Hindawi Publishing Corporation
Chinese Journal of Engineering
Volume 2014, Article ID 924927, 11 pages
http://dx.doi.org/10.1155/2014/924927

Research Article

Hardware Neural Networks Modeling for Computing Different Performance Parameters of Rectangular, Circular, and Triangular Microstrip Antennas

Taimoor Khan and Asok De

Department of Electronics and Communication Engineering, National Institute of Technology, Patna 800005, India

Correspondence should be addressed to Taimoor Khan; ktaimoor@gmail.com

Received 11 December 2013; Accepted 16 January 2014; Published 26 February 2014

Academic Editors: J. Deng, Z. Ji, and L. Li

In the last one decade, neural networks-based modeling has been used for computing different performance parameters of microstrip antennas because of learning and generalization features. Most of the created neural models are based on software simulation. As the neural networks show massive parallelism inherently, a parallel hardware needs to be created for creating faster computing machine by taking the advantages of the parallelism of the neural networks. This paper demonstrates a generalized neural networks model created on field programmable gate array- (FPGA-) based reconfigurable hardware platform for computing different performance parameters of microstrip antennas. Thus, the proposed approach provides a platform for developing low-cost neural network-based FPGA simulators for microwave applications. Also, the results obtained by this approach are in very good agreement with the measured results available in the literature.

1. Introduction

Low profile, conformable to planar and nonplanar surfaces, most economical, mechanically robust, light weight, and easily mount-ability are the key advantages of microstrip antennas (MSAs). Because of these add-on advantages, the microstrip antennas are widely used in many communication applications. Since the microstrip antenna operates only in the vicinity of the resonant frequency, it needs to be calculated accurately for analyzing the microstrip antennas. Similarly, for designing the microstrip antennas, the physical dimension(s) must also be calculated precisely [1]. There are two conventional ways for analyzing and/or designing the microstrip antennas, analytical methods and numerical methods. The analytical methods provide a good spontaneous explanation for the operation of microstrip antennas. As the analytical methods are based on the physical assumptions for simplifying the radiation mechanism of the microstrip antennas, these methods are not suitable for many structures, where the thickness of the substrate is not very thin. The numerical methods also provide the accurate results but the analysis using these methods leads to the expressions as an integral equation. The choice of test functions and path integrations appears to be more critical without any initial assumption in the final stage of the numerical results. Also, these methods require a new solution for any sort of alteration in the geometry. The problems associated with these conventional methods can be overcome by selecting the appropriate neural network methods [1]. In recent years, artificial neural networks (ANNs) have acquired tremendous utilization in microwave communications because of high classifying power and learning ability of the ANN [2–4]. The ANN model is trained using measured, calculated, and/or simulated patterns. Once the model is trained for a specified error, it returns the results for every infinitesimal change in the input patterns within a fraction of a second. Several neural models are available in the literature [5–8] for computing single performance parameter (resonance frequency or geometric dimensions) of the microstrip patch antennas, respectively. But having individual neural model for each performance parameter becomes sometimes unattractive to include in modern antenna computer-aided-design (CAD) programs. It has recently been overcome by introducing

the concept of generalized neural approach for computing different performance parameters, simultaneously [9–12]. The resonance frequencies of rectangular, circular, and triangular MSAs have been computed [9–11] using generalized neural networks model. The resonance frequencies and the physical dimensions of the rectangular MSAs have been computed using generalized neural networks model [12]. Hence, the generalized neural models [9–12] have been used only for computing three different parameters of MSAs, respectively. Recently, authors have also proposed more accurate and very simple generalized neural networks approaches for computing two parameters [13–15], three parameters [16, 17], four parameters [18], and seven parameters [19], respectively.

The neural networks models [2–19] have been realized using software simulation. Since the software simulation involves large number of complex arithmetical operations, therefore it does not have the desired performance as in case of real-time computation [20]. For speeding-up the computing process, some sort of hardware needs to be created for training and/or testing algorithm of the neural networks. Replication of neural networks on a dedicated hardware is a challenging task [20]. There are two conventional methods for implementing the algorithm(s) of a neural model on a hardware platform, either by using the hardwired technology or reconfigurable hardware approach. Further, there are two ways for using the hardwired technology, either by creating the application specific integrated circuits (ASICs) or by using a group of individual components using printed circuit board (PCB) technology. ASICs are very fast and efficient but they cannot be altered after fabrication. This forces a redesign and a refabrication of the chip for the requirement of any kind of modification and, therefore, becomes an expensive process. PCB-based approach is also inflexible, as it requires redesign and the replacement of the board in the event of changes in the application [21]. Presently, reconfigurable hardware is being widely used as a technique for projecting and prototyping the hardware because it allows fast designing and prototyping [22]. Reconfigurable hardware platforms have a balance between hardware and software solutions exclusively, as they have the programmability of software with performance capacity approaching that of a custom hardware implementation [22]. Because of several attractive features like low cost as compared to MPLDs (mask programmable logic devices), easy implementation and reprogrammability, the FPGAs are being widely used for creating reconfigurable hardware of neural networks for different applications [23–27] but the literature on this approach for microwave applications domain is limited [28–31]. To et al. [28] have described prototyping of a neuroadaptive smart antenna beam-forming algorithm using hardware-software approach by implementing the RBF neural network on FPGA platform. For prototyping strategy, they have used three steps, implementing a simulation model in MATLAB software, translating it into generic C/C++ working model with an external matrix arithmetic library, and, finally, implementing this C/C++ model on the Altera "APEX FPGA EP20K200E" embedded processor platform [28]. Al Zuraiqi et al. [29] have designed a neural network FPGA controller for reconfigurable antennas and the optimized weights, biases, and network topology have been configured on Xilinx's FPGA. Recently, Ghayoula et al. [30] have explored prototyping and implementing concept of neural networks on FPGA platform for designing phased antenna array. In this purpose, they have optimized the neural model with 17 neurons in the input layer and 8 neurons in the output layer. They have then implemented the optimized model with 8-bit precision using Xilinx 8.1i, simulated with ModelSim SE 6.0, and downloaded and tested on Xilinx "XC3S500". Fournier et al. [31] have implemented the neural networks model on FPGA platform for beam steering of an array of four microstrip patch antennas. The feeding structure of Butler matrix has been optimized using the momentum feature of advance design system (ADS) for 2.4 GHz frequency. For controlling the beam angle, they have performed the training of the ANN model in MATLAB Simulink; coding for the trained model has been created using Xilinx system generator and, finally, it is downloaded on Xilinx' FPGA board [31]. The beauty of the present work lies in creating an FPGA-based reconfigurable hardware for a generalized neural model which is capable of computing seven different performance parameters. The training and/or testing algorithms of the neural networks can be implemented in reconfigurable hardware. In the proposed work, training of the model is done offline in personal computing machine and the testing algorithm is implemented on Xilinx's FPGA board, XC3S500E.

