, Digital Neural Networks.

## I. INTRODUCTION

Testing of digital systems becomes more and more challenging due to the great innovation in semiconductor technology [1]. Transistor density growth pace coupled with increasing inclination to System On Chip (SoC) realizations, in order to meet current intricate and high performance system requirements, has increased the probability of fault existence [2]. Faults might be the result of fabrication deficiencies as well as introduced transient and permanent defects during system operation. Crosstalk, reduced voltage levels, cosmic rays and alpha particles which are enumerated as different causes of operation-time faults, are more probable in the state-of-the-art chips because of the current level of compaction and integration [3].

During the life time of a digital system, the correctness of its behavior has to be checked and verified regularly especially for applications which require high availability, *e.g.*, space mission controllers. Furthermore, complex systems are getting harder to test using external test equipments because of limited available number of IO ports. Self-Test is one of the proposed approaches to tackle those issues. In this technique, called Built-In Self Test (BIST), a dedicated hardware is added to the system and some provisions have to be considered in the original design. BISTs are categorized as offline or online.

In offline self-test the test process is performed when the system is in idle state. Since an offline BIST generates its own test inputs, it is not possible to check the functionality of the system when it is working. In contrast, online BIST can be used to check the correctness of the system even in its working period. Online self-tester checks the correct behavior of

corresponding design by constantly monitoring input/output without interrupting Circuit Under Test (CUT). Exertion of online test techniques is crucial in fault sensitive applications [4] where other Design For Test (DFT) techniques, such as Offline-BIST are not feasible due to the impossibility of temporary suspension of CUT. Different approaches for online testing have been proposed. The most challenging problems are the imposed area overhead and generality of the tester. To the best of our knowledge, efficient on-line test techniques (in terms of imposed area overhead) have been proposed and used for specific applications and limited circuit types. In the next section we will review some of those methods which are related and can be compared to our work.

In this paper, we propose a novel method of designing online testers for combinational circuits. In our approach, a design is completely modeled by an artificial neural network (simply Neural Network or NN) using its truth table or logic equations. Our NN architecture permits reconfiguration for modeling different combinational designs (*e.g.*, parts of a complex circuit) by simply adjusting a memory pointer. Here, reconfiguration means that the same hardware can be configured to represent different NNs on the fly. The configured network is implemented in hardware as an online tester along the original circuit. Some special considerations must be taken into account during the hardware implementation phase in order to reduce the complexity and area overhead. A simple comparator will compare the output of the circuit and the expected output obtained from the network. Since a fully combinational and precise NN will consume unacceptable area, we have to incorporate a variety of methods to get a minimal NN capable of modeling the design via approximation and discarding (if possible) unnecessary computations determined by prior simulations. Our NN model can also be used as a powerful verification tool in early stages of CUT design similar to the approach presented in [5].

The rest of this paper is organized as follows. Several previously published techniques on online testing besides the usage of NNs to test digital circuits are given in Section II. Comparisons between some previously published works in this domain and our approach are also given in this section. Section III is dedicated to NNs. Several important aspects on hardware implementation of NNs after a brief introduction are discussed in this section. Our proposed approach to obtain and set up a hardware-implemented NN for a given circuit is

thoroughly explained in Section IV. Section V shows the experimental results, while Section VI concludes the paper and summarizes future works.

## II. RELATED WORKS

The increasing appearance of soft errors usually caused by ionizing radiation in the state-of-the-art chips put another big challenge in the area of hardware testing. In this section, the existing literature on online testing to detect soft errors will be reviewed and compared to our method. Then previous usages of NNs in digital test domain will be presented.

In [6] the authors first emphasized the significant difference between online memory testing [7] and online logic testing [8]. A variety of techniques using redundancy (*e.g.*, duplication, coding, online BIST) were surveyed and a new online error detection approach in logic circuit using logic implication was introduced. Despite the wide usage of logic implications in some application such as logic synthesis optimization, extraction of these implications for large circuit might be complex and time consuming. In our method the behavior of a CUT is used to make an exact model while in [6] they benefit from the internal structure of the CUT.