2. Generation of Patterns

A microstrip antenna, in its simplest configuration, consists of a radiating conductive patch on one side of a dielectric substrate of relative permittivity "ε_r" and of thickness "h" having a ground plane on the other side [1]. Figure 1 illustrates the geometry for three different shapes of microstrip antennas, that is, rectangular microstrip antenna (RMSA), circular microstrip antenna (CMSA), and triangular microstrip antenna (TMSA), respectively. The side view of three different patches is shown in Figure 1(a), whereas top views for RMSA, CMSA, and TMSA are mentioned in Figures 1(b)–1(d), respectively.

For analyzing a microstrip antenna (MSA), the resonance frequency for these geometries can be easily calculated, if the physical dimensions, relative permittivity, dielectric thickness, and mode of propagation are given [32–44]. Total 81 measured patterns (46 for RMSA, 20 for CMSA, and 15 for TMSA) are taken as training and testing patterns for the neural networks modeling. The 46 patterns for RMSAs, 20 patterns for CMSAs, and 15 patterns for TMSAs are mentioned in Tables 1, 2, and 3, respectively.

For designing the MSAs, the physical dimensions of the patch can be determined if the resonance frequency, thickness, and relative permittivity of the substrate and mode of propagation are given [32–44]. This is mentioned in Table 4. The total of 92 patterns (46 for each dimension) for designing the RMSAs can be created using Tables 4 and 1, simultaneously. Similarly, total of 20 patterns for designing the RMSAs can be created using Tables 4 and 2, simultaneously. Similar approach is used for creating 15 patterns

Table 1: Patterns for analyzing RMSAs [32–35].

W_r (cm)	L_r (cm)	h (cm)	ε_r	p	f_r (GHz)
5.7000	3.8000	0.3175	2.3300	1	2.3100
4.5500	3.0500	0.3175	2.3300	1	2.8900
2.9500	1.9500	0.3175	2.3300	1	4.2400
1.9500	1.3000	0.3175	2.3300	1	5.8400
1.7000	1.1000	0.3175	2.3300	1	6.8000
1.4000	0.9000	0.3175	2.3300	1	7.7000
1.2000	0.8000	0.3175	2.3300	1	8.2700
1.0500	0.7000	0.3175	2.3300	1	9.1400
1.7000	1.1000	0.9525	2.3300	1	4.7300
1.7000	1.1000	0.1524	2.3300	1	7.8700
4.1000	4.1400	0.1524	2.5000	1	2.2280
6.8580	4.1400	0.1524	2.5000	1	2.2000
10.800	4.1400	0.1524	2.5000	1	2.1810
0.8500	1.2900	0.0170	2.2200	1	7.7400
0.7900	1.1850	0.0170	2.2200	1	8.4500
2.0000	2.5000	0.0790	2.2200	1	3.9700
1.0630	1.1830	0.0790	2.5500	1	7.7300
0.9100	1.0000	0.1270	10.200	1	4.6000
1.7200	1.8600	0.1570	2.3300	1	5.0600
1.8100	1.9600	0.1570	2.3300	1	4.8050
1.2700	1.3500	0.1630	2.5500	1	6.5600
1.5000	1.6210	0.1630	2.5500	1	5.6000
1.3370	1.4120	0.2000	2.5500	1	6.2000
1.1200	1.2000	0.2420	2.5500	1	7.0500
1.4030	1.4850	0.2520	2.5500	1	5.8000
1.5300	1.6300	0.3000	2.5000	1	5.2700
0.9050	1.0180	0.3000	2.5000	1	7.9900
1.1700	1.2800	0.3000	2.5000	1	6.5700
1.3750	1.5800	0.4760	2.5500	1	5.1000
0.7760	1.0800	0.3300	2.5500	1	8.0000
0.7900	1.2550	0.4000	2.5500	1	7.1340
0.9870	1.4500	0.4500	2.5500	1	6.0700
1.0000	1.5200	0.4760	2.5500	1	5.8200
0.8140	1.4400	0.4760	2.5500	1	6.3800
0.7900	1.6200	0.5500	2.5500	1	5.9900
1.2000	1.9700	0.6260	2.5500	1	4.6600
0.7830	2.3000	0.8540	2.5500	1	4.6000
1.2560	2.7560	0.9520	2.5500	1	3.5800
0.9740	2.6200	0.9520	2.5500	1	3.9800
1.0200	2.6400	0.9520	2.5500	1	3.9000
0.8830	2.6760	1.0000	2.5500	1	3.9800
0.7770	2.8350	1.1000	2.5500	1	3.9000
0.9200	3.1300	1.2000	2.5500	1	3.4700
1.0300	3.3800	1.2810	2.5500	1	3.2000
1.2650	3.5000	1.2810	2.5500	1	2.9800
1.0800	3.4000	1.2810	2.5500	1	3.1500

p means $m = 1$ and $n = 0$.

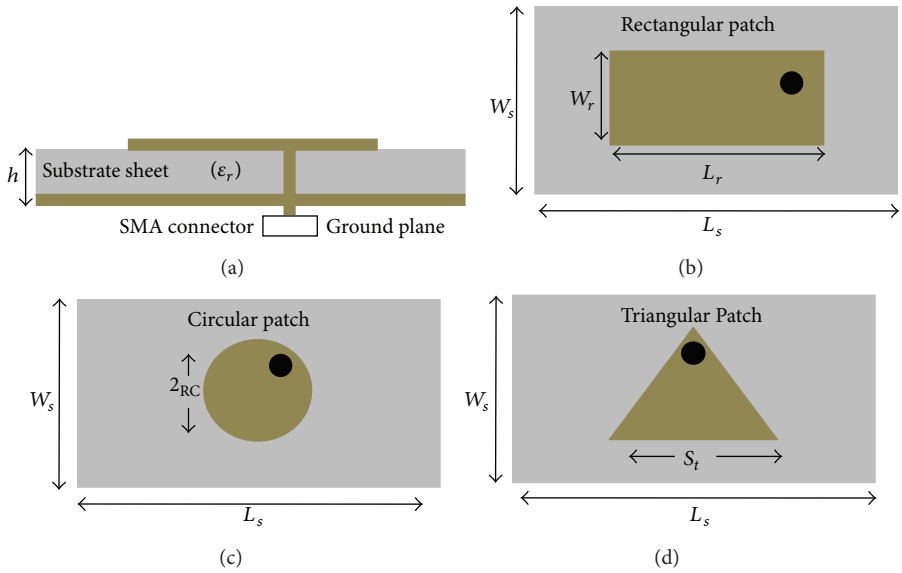

Figure 1: Geometries of MSAs for patterns generation.