The authors of [9] have used Viterbi coding to design Finite State Machines (FSMs) to facilitate online error detection. Although, the area overhead of this technique is relatively less than the previously published self-checking FSM structures, it requires re-synthesis of the original circuit and also its applicability is confined to a limited range of digital hardware designs. Yang et al. in [10] have proposed an enhanced logic BIST architecture for online testing. Since for periodic and online testing, importance of stuck-at-open defects that arise due to the wear and tear of the circuit becomes exalted, the authors paid attention to this category of faults. They proposed a new set of tests to detect this kind of faults in addition to a suitable BIST architecture. Considering just open faults, apparently, reduces the generality of this approach. In [11] a concurrent BIST based on the deterministic test set has been proposed to detect permanent faults and prevent their accumulation. Although, their method adjusts a deterministic test set by reducing the number of specified bits to diminish the hardware, the imposed area overhead cannot be neglected and in some cases it exceeds 100%.

Time-Multiplexed Online Checking (TMOS) is introduced in [12] as an effective way to reduce area overhead by dynamically reprogramming a dedicated region of a FPGA for testing another part of the CUT. However, a time lag in producing tester data is inevitable which results in latency to the fault detection process. Our idea to minimize area overhead by increasing delay in the tester is similar to their method.

Artificial neural networks, because of their capability to solve NP-Complete problems, also establish interesting and useful approaches in digital system test domain. In [13], a feed-forward NN trained by resilient error back-propagation has been presented to detect defects in analog integrated circuits. Chakradhar et al. presented a test generation algorithm for single and multiple faults in a specific class of combinational circuits, called (k, k)-circuits, using a bidirectional NN model of the original circuit [14]. In [15] a new automatic test pattern

generation technique, in which stuck at open faults are considered, has been presented by means of NN models. Ortega et al. in [16] has shown a Generalized Hopfield Neural Network (GHN) can be used to design the checking circuitry of a concurrent testable circuit. Finding a solution for the checking circuit of the system, that could be capable of detecting as many error as possible with minimum hardware complexity, is done by mapping the problem into a NN.

## III. NEURAL NETWORKS

This section contains a concise introduction to neural networks. First we explain the fundamentals of NN, Feed-forward network and its learning, problems associated with its hardware implementations, and finally usage of NNs to model a digital hardware.

### A. Preliminaries

Neural networks are non-Von Neumann computational paradigms capable of solving complex non-linear problems. NNs have been successfully used in optimization, control and recognition applications [17]. They are composed of simple PEs called neurons. Each neuron (see Fig. 1.a) accepts a number of inputs and performs a function (usually some form of sigmoid) on summation of all inputs and a bias value. Bias link is a unique value for each neuron that is summed with other inputs of a specific neuron. A directed connection might exist between two neurons, called a link, with an associated value, called weight of that link. Weights are multiplied by each link value (produced from the source neuron) prior to enter as an input to the destination neuron.

A group of neurons is called a layer where there is not any link among them. Their inputs come from previous layer and feed outputs to successive layer (or primary outputs). Feed-forward network is one of the most common topologies of NNs (Fig. 1.b). It consists of one input layer, usually one hidden layer and an output layer. The number of neurons of input and output layer is equal to input signals and output signals, respectively. The computational capacity of feed-forward neural network is dependent on the number of hidden layer neurons [18]. The real computational power of NNs comes from two features: inherent parallelism and the ability to learn.

Learning in NNs is the process of adjusting weights in order to make outputs of the network for a specific input closer to the target output *i.e.*, its expected correct output. Network is being learned using a set of training data (inputs and their targets) prior to its deployment. An epoch is defined as the exhibition of the whole training data once to the network. Usually, many epochs are applied in learning procedure in order to get correct output from the network and converge the network coefficients (*i.e.*, weights and biases) further to their optimum points.

Several learning algorithms have been proposed for different application types and network topologies. Back-propagation is a popular learning algorithm in feed-forward networks. It first computes the errors of outputs to target data for a given input. Then this error value is propagated backward from outputs to input neurons considering relative contribution of each neuron to the final error. The calculated error for each