Table 2: Patterns for analyzing CMSA [36–42].

R_c (cm)	h (cm)	ε_r	m	n	f_c (GHz)
0.7400	0.1588	2.6500	1	1	6.6340
0.7700	0.2350	4.5500	1	1	4.9450
0.8200	0.1588	2.6500	1	1	6.0740
0.9600	0.1588	2.6500	1	1	5.2240
1.0400	0.2350	4.5500	1	1	3.7500
1.0700	0.1588	2.6500	1	1	4.7230
1.1500	0.1588	2.6500	1	1	4.4250
1.2700	0.0794	2.5900	1	1	4.0700
2.0000	0.2350	4.5500	1	1	2.0030
2.9900	0.2350	4.5500	1	1	1.3600
3.4930	0.1588	2.5000	1	1	1.5700
3.4930	0.3175	2.5000	1	1	1.5100
3.8000	0.1524	2.4900	1	1	1.4430
3.9750	0.2350	4.5500	1	1	1.0300
4.8500	0.3180	2.5200	1	1	1.0990
4.9500	0.2350	4.5500	1	1	0.8250
5.0000	0.1590	2.3200	1	1	1.1280
6.8000	0.0800	2.3200	1	1	0.8350
6.8000	0.1590	2.3200	1	1	0.8290
6.8000	0.3180	2.3200	1	1	0.8150

Table 3: Patterns for analyzing TMSA [43, 44].

S_t (cm)	h (cm)	ε_r	m	n	f_t (GHz)
4.1000	0.0700	10.500	1	0	1.5190
4.1000	0.0700	10.500	1	1	2.6370
4.1000	0.0700	10.500	2	0	2.9950
4.1000	0.0700	10.500	2	1	3.9730
4.1000	0.0700	10.500	3	0	4.4390
8.7000	0.0780	2.3200	1	0	1.4890
8.7000	0.0780	2.3200	1	1	2.5960
8.7000	0.0780	2.3200	2	0	2.9690
8.7000	0.0780	2.3200	2	1	3.9680
8.7000	0.0780	2.3200	3	0	4.4430
10.000	0.1590	2.3200	1	0	1.2800
10.000	0.1590	2.3200	1	1	2.2420
10.000	0.1590	2.3200	2	0	2.5500
10.000	0.1590	2.3200	2	1	3.4000
10.000	0.1590	2.3200	3	0	3.8240

for designing TMSAs using Tables 4 and 3, simultaneously. These total 208 generated patterns (81 analyzing patterns + 127 designing patterns) are used for software implementation of training algorithm and hardware implementation of testing algorithm to be discussed in Sections 3 and 4, respectively.

3. Proposed Neural Networks Modeling

A radial basis function (RBF) neural network consists of three-layer feed-forward neural network with entirely different roles. The input layer is made up of source nodes which connect the network to its outside environment. The input layer does not accomplish any process but simply buffers the data. The second layer, that is, the hidden layer, applies a multivariate Gaussian nonlinear transformation from the input space to the hidden space [45]. The output layer is linear, supplying the response of the network to the patterns applied to the input layer. As far as learning (or training) of the neural network is concerned, the RBF neural network is much faster than the multilayered perceptron (MLP) neural network. It is so because the learning process in RBF neural network has two stages and both of the stages are made more efficient by using appropriate learning algorithm. This is the prime reason of using RBF neural network instead of MLP neural network in the present work. 208 measured patterns discussed in Section 2 are used for training and testing of RBF neural network. The strategy for implementing the training

Table 4: Designing of RMSAs, CMSAs, and TMSAs.

Designing of RMSAs-1 [32–35]					
Inputs					Output
f_r (GHz)	h (cm)	ε_r	m	n	w_r (cm)
Designing of RMSAs-2 [32–35]					
Inputs					Output
f_r (GHz)	h (cm)	ε_r	m	n	L_r (cm)
Designing of CMSAs [36–42]					
Inputs					Output
f_c (GHz)	h (cm)	ε_r	m	n	R_c
Designing of TMSAs [43, 44]					
Inputs					Output
f_t (GHz)	h (cm)	ε_r	m	n	S_t

algorithm for RBF neural networks model is being discussed in this section, whereas hardware implementation of testing algorithm is to be discussed in Section 4.

3.1. Training Algorithm: Software Implementation. The input-output patterns for analyzing and designing the RMSAs, CMSAs, and TMSAs are discussed in Section 2 and mentioned in Tables 1, 2, 3, and 4. It is clear from these tables that the input pattern in all cases is five-dimensional, which is, in general, being represented by five variables x_1,x_2, x_3, x_4, and x_5, respectively. Seven different cases, computation of resonance frequency, width and length of RMSAs, computation of resonance frequency and radius of CMSAs, and the computation of resonance frequency and side-length are being included in a single RBF neural network model. To distinguish these different cases, an additional variable, "M", is included in the existing five-dimensional patterns. Here, M = 1, 2, and 3 represented the computation of resonance frequency, width, and length of RMSAs, M = 4 and 5 represented the resonance frequency and radius of CMSAs, and, finally, M = 6 and 7 indicateed the resonance frequency and side length of the TMSAs, respectively. This six-dimensional input pattern, that is, $[x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad M]$, is used as the training pattern and testing pattern for computing each performance parameter of the microstrip antennas, respectively.

An RBF neural network model with six input nodes, 45 hidden nodes, and single output node is illustrated in Figure 2, in which weight matrices at hidden nodes and at output node are designated as $[w]_{6\times45}$ and $[t]_{1\times45,}$ whereas the bias matrices at these nodes are represented as $[b]_{45\times1}$ and $[s]_{1\times1}$, respectively.

Initially, some random values are assigned for the weight and bias values and these values are optimized using Levenberg-Marquardt (LM) training algorithm [46] and the codes for implementing the training algorithm of the proposed model is written using MATLAB software on a personal computing machine with system configuration, Dell Optiplex 780 Core 2 Duo CPU E8400, 3.0 GHz with 4.0 GB RAM. The basic approach for training the model can be understood by the flow diagram shown in Figure 3, in which the structural configuration of the model is selected

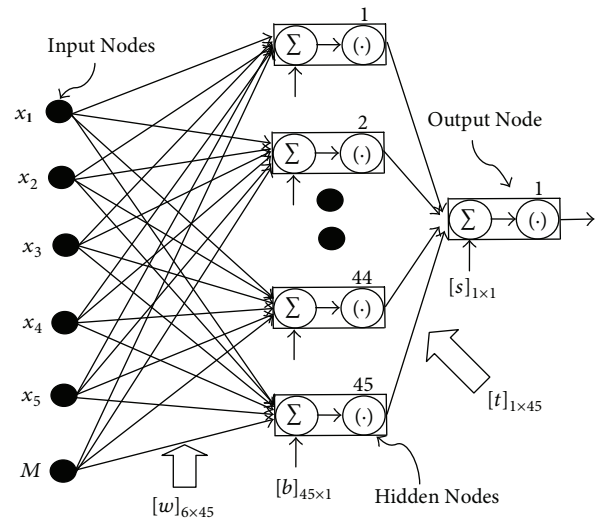

Figure 2: Proposed RBF neural network (6 ∗ 45 ∗ 1).