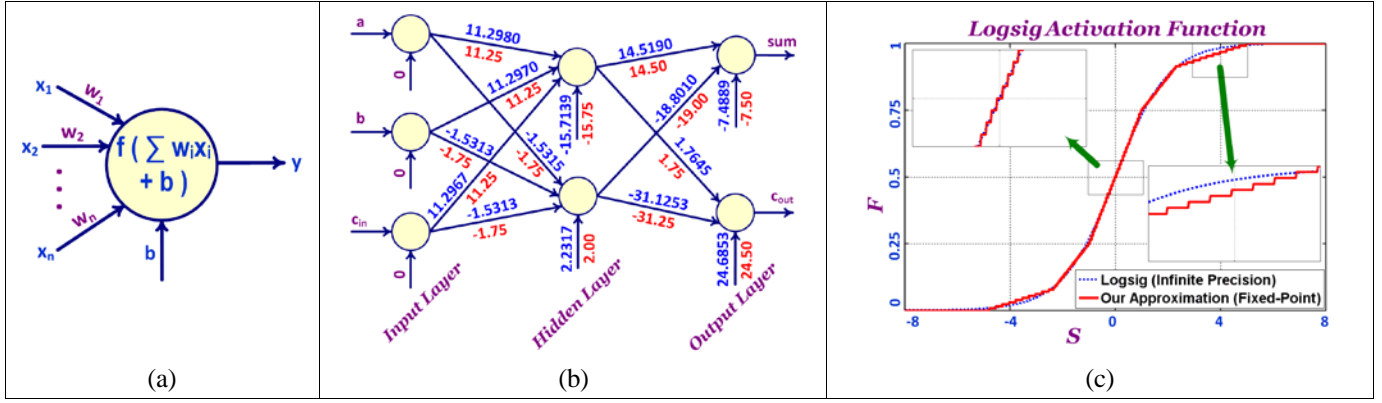


Figure 1. (a) A typical neuron (b) A Feed-forward neural network to model a full-adder (c) Logsig Activation Function (AF)

weight is then used to tune the weight (by using a learning rate constant) to make it closer to optimal value. Complete steps of Back-propagation algorithm and its variations can be found in any neural network text like [19].

### B. Hardware Implementation Issues

From hardware point of view in custom NN implementation, a neuron consists of three components: multiplier, accumulative adder and a non-linear activation function (AF). Since these computational circuits consume a huge amount of hardware, the main problem is how to reduce the area required for a NN. Also area varies greatly by using whether floating-point or fixed-point representations. If fixed-point suffices for an application, including our case, as we will show in Section IV, it has great area benefits over floating-point realizations [20]. The main issue in fixed-point representation is how to decide bit-width for different arithmetic units (See Section IV.B).

Fully combinational realization of a typical NN is possible by allocating required hardware to each neuron and hard-wiring weights and biases as multiplier coefficients. If a given neuron has  $n$  inputs, it requires  $n$  multipliers, an  $(n+1)$ -operand adder, and a dedicated AF for that neuron. This leads to an unreasonable amount of hardware overhead for a testing architecture. In contrast, sequential realization of NNs, uses time-multiplexing technique to get the result of network by using just one neuron. In this method, in addition to a controller, a ROM and a RAM are needed to store weights and biases and to keep intermediate values, respectively. The imposed area can be further reduced by using a sequential multiplier.

In our approach, a reconfigurable NN is needed to model different NNs (different input, hidden, output neurons and weights). Fully combinational NN is not suitable because its topology is fixed. Therefore our first problem is automatically tackled by using the sequential alternative. The major drawback of this approach is the delay between applying the inputs and getting the outputs. It will result in missing some CUT inputs/outputs. Nevertheless, as stated before, online testing is applied to circuits during their working hours, thus, as will be shown later, it is probable that missed faults due to the imposed lag, be caught later. Our idea to minimize area

overhead by increasing delay in the tester is similar to the method proposed in [12]. In their approach, Time-Multiplexed Online Checking (TMOS), they reduce area overhead by increasing fault detection latency by reprogramming the tester area of FPGA for testing another part of circuit.

The activation function, which is used in our NN trainings, is Logsig and is defined as:

$$F = \frac{1}{1 + e^{-S}} \quad (1)$$

Fig. 1.c shows the exact values (infinite precision) of  $F$  for different inputs in range  $-8$  to  $8$ . Since exponential and division are expensive in terms of area and time, other methods of approximation should be utilized. In [21] an extensive review on different methods of hardware implementation of Logsig AF is presented. Among them, Piecewise Linear Approximation of a Nonlinear Function (PLAN) [22] is chosen to be used as our AF because of its acceptable error factors and its efficient hardware implementation [21]. Fig. 1.c also depicts the outputs of our PLAN implementation with seven bits for its fractional part.

### C. Hardware Modeling Using Neural Networks

The first step in modeling a digital hardware using NNs is to establish a relation between inputs/outputs of hardware and NN. Although our modeling approach is similar to the models presented in [5], [15] and [16], our proposed mechanism to obtain an optimal networks differs. In [23], the authors have proved that for any arbitrary combinational digital circuit, a neural network that is capable of modeling the circuit exists.

Our NNs are Feed-forward networks with one input, one hidden and one output layer. For a given circuit, assume we have  $n$  bits of inputs and  $m$  bits of outputs. Our NN will have  $n$  lines of inputs, each of which connected to one neuron. Therefore input layer has  $n$  neurons. Because inputs of NNs are supposed to be real values, each input is padded with zeros in fractional part to form a fixed-point representation of '0' and '1'. Minimum number of required neurons in hidden layer is obtained by an algorithm explained in the next section. Output layer will contain  $m$  neurons, and hence,  $m$  output bits. Outputs of neurons on output layer must be rounded to obtain one bit



contains a small RAM, a ROM, one saturate adder, one sequential multiplier, one Logsig AF, and some logic for holding IO and validity checking of the output. All values through the data-path are in fixed-point representation.

When an input value is selected for testing, it is loaded into a circular shift register. For neurons on the hidden layer, each bit of the input value in this register should be shifted out and multiplied by the corresponding weight value located in the ROM rows. These intermediate results will be stored in RAM. For neurons on the output layer, the results of the hidden layer from RAM are multiplied by the corresponding weights from the ROM. The multiplier is sequential and activates a ready signal when the result is ready. After the truncation of the multiplication result, this value is accumulated into a register using a saturation adder. When all input values according to the structure of a neuron are multiplied and accumulated in the register, the result must be confined by passing through the AF module. If the neuron is related to the hidden layer, the output of the activation function should be written to the RAM rows for later use. If the neuron is an output layer's neuron, the two most significant bits of the activation function's output (according to the fixed-point representation) should be checked to see if the output value is zero or one. Obtained bit value then compared to the related output bit value of the CUT which is shifted out from a shift register at the output. If both bits are the same, the output of the circuit is fine; otherwise an error signal is activated to indicate that the CUT is faulty.

Controller is responsible to issue appropriate control signals for proper order of computations. For making the controller as simple as possible, we have arranged biases and weights in a special sequential manner and also adjusted order of neuron computations to avoid random access and address calculations to ROM and RAM. Data required for calculating the output of the first hidden neuron is first stored in ROM, followed by data of the second hidden neuron and so on. After hidden neurons, coefficients of output neurons are kept in a same manner. In this case, if we calculate output one by one and in order, the RAM is also accessed sequentially.

#### D. Reconfiguring Hardware NN

In order to make the same hardware usable for online testing of different CUTs, the required bit-width for different parts of the data-path is re-calculated by maximizing

corresponding bits in all CUTs. The final NN is also made reconfigurable by storing weights and biases of all previously obtained bit-true NN models in a single ROM. A pointer to the start location of coefficients of the current CUT, in addition to the number of inputs, hidden, and outputs neurons are also stored in a separate controller ROM. The reconfigurable hardware needs a final step to switch into another configuration, on the fly, by appropriate multiplexing of CUTs inputs/outputs. We implemented the round-robin policy for switching (and testing) between CUTs. By adjusting a constant value stored in the controller, the number of inputs/outputs that should be tested for a CUT before switching to the next one can be changed dynamically.

## V. EXPERIMENTAL RESULTS

To assess the proposed technique, we have considered a variety of combinational circuits. For each circuit an online BIST architecture based on presented approach is generated. Finally to show the effectiveness of our method, all of them are combined and tested in a separate reconfigurable NN.

The circuits chosen to be used as benchmarks and the extracted result are reported in TABLE I. Each row shows the obtained data for each benchmark. The first group of columns denotes each benchmark specification (*i.e.*, name, short description, the number of input and output ports). The next two columns show the number of hidden neurons and epochs (see Section III.A), respectively. Required bits for the different parts of our architecture with respect to the Fig. 2 are given in the third group of columns in TABLE I. We use the notation of  $sx.y$  to show signed fixed-point format that uses  $x$  bits for the integer part and  $y$  bits for the fractional part. Likewise  $ux.y$  is used to denote an unsigned fixed-point format. The required bit-width for RAM, ROM and multiplier output are extracted and reported in column 6 to 8 of the table, respectively.