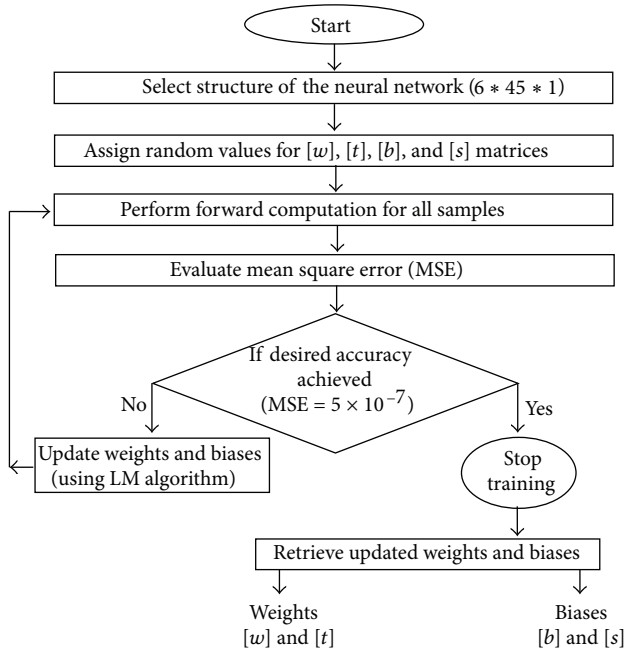

Figure 3: Flow-diagram for training algorithm implementation.

as 6 ∗ 45 ∗ 1. For the applied input pattern, some random numbers between +1 and −1 are assigned to the weights and biases and the output of the model is computed corresponding to that input pattern. Some arbitrary parameters required for training of the neural model like mean square error, learning rate, momentum coefficient, and spread value have also been taken as 5×10^{-7}, 0.1, 0.5, and 0.5, respectively. The error between the calculated and the measured result is then computed and, according to this computed error, all the weights and biases are updated with the help of LM training algorithm. This updating process is carried out after presenting each set of input pattern, until the calculated

accuracy of the model is estimated to be satisfactory. Once it is achieved, the final updated weight matrices $[w]$ and $[t]$ and the bias matrices $[b]$ and $[s]$ are taken out from the trained neural model and these matrices are used for implementing the testing algorithm on FPGA board which is to be discussed in Section 4.

4. Testing Algorithm: Hardware Implementation

The FPGA logic is a class of digital integrated circuitry, whose internal structure can be configured in parallel computing units and it can be repeatedly reprogrammed after manufacturing. Also, the ability to program parallel computing makes FPGA logic very suitable for implementing the parallel structure of the neural networks. The property of hardware reconfigurability is very attractive for the implementation of neural network, as it allows maximum network flexibility. FPGAs have the advantages to be on-site reconfigurable, allowing the use of circuit in few minutes after the design versus many months for the MPLDs (mask programmable logic devices). The FPGA board is basically characterized as functional unit and storage unit. The functional unit consists of processing operations (e.g., multipliers, adders, and activation functions) and storage unit (e.g., RAM containing synaptic weights, bias values, and input buffers). Also, FPGA programmability authorizes design upgrades in the field with no hardware replacement necessary, an impossibility with ASICs [47]. The Spartan-3E family of FPGAs builds on the success of the earlier Spartan-3 family by increasing the amount of logic per I/O, significantly reducing the cost per logic cell. These features improve system performance and reduce the cost of configuration. The Spartan-3E FPGAs, combined with advanced 90 nm process technology, deliver more functionality and bandwidth [47]. The Spartan-3E family is a better alternative to mask programmed ASICs. FPGAs avoid the high initial cost, the lengthy development cycles, and the inherent inflexibility of conventional ASICs. In 1985, Xilinx introduced the world's first FPGA, "XC2064", which contained approximately 1000 logic gates. Since then, the gate density of Xilinx FPGAs has increased 500 times in XC3S500E [48]. In the present work, Sparton-3E XC3S500E FPGA is used which is having two crystal oscillators "X_1" and "X_2" of frequencies 24.0 MHz and 25.0 MHz, respectively.

In Section 2, the weights $[w]$ and $[t]$ and the biases $[b]$ and $[s]$ are optimized during training of the RBF neural network. In this section, these weights and biases have been arranged as per the requirement of testing algorithm. The schematic flow of the testing algorithm to be implemented on FPGA board is shown in Figure 4. The values of weight and bias matrices used in Figure 4 have already been defined in Figure 3. For making it convenient, it has been decided to split the overall implementation into the implementation of input-to-hidden layer and the implementation of hidden-to-output layer. There are a total of 45 symmetrical nodes in the hidden layer and single node in the output layer. All the nodes in hidden layer are fired by the radial basis activation function, whereas the output node is fired by the pure linear

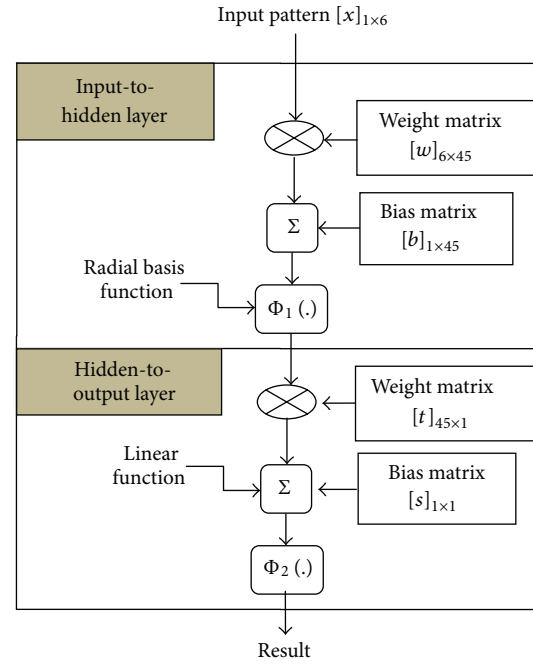

Figure 4: Schematic-flow of the testing algorithm implementation.

activation function. The procedure to implement all nodes in the hidden layer is identical and it is described as below.