Finally, the last group of columns provides area and delay measurements to facilitate evaluation of the proposed approach. Each benchmark with the extracted BIST architecture is modeled using 'VHDL' and 'Verilog' hardware description languages and then, synthesized on an ASIC library. The imposed area overhead of the BIST architecture is also reported. Mean Fault Detection Latency (MFDL) is used as a quality of test factor. It defines the mean time between appearance of a fault and its detection by online tester. For a

TABLE I. EXPERIMENTAL RESULT

Benchmark Circuits			Neural Network Detail		Data-Path Bit Widths			Extracted Measures		
Name	Description	# of In/Out	# of Hidden Neurons	# of Epochs	r (RAM)	p (ROM)	s (Mult)	CUT Area	BIST Area	MFDL (Clock Cycles)
C17	ISCAS Benchmark	5/2	3	200	u1.2	s5.0	s5.0	7	80	221
74182	4-bit Lookahead carry Generator	9/5	5	400	u1.4	s5.4	s5.3	14	125	1417
74283	4-bit Binary Full Adder	9/5	8	500	u1.5	s6.6	s6.6	21	146	3121
7485	4-bit Comprator	11/3	3	300	u1.4	s6.4	s6.4	29	131	1955
Div4	4-bit Divider	8/8	43	1000	u1.7	s6.7	s6.7	191	207	20633
RNN	BIST of Above Circuits	11/8	62	N/A	u1.7	s6.7	s6.7	262	251	31432

given CUT, all of the stuck-at faults have been injected one by one and the time (in clock cycles) between its injection and detection is measured. MFDL is the mean of above calculated times. Input patterns are generated in a random manner. Note that faults which are impossible to detect were excluded from the fault lists. Because no test pattern exists to detect this category of faults, it is impractical for an online BIST to recognize them without access to the internal parts of the CUT, for example by means of inserting some controllability and accessibility points.

It can be deduced from TABLE I that in spite of the unacceptable area overhead for each CUT, the overall overhead is reasonable when all of them is combined. It can be explained by the fact that the area of our hardware remains almost intact, especially for the same number of data-path bit-widths. In other words, area for different CUTs and for final reconfigurable on-line BIST does not vary greatly.

## VI. CONCLUSIONS

Artificial neural networks can be utilized to model any arbitrary combinational hardware defined via its truth-table or logic equations. In this paper, we presented a novel online BIST using a reconfigurable neural network. Our architecture permits switching to other configurations to model and test different CUTs on the fly. Similarities among computations in NNs are employed to reduce imposed area overhead in expense of allowing more delay to produce expected output. An algorithm to obtain the bit-true NN model of each circuit and a reconfigurable NN architecture was also presented. Several measures to assess the proposed method were introduced and reported for a number of combinational circuits.

The idea can be extended in different aspects. Other NN topologies like Hopfield and SOMs are good candidates to be used instead of Feed-forward networks. Bit-true extraction of a precise network can be made more hardware-aware by considering how much area will be reduced by bit savings in different parts. Some improvements can be applicable to specific circuits. For instance, in order to gain further reduction in area in expense of lower test qualities, only some of the output bits can be learned and tested, which is especially convenient in arithmetic circuits where MSBs are more important than LSBs.

## REFERENCES

[1] Semiconductor Industry Association, International Technology Roadmap for Semiconductors, Edition 2008, available: <http://www.itrs.net/reports.html>.

[2] Y. Zorian, S. Dey, and M. J. Rodgers, "Test of Future System-on-Chips," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD'00)*, Nov. 2000, pp. 392-398.

[3] C. Metra, A. Pagano, and B. Riccò, "On-Line Testing of Transient and Crosstalk Faults Affecting Interconnections of FPGA Implemented Systems," in *Proc. IEEE Int. Test Conf.*, 2001, pp. 939-947.

[4] A. Steininger, and C. Scherrer, "On the Necessity of On-line-BIST in Safety-Critical Applications – A Case-Study," in *Proc. 29<sup>th</sup> Annual Int. Symp. on Fault-Tolerant Computing (FTCS'99)*, 1999, pp. 208-215.

[5] R. Raelis, "Modeling and Verification of Digital Logic Circuit Using Neural Networks," *American Society for Engineering Education (ASEE) IL/IN Sectional Conference*, Session B-T2-3, Apr. 2005.

[6] K. Nepal, N. Alves, J. Dworak, and R. I. Bahar, "Using implications for online error detection," in *Proc. International Test Conference (ITC'08)*, 2008, page 24.2.

[7] M. Choi, N. Park, F. Lombardi, Y.B. Kim, and V. Piuri, "Balanced redundancy utilization in embedded memory cores for dependable systems," in *Proc. 17<sup>th</sup> IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'02)*, 2002, pp. 419-427.

[8] P. Drineas and Y. Makris, "Concurrent fault detection in random combinational logic," in *Proc. 4<sup>th</sup> International Symposium on Quality Electronic Design (ISQED'03)*, 2003, pp. 425-430.

[9] M. Li, S. Xu, E. Xia, and F. Wan, "Design of FSM with Concurrent Error Detection Based on Viterbi Decoding," in *Proc. 17<sup>th</sup> Asian Test Symposium (ATS'08)*, 2008, pp. 383-388.

[10] F. Yang et al., "An Enhanced Logic BIST Architecture for Online Testing," in *Proc. 14<sup>th</sup> On-Line Testing Symposium (IOLTS'8)*, Jul. 2008, pp. 10-15.

[11] M. A. Kochte et al., "Concurrent Self-Test with Partially Specified Patterns for Low Test Latency and Overhead," in *Proc. 14<sup>th</sup> IEEE European Test Symp. (ETS'2009)*, May 2009, pp. 33-58.

[12] M. Gao, H. Chang, P. Lisherness and K. Cheng, "Time-Multiplexed Online Checking: A Feasibility Study", in *Proc. Asian Test Symposium (ATS'08)*, 2008, pp. 371-376.

[13] V. Stopjakova, P. Malosek, D. Micusik, M. Matej, and M. Margala, "Classification of Defective Analog Integrated Circuits Using Artificial Neural Networks," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 20, no. 1, pp. 25-37, 2004.

[14] S. T. Chakradhar, V. D. Agrawal, and M. L. Bushnell, "Energy Minimization and Design for Testability," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 5, no. 1, pp. 57-66, 1994.

[15] Z. Zhang, R. D. McLeod, and W. Pedrycz, "A Neural Network Algorithm for Testing Stuck-open Faults in CMOS Combinational Circuits," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 4, no. 3, pp. 225-235, 1993.

[16] J. Ortega, A. Prieto, A. Lloris, and F.J. Pelayo, "Generalized Hopfield Neural Network for Concurrent Testing," *IEEE Transactions on Computers*, vol. 42, no. 8, pp. 898-912, 1993.

[17] M. Ananda Ro and J. Srinivas, *Neural Networks Algorithms and Applications*, Alpha Science International Ltd., Part III, pp.157, 2003.

[18] Hopfield J.J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," in *Proc. National Academy of Sciences USA*, vol. 79, 1982, pp. 2554-8.

[19] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd Ed., Prentice-Hall, 1999.

[20] M. Moussa, S. Areibi, and K. Nichols, "On the Arithmetic Precision for Implementing Back-Propagation Networks on FPGA: A Case Study", Book Chapter in *FPGA Implementations of Neural Networks*, Springer, The Netherlands, pp. 37-61, 2006.

[21] M.T. Tommiska, "Efficient Digital Implementation of the Sigmoid Function for Reprogrammable Logic," *IEE Proc. Computers and Digital Techniques*, vol. 150, no. 6, pp. 403-411, Nov. 2003.

[22] H. Amin, K. M. Curtis, and B. R. Hayes-Gill, "Piecewise Linear Approximation Applied to Nonlinear Function of a Neural Network," *IEE Proc. Circuits, Devices and Systems*, vol. 144, no. 6, pp. 313-317, Dec. 1997.

[23] P. Zhongliang, C. Ling, L. Shouqiang, and Z. Guangzhao, "Neural Network Approach for Multiple Fault Test of Digital Circuit", in *Proc. 6<sup>th</sup> Int. Conf. on Intelligent Systems Design and Applications (ISDA'06)*, 2006, pp. 24-29.