The input (H_{i1}) and output (H_{o1}) of hidden node-1 in Figure 2(a) are written as

$$H_{i1} = \left[\left(x_1 \times w_{11}\right) + \left(x_2 \times w_{21}\right) + \cdots + \left(M \times w_{61}\right)\right] + b_1, \tag{1}$$

$$H_{o1} = \exp\left(-{H_{i1}}^2\right). \tag{2}$$

As there is no activation function in the output layer, the role of this layer is just to accumulate the results processed in the earlier hidden layer. Hence, the final result obtained at the output node in Figure 2 is also written as

$$\begin{aligned}\text{Result} = &\left[\left(H_{o1} \times t_{11}\right) + \left(H_{o2} \times t_{12}\right)\right. \\ &\left. + \cdots + \left(H_{o45} \times t_{145}\right)\right] + s.\end{aligned} \tag{3}$$

It is clear from (1) that 6 multipliers and 6 adders are required to implement this node. As a total of 45 symmetrical nodes are there in the hidden layer, then the number of multipliers and adders required to implement the hidden layer is coming out to be 270 each ($6\times45 = 270$). But doing 270 simultaneous multiplications and 270 simultaneous additions is a tedious and time-consuming job. To reduce this complexity, 6 multipliers and 6 adders are used along with some memory-write (MW) and memory-read (MR) operations to implement the 45 nodes in the hidden layer. These multipliers and adders process one node at a time and the result corresponding to that node is stored in the memory storage available on the block RAM of the FPGA board using memory-write (MW) operation. The same procedure is followed for the rest of the hidden nodes. For this purpose, a total of 45 memory-write

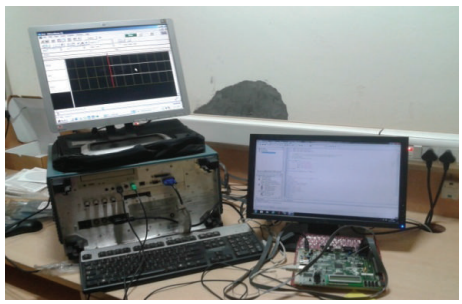

Figure 5: Screen-shot of the set-up during implementation.

(MW) operations are performed for implementing the input-to-hidden layer. Again for implementing the hidden-to-output layer defined in (3), the results stored in the memory are retrieved through 45 memory-read (MR) operations. In this work, a VHDL library for an IEEE 32-bit floating point number precision is also designed for floating point addition fp_add and floating point multiplication fp_mul. Each node value along with their bias counterpart is further added in the adder circuit. The added values obtained on 45 hidden nodes are then passed in radial basis activation function through lookup tables (LUTs). The results of LUTs are further stored for next layer (i.e., hidden-to-output layer) implementation. The same procedure is adopted for implementing this layer as it is done for implementing the input-to-hidden layer. Finally, the required result is obtained at this output node. Another value to the input matrix $[x]$ is assigned as per the requirement and it is then passed through the same process to get the corresponding result.

For the above-mentioned procedure, the testing algorithm of RBF neural network is coded by means of VLSI hardware description language (VHDL) required for implementation on FPGA board "XC3S500E." The generalized RBF neural network is implemented using Xilinx Embedded Development Kit (EDK) version 9.2i, simulated with Mentor Graphics ModelSim PE 10.1, and further downloaded and tested on the FPGA board. The different attributes available on the FPGA board and their use for this implementation are given in Table 5. The screenshot of the set-up used during this implementation is shown in Figure 5.

Some basic operations, data representation, multiplications, additions, and activation functions implementation, used in present work are also discussed below.

4.1. Data Representation. Neural Networks, in general, work with floating-point numbers but working with floating-point numbers in FPGA-based digital hardware is a difficult problem. As a solution to make it easier and further to improve its performance, it is required to convert the floating-point numbers to binary ones. The data patterns, synaptic weights, and bias values are encoded in binary representation. Each value of weights and biases is encoded into 32-bit binary equivalent in which the MSB (most significant bit) is employed as signed bit (0 for plus and 1 for minus) and rest 31 bits represented as magnitude values for weights and biases, whereas the input pattern is represented by 32-bit binary values as there is no negative sign with the input pattern.

Table 5: Attributes of Sparton-3E XC3S500E [48, 49].

Name of attribute	Available	Used	% used
System gates	500 K	12 K	2.40%
Equivalent logic cell	10,476	273	2.61%
Configurable logic block (CLB) array			
Rows	46	27	58.69%
Columns	34	13	38.24%
Total CLBs	1,164	851	73.11%
Total slices	4,656	3364	72.25%
Distributed RAM	73 K	3 K	4.11%
Block RAM	360 K	8 K	2.22%
Dedicated multipliers	20	6	35%
Digital clock managers (DCMs)	4	1	25%
Maximum user I/O	232	7	3.02%
Maximum differential I/O pairs	92	6	6.52%

4.2. Multiplications and Additions. FPGA-based digital processors perform a variety of information-processing tasks. The basic functions encountered in a processing are the various arithmetic operations and, particularly, multiplication and addition of two binary strings are, no doubt, the basic arithmetic building blocks of the present work. The multiplication is done using lookup tables (LUTs) approach. To realize these LUTs, configurable logic blocks (CLBs) have been programmed.

The 32-bit input is multiplied by a 32-bit signed weight to form a 64-bit signed product that is further accumulated with a 32-bit signed bias value into a 64-bit signed sum. Here, 2's complement representation is implemented to handle the multiplication and addition of negative and/or positive numbers. To save the memory space, this 64-bit signed sum is scaled down to 40-bit value. This 24-bit scaling is done after running a software program of neural network in a computer system using the same input data of that of hardware implementation. The 40 bits are the minimum number of bits that can be retained without deteriorating the accuracy ofthe sum. Multiplying a 32-bit input by a 32-bit signed weight produces a 64-bit signed product that is further accumulated with a 32-bit signed bias value into a 64-bit signed sum. The 32×32 bit multiplication is broken into four 8×8 bit multiplications with four addition processes. The most significant partial product (8×8) of each of multiplicand and multiplier is shifted by eight bits before adding it to the least significant partial product.

4.3. Implementation of Activation Functions. The most challenging job in reconfigurable hardware (design or implementation) is the handling of activation functions. Multivariate Gaussian nonlinear function given in (4) is used for the hidden neurons. This Gaussian function is not suitable for FPGA implementation as it consists of an infinite terms in the exponential series. Thus, it needs some truncation as done in (5) as follows:

$$\phi_j(x_i) = \exp\left(-x_i^2\right), \tag{4}$$

$$\phi_j(x_i) = 1 - \frac{x_i^2}{|1} + \frac{x_i^4}{|2} - \frac{x_i^6}{|3} + \cdots + \frac{x_i^{18}}{|9}, \tag{5}$$

where x_i represents the output of ith neuron in the input layer and $\phi_j(\cdot)$ represents the output of jth neuron in the hidden layer.

5. Computed Results and Comparison

The computed seven different parameters of RMSAs, CMSAs, and TMSAs are given in Table 6, whereas their reference counterparts are mentioned in Tables 1, 2, and 3, respectively. For RMSAs, the computed results are compared with their measured counterparts mentioned in Table 1. For CMSAs, the computed results are compared with their measured counterparts mentioned in Table 2 and, for TMSAs, the computed results are compared with their measured counterparts mentioned in Table 3. It is clear from this comparison that most of the calculated points are in very good convergence and only few are off.

A comparison between the present method results and previously computed neural networks results using software implementations [5–11] is given in Table 7, which shows that, in the neural models [5, 10–12], the total absolute error for resonance frequency of RMSAs is calculated as 751.0 MHz, 750.0 MHz, 203.6 MHz, and 557.1 MHz, whereas, in the present model, it is only 55.2 MHz. In case of physical dimensions (both widths and lengths) of rectangular MSAs, the model [12] is having the total absolute error of 0.0653 cm and 0.0490 cm, whereas, in the present work, these are calculated as 0.0179 cm and 0.0146 cm, respectively. In case of resonance frequency of circular MSAs, the proposed method is having the total absolute error of 24 MHz, whereas, in the models [9–11], it is calculated as 92 MHz, 116 MHz, and 462 MHz, respectively. In case of radius of circular MSAs, the present model is having the total absolute error of 0.0280 cm, whereas there is no neural model proposed for computing the radius of circular MSAs in the open literature [5–12]. For resonance frequency of equilateral triangular MSAs, the models [6, 9–11] are having total absolute error as 27 MHz, 220 MHz, 272 MHz, and 23.00 MHz, respectively, whereas, in the present method, it is only 19.5 MHz. In calculating the side length of equilateral triangular MSAs, the error in testing patterns of the present method is only 0.0022 cm, whereas, in the model [8], it is calculated as 0.0213 cm.

The processing time of the hardware implementation is measured by interfacing the FPGA board with TLA 601 logic analyzer, which is a 34-channel Tektronix logic analyzer, having external display with 2 GHz timing, 100 MHz State, and 64 K depth options for up to 256 K depth and/or 200 MHz State [47]. The screen shot of the logic analyzer (TLA601) and FPGA board (XC3S500E) is shown in Figure 4. This measured processing time is in the confirmation to the calculated processing time from the datasheet of FPGA board [48, 49]. The testing algorithm is also implemented in MATLAB software on a computer system for making a comparison of processing time with its hardware counterpart. The time elapsed in software implementation is calculated using MATLAB syntax "cputime" as mentioned in Algorithm 1.

```
clc;
clear all;
Time_1=cputime;
Start of Program
⋮
End of program
Time_2=cputime
Processing_Time=Time_2 − Time_1
= 0.0650 second.
```

Algorithm 1

By doing this procedure, it has been observed that the average time elapsed in software implementation of the testing algorithm on computer system is coming out to be 65 ms, whereas, in its hardware counterpart, it was measured as 320.301 ns.

6. Conclusions

In this paper, the authors have proposed a novel approach for prototyping the neural network models on FPGA platform, which has not been attempted earlier for analyzing and/or designing the microstrip antennas (MSAs). Secondly, the approach is capable of designing the circular MSAs as there is no reported neural model available in the open literature [1–12]. Thirdly, the prototype is capable of computing seven different parameters of three different MSAs, simultaneously, as no neural model has been suggested in the literature [1–12] for computing more than three parameters, simultaneously. Finally, at the same time, a common model has produced more encouraging results for all seven different cases.

The processing time for the approach has been calculated in software implementation and measured in hardware implementation. The average time elapsed in software implementation is coming out to be 65 ms, whereas, in its hardware counterpart, it has been measured as 320.301 ns. Thus, in the present case, the processing of hardware implementation is faster than its software counterparts of the same problem.

In general, the hardware implementation of seven different parameters may require seven different hardware modules, whereas, in the present work, only one hardware module is fulfilling the requirement of seven different hardware chips. The present approach can be considered low cost in this sense.

Table 6: Computed results for RMSAs, CMSAs, and TMSAs.

For RMSAs			For CMSAs		For TMSAs	
f_{rh} (GHz)	W_{rh} (cm)	L_{rh} (cm)	f_{ch} (GHz)	R_{ch} (cm)	f_{th} (GHz)	s_{th} (cm)
2.3107	5.6996	3.7994	6.6351	0.7422	1.5181	4.1006
2.8897	4.5494	3.0498	4.9447	0.7700	2.6355	4.1008
4.2396	2.9498	1.9508	6.0708	0.8227	2.9952	4.0996
5.8400	1.9515	1.3023	5.2296	0.9628	3.9743	4.1001
6.8003	1.7016	1.1020	3.7499	1.0397	4.4394	4.0987
7.6999	1.4019	0.9009	4.7171	1.0721	1.4886	8.7007
8.2695	1.2020	0.8005	4.4275	1.1516	2.5987	8.7004
9.1393	1.0527	0.6998	4.0706	1.2708	2.9704	8.7009
4.7310	1.6992	1.0994	2.0029	1.9994	3.9692	8.7015
7.8705	1.7024	1.1019	1.3606	2.9905	4.4462	8.6990
2.2279	4.1000	4.1400	1.5699	3.4942	1.2788	10.0006
2.2010	6.8587	4.1390	1.5102	3.4939	2.2445	10.0005
2.1811	10.8013	4.1401	1.4429	3.8014	2.5499	10.0006
7.7397	0.8528	1.2924	1.0309	3.9762	3.4013	10.0016
8.4495	0.7927	1.1867	1.0991	4.8514	3.8257	9.9991
3.9693	1.9999	2.5013	0.8255	4.9515	—	—
7.7295	1.0645	1.1857	1.1283	5.0018	—	—
4.6003	0.9068	0.9997	0.8354	6.8012	—	—
5.0592	1.7218	1.8622	0.8293	6.8015	—	—
4.8040	1.8114	1.9619	0.8152	6.8015	—	—
6.5597	1.2706	1.3522	—	—	—	—
5.5999	1.5020	1.6233	—	—	—	—
6.2001	1.3398	1.4143	—	—	—	—
7.0496	1.1213	1.2018	—	—	—	—
5.7998	1.4040	1.4870	—	—	—	—
5.2697	1.5315	1.6319	—	—	—	—
7.9889	0.9063	1.0190	—	—	—	—
6.5696	1.1712	1.2817	—	—	—	—
5.0998	1.3753	1.5812	—	—	—	—
7.9985	0.7768	1.0809	—	—	—	—
7.1325	0.7901	1.2559	—	—	—	—
6.0694	0.9862	1.4507	—	—	—	—
5.8196	1.0019	1.5213	—	—	—	—
6.3789	0.8154	1.4409	—	—	—	—
5.9893	0.7898	1.6205	—	—	—	—
4.6603	1.1990	1.9705	—	—	—	—
4.6006	0.7815	2.2997	—	—	—	—
3.5805	1.2547	2.7539	—	—	—	—
3.9808	0.9727	2.6181	—	—	—	—
3.9007	1.0184	2.6384	—	—	—	—
3.9806	0.8826	2.6744	—	—	—	—
3.9004	0.7757	2.8330	—	—	—	—
3.4702	0.9192	3.1275	—	—	—	—
3.2002	1.0310	3.3774	—	—	—	—
2.9802	1.2645	3.4978	—	—	—	—
3.1503	1.0766	3.3980	—	—	—	—

Table 7: Comparison of absolute errors.

Resonance frequency of RMSAs				
*PM	Ref. [5]	Ref. [12]	Ref. [10]	Ref. [11]
55.2 MHz	751.0 MHz	750.0 MHz	203.6 MHz	557.1 MHz
Width of RMSAs		Length of RMSAs		
*PM	Ref. [12]	*PM	Ref. [12]	
0.0179 cm	0.0653 cm	**0.0146 cm**	0.0490 cm	
Resonance frequency of CMSAs				
*PM	Ref. [10]	Ref. [11]	Ref. [9]	
24.0 MHz	92.0 MHz	116.0 MHz	462.0 MHz	
Radius of circular MSAs				
*PM	Literature			
0.0280 cm	No model is available in the literature [5–12]			
Resonance frequency of TMSAs				
*PM	Ref. [10]	Ref. [11]	Ref. [9]	Ref. [6]
19.5 MHz	27.0 MHz	220.0 MHz	272.0 MHz	23.0 MHz
Side length of TMSAs				
*PM	Ref. [8]			
0.0022 cm	0.0213 cm (for testing patterns only)			

*PM: proposed method.

This study provides a roadmap for implementing neural networks on reconfigurable hardware for microwave applications. As the approach is comparatively faster, having low manufacturing cost, it can be used for creating neural-network-based-FPGA simulators for microwave applications. Also, the proposed approach can be generalized for any number of computing parameters of the microstrip antennas. But, as the dimensionality of the parameters to be computed or the dimensionality of the input patterns increases, the structural configuration such as number of nodes in the hidden layer or sometimes the number of hidden layers also increases. Thus, this results in a considerable increase in the reconfiguration time and finally the training time. It may further require the large memory space if it is implemented on hardware platform. Thus, there is a trade-off between the computing parameters of the microstrip antennas and available environment for the implementing the hardware neural networks.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] Q. J. Zhang and K. C. Gupta, *Neural Networks for RF and Microwave Design*, Artech House, Norwood, Mass, USA, 2000.

[2] P. M. Watson and K. C. Gupta, "EM-ANN models for microstrip vias and interconnects in dataset circuits," *IEEE Transactions on Microwave Theory and Techniques*, vol. 44, no. 12, pp. 2495–2503, 1996.

[3] P. M. Watson and K. C. Gupta, "Design and optimization of cpw circuits using em-ann models for cpw components," *IEEE Transactions on Microwave Theory and Techniques*, vol. 45, no. 12, pp. 2515–2523, 1997.

[4] P. M. Watson, K. C. Gupta, and R. L. Mahajan, "Development of knowledge based artificial neural network models for microwave components," in *Proceedings of the IEEE MTT-S International Microwave Symposium*, vol. 1, pp. 9–12, Baltimore, Md, USA, June 1998.

[5] D. Karaboga, K. Guney, S. Sagiroglu, and M. Erler, "Neural computation of resonant frequency of electrically thin and thick rectangular microstrip antennas," *IEE Proceedings: Microwaves, Antennas and Propagation*, vol. 146, no. 2, pp. 155–159, 1999.

[6] S. Sagiroglu, K. Guney, and M. Erler, "Resonant frequency calculation for circular microstrip antennas using artificial neural networks," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 8, no. 3, pp. 270–277, 1998.

[7] S. Sagiroglu and K. Guney, "Calculation of resonant frequency for an equilateral triangular microstrip antenna with the use of artificial neural networks," *Microwave and Optical Technology Letters*, vol. 14, no. 2, pp. 89–93, 1997.

[8] R. Gopalakrishnan and N. Gunasekaran, "Design of equilateral triangular microstrip antenna using artifical neural networks," in *Proceedings of the IEEE International Workshop on Antenna Technology: Small Antennas and Novel Metamaterials (IWAT '05)*, pp. 246–249, March 2005.

[9] K. Guney, S. Sagiroglu, and M. Erler, "Generalized neural method to determine resonant frequencies of various microstrip antennas," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 12, no. 1, pp. 131–139, 2002.

[10] K. Guney and N. Sarikaya, "A hybrid method based on combining artificial neural network and fuzzy inference system for simultaneous computation of resonant frequencies of rectangular, circular, and triangular microstrip antennas," *IEEE Transactions on Antennas and Propagation*, vol. 55, no. 3, part 1, pp. 659–668, 2007.

[11] K. Guney and N. Sarikaya, "Concurrent neuro-fuzzy systems for resonant frequency computation of rectangular, circular, and triangular microstrip antennas," *Progress in Electromagnetics Research*, vol. 84, pp. 253–277, 2008.

[12] N. Türker, F. Güneş, and T. Yildirim, "Artificial neural design of microstrip antennas," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 14, no. 3, pp. 445–453, 2006.

[13] T. Khan and A. De, "Computation of different parameters of triangular patch microstrip antennas using a common neural model," *International Journal of Microwave and Optical Technology*, vol. 5, pp. 219–224, 2010.

[14] T. Khan and A. De, "A common neural approach for computing different parameters of circular patch microstrip antennas," *International Journal of Microwave and Optical Technology*, vol. 6, no. 5, pp. 259–262, 2011.

[15] T. Khan and A. De, "Design of circular/triangular patch microstrip antennas using a single neural model," in *Proceedings of the IEEE Applied Electromagnetics Conference (AEMC '11)*, pp. 18–22, Kolkata, India, December 2011.

[16] T. Khan and A. De, "A generalized neural method for simultaneous computation of resonant frequencies of rectangular, circular and triangular microstrip antennas," in *Proceedings of the International Conference on Global Innovation Technology and Science (ICGITS '13)*, pp. 4–6, Saintgits College of Engineering, Kerala, India, April 2013.

[17] T. Khan and A. De, "Prediction of resonant frequencies of rectangular, circular and triangular microstrip antennas using

a generalized RBF neural model," *International Journal of Scientific and Engineering Research*, vol. 4, pp. 182–187, 2013.

[18] T. Khan and A. De, "A generalized neural simulator for computing different parameters of circular/triangular microstrip antennas simultaneously," in *Proceedings of the IEEE Asia-Pacific Conference on Applied Electromagnetics (APACE '12)*, pp. 350–354, Melaka, Malaysia, December 2012.

[19] T. Khan and A. De, "A generalized ANN model for analyzing and synthesizing rectangular, circular and triangular microstrip antennas," *Chinese Journal of Engineering*, vol. 2013, Article ID 647191, 9 pages, 2013.

[20] D. F. Wolf, R. A. F. Romero, and E. Marques, "Using embedded processors in hardware models of artificial neural networks," in *Proceedings of the 5th Brazilian Symposium of Intelligent Automation (SBAI '01)*, pp. 78–83, 2001.

[21] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," *ACM Computing Surveys*, vol. 34, no. 2, pp. 171–210, 2002.

[22] O. R. Amos and J. C. Rajapakse, *FPGA Implementations of Neural Networks*, Springer, New York, NY, USA, 2006.

[23] A. S. Patwal, J. Prashanth, and B. Girish, "Implementation of artificial neural networks on FPGA for adaptive interference canceling," in *Proceedings of the 1st International Conference on Computers, Communications and Signal Processing with Special Track on Biomedical Engineering (CCSP '05)*, pp. 310–312, November 2005.

[24] D. Zhang, H. Li, and S. Y. Foo, "A simplified FPGA implementation of neural network algorithms integrated with stochastic theory for power electronics applications," in *Proceedings of the 31st Annual Conference of IEEE Industrial Electronics Society (IECON '05)*, Raleigh, NC, USA, November 2005.

[25] J. Zhai, J. Zhou, L. Zhang, J. Zhao, and W. Hong, "ANFIS implementation in FPGA for power amplifier linearization with digital predistortion," in *Proceeding of the International Conference on Microwave and Millimeter Wave Technology (ICMMT '08)*, vol. 3, pp. 1474–1476, Nanjing, China, April 2008.

[26] A. Gomperts, A. Ukil, and F. Zurfluh, "Development and implementation of parameterized FPGA-based general purpose neural networks for online applications," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 1, pp. 78–89, 2011.

[27] J. Shawash and D. R. Selviah, "Real-time non-linear parameter estimation using the Levenberg-Marquardt algorithm on field programmable gate arrays," *IEEE Transactions on Industrial Electronics*, vol. 60, no. 1, pp. 170–176, 2012.

[28] W. To, Z. Salcic, and S. K. Nguang, "Prototyping neuroadaptive smart antenna for 3G wireless communications," *Eurasip Journal on Applied Signal Processing*, vol. 2005, no. 7, pp. 1093–1109, 2005.

[29] E. Al Zuraiqi, M. Joler, and C. G. Christodoulou, "Neural networks FPGA controller for reconfigurable antennas," in *Proceedings of the IEEE Antennas and Propagation Society International Symposium (APSURSI '10)*, pp. 1–4, Toronto, Canada, July 2010.

[30] R. Ghayoula, N. Fadlallah, G. Bertand, G. Ali, and R. Mohamed, "A novel design of phased antenna array based on digital beamforming," *International Journal of Research and Reviews in Wireless Communications*, vol. 1, no. 3, 2011.

[31] J.-L. Fournier, D. Titz, F. Ferrero, C. Luxey, E. Dekneuvel, and G. Jacquemod, "Phased array antenna controlled by neural network FPGA," in *Proceedings of the Loughborough Antennas and Propagation Conference (LAPC '11)*, pp. 1–5, Loughborough, UK, November 2011.

[32] E. Chang, S. A. Long, and W. F. Richards, "An experimental investigation of electrically thick rectangular microstrip antennas," *IEEE Transactions on Antennas and Propagation*, vol. AP-34, no. 6, pp. 767–772, 1986.

[33] K. R. Carver, "Practical analytical techniques for the microstrip antenna," in *Proceedings of the Workshop on Printed Circuit Antenna Technology*, pp. 7.1–7.20, New Mexico State University, Las Cruces, NM, USA, 1979.

[34] M. Kara, "The resonant frequency of rectangular microstrip antenna elements with various substrate thicknesses," *Microwave and Optical Technology Letters*, vol. 11, no. 2, pp. 55–59, 1996.

[35] M. Kara, "Closed-form expressions for the resonant frequency of rectangular microstrip antenna elements with thick substrates," *Microwave and Optical Technology Letters*, vol. 12, no. 3, pp. 131–136, 1996.

[36] J. S. Dahele and K. F. Lee, "Effect of substrate thickness on the performance of a circular-disk microstrip antenna," *IEEE Transactions on Antennas and Propagation*, vol. 31, no. 2, pp. 358–360, 1983.

[37] J. S. Dahele and K. F. Lee, "Theory and experiment on microstrip antennas with air-gaps," *Proceedings of IEE*, vol. 132, no. 7, pp. 455–460, 1985.

[38] K. R. Carver, "Practical analytical techniques for the microstrip antenna," in *Proceedings of the Workshop on Printed Circuit Antennas*, pp. 7.1–7.20, New Mexico State University, 1979.

[39] K. Antoskiewicz and L. Shafai, "Impedance characteristics of circular microstrip patches," *IEEE Transactions on Antennas and Propagation*, vol. 38, no. 6, pp. 942–946, 1990.

[40] J. Q. Howell, "Microstrip antennas," *IEEE Transactions on Antennas and Propagation*, vol. 23, no. 1, pp. 90–93, 1975.

[41] T. Itoh and R. Mittra, "Analysis of a microstrip disk resonator," *Archiv fur Elektronik und Ubertragungstechnik*, vol. 27, no. 11, pp. 456–458, 1973.

[42] F. Abboud, J. P. Damiano, and A. Papiernik, "New determination of resonant frequency of circular disc microstrip antenna: application to thick substrate," *Electronics Letters*, vol. 24, no. 17, pp. 1104–1106, 1988.

[43] W. Chen, K. F. Lee, and J. S. Dahele, "Theoretical and experimental studies of the resonant frequencies of the equilateral triangular microstrip antenna," *IEEE Transactions on Antennas and Propagation*, vol. 40, no. 10, pp. 1253–1256, 1992.

[44] J. S. Dahele and K. F. Lee, "On the resonant frequencies of the triangular patch antenna," *IEEE Transactions on Antennas and Propagation*, vol. 35, no. 1, pp. 100–101, 1987.

[45] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302–309, 1991.

[46] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[47] http://www.xilinx.com/bvdocs/publications/.

[48] http://uk.farnell.com/xilinx/xc3s500e-4pqg208i/fpgaspartan-3e-500k-gates-208pqfp/dp/1762479.

[49] http://www.tequipment.net/TektronixTLA601.html.