Analog Integr Circ Sig Process (2012) 73:819–828
DOI 10.1007/s10470-012-9857-0

FPGA-implementation of dynamic time delay neural network for power amplifier behavioral modeling

Mohammed Bahoura · Chan-Wang Park

Received: 23 September 2011 / Revised: 8 March 2012 / Accepted: 23 March 2012 / Published online: 11 April 2012

Abstract In this paper, we propose new architectures for FPGA-implementation of a dynamic neural network power amplifier behavioral modeling. The real-valued time-delay neural network (RVTDNN) and the backpropagation (BP) learning algorithm were implemented on FPGA using Xilinx system generator for DSP and the Virtex-6 FPGA ML605 evaluation kit. Different RVTDNN architectures are proposed for various values of the number of hidden neurons, the activation function resolution, and the fixed-point data format. These architectures are evaluated and compared in terms of modeling performances and resource utilization using 16-QAM modulated test signal.

Keywords Adaptive modeling · Neural network · Memory effects · Power amplifier · FPGA

1 Introduction

Current wide band 3rd generation and 4th generation power amplifier has very severe non-linear distortion by envelope variable signal. To correct this non-linearity of power amplifier (PA), we use linearizer such as feedback, feed-forward, and predistortion. Among those methods, digital predistortion is widely used because of its simple structure to implement. To design a precise linearizer, we have to model the PA nonlinearity accurately. The most of previous models consider the memory effect in real PAs that often arise due to thermal effects and long time constant in a dc bias circuits. This means that the AM/AM and AM/PM functions are not static, but change depending on the history of past levels. It is known that these phenomena come from memory effects, thus a model to treat these phenomena is needed. To take account memory effects, dynamic AM/AM and AM/PM are considered to correct non-linearity of PA. So linearizing PAs must be adaptive to take into account memory effects. Previously many PA models were reported such as Volterra series model [1], polynomial model [2], neural network model [3–6] and the neural fuzzy model [7]. The neural network approach is interesting because of the capability of accurately modeling the PA. With neural networks based on time delayed lines (TDNN), we can even compensate the short term memory effects presented in the characteristic of PAs. Besides, the realization of adaptive architecture allows to remedy the problems of the long term memory effects such as electro-thermal memory effect.

The most of PA modeling algorithms proposed by previous authors [3–6] were just implemented in a software environment, e.g. MATLAB with test equipments to model the PAs without using field programmable gate array (FPGA). Different implementation structures will lead to very different results in terms of cost and performance values. The important issues related to practical hardware implementation in FPGA were not discussed previously by other authors. In this paper, we present a low complexity and low cost solution for real-time implementation of the proposed neural network PA model using FPGA chip. The proposed architectures have been implemented on FPGA using Xilinx system generator and Virtex-6 FPGA ML605 evaluation kit. The rest of the paper is organized as follows. Section 2 presents the real-valued time-delay neural network (RVTDNN) and the backpropagation (BP) learning

M. Bahoura (✉) · C.-W. Park
Department of Engineering, Université du Québec à Rimouski, 300, allée des Ursulines, Rimouski, QC G5L 3A1, Canada
e-mail: Mohammed_Bahoura@uqar.qc.ca

C.-W. Park
e-mail: Chan-Wang_Park@uqar.qc.ca

algorithm. Section 3 describes in detail the hardware implementation and the resource requirement of the proposed architectures. Simulation results are presented and discussed in Sect. 4. Finally, conclusion is given in Sect. 5.

2 Method

2.1 Real-valued time-delay neural network

The real-valued time-delay neural network (RVTDNN) is based on a multi-layer perceptron (MLP) by adding tapped delay lines (TDLs) in its inputs [4]. At any moment n, the past m values of the baseband inputs $I_{in}(n)$ and $Q_{in}(n)$ are also presented to the MLP network in order to consider the memory effects. The input data are defined by an input vector $\mathbf{x}$ as follows:

$$\mathbf{x} = \begin{array}{l}[I_{in}(n), I_{in}(n-1), \ldots, I_{in}(n-m),\\ Q_{in}(n), Q_{in}(n-1), \ldots, Q_{in}(n-m)]\end{array} \tag{1}$$

where m is the memory depth.

As shown in Fig. 1, the MLP network is a collection of neurones (nodes), arranged together in layers in feed-forward manner. It has been demonstrated that a MLP with only one hidden layer can serve as an universal estimator if sufficient neurons are used [8, 9]. Signals pass into the input layer nodes, progress forward through the hidden layers and finally emerge from the output layer [10]. Figure 1 represents an example of an MLP network characterized by $2m+2$ inputs, one hidden of N nodes, and 2 outputs. Each node j, in the hidden layer, receives the output of each node i from the input layer through a connection of weight $w^h_{j,i}$ and then produce a corresponding response y^h_j which is forwarded to the output layer. In fact, each node j performs a weighted sum which is transferred by a nonlinear function φ_h according to [11].

$$v^h_j = \sum_{i=1}^{2m+2} w^h_{j,i} x_i + w^h_{j,0} \qquad j = 1, \ldots, N \tag{2}$$

$$y^h_j = \varphi_h\left(v^h_j\right) \tag{3}$$

where $w^h_{j,0}$ represents the bias of the jth neuron and m is the memory depth.

In the same manner, the output of each neuron k, in output layer, is given by.

$$v^o_k = \sum_{j=1}^{N} w^o_{k,j} y^h_j + w^o_{k,0} \qquad k = 1, 2 \tag{4}$$

$$y^o_k = \varphi_o\left(v^o_k\right) \tag{5}$$

where φ_o is the activation function, $w^o_{k,j}$ is synaptic weight connecting the output of the jth neuron in the hidden layer to the kth neuron of the output layer, and $w^o_{k,0}$ is the bias of the kth neuron. The activation functions of the hidden neurons are typically hyperbolic tangent or logistic sigmoid. The output neurons activation function can be linear or non-linear depending on the task performed by the network: a function approximation or a classification, respectively [12]. In this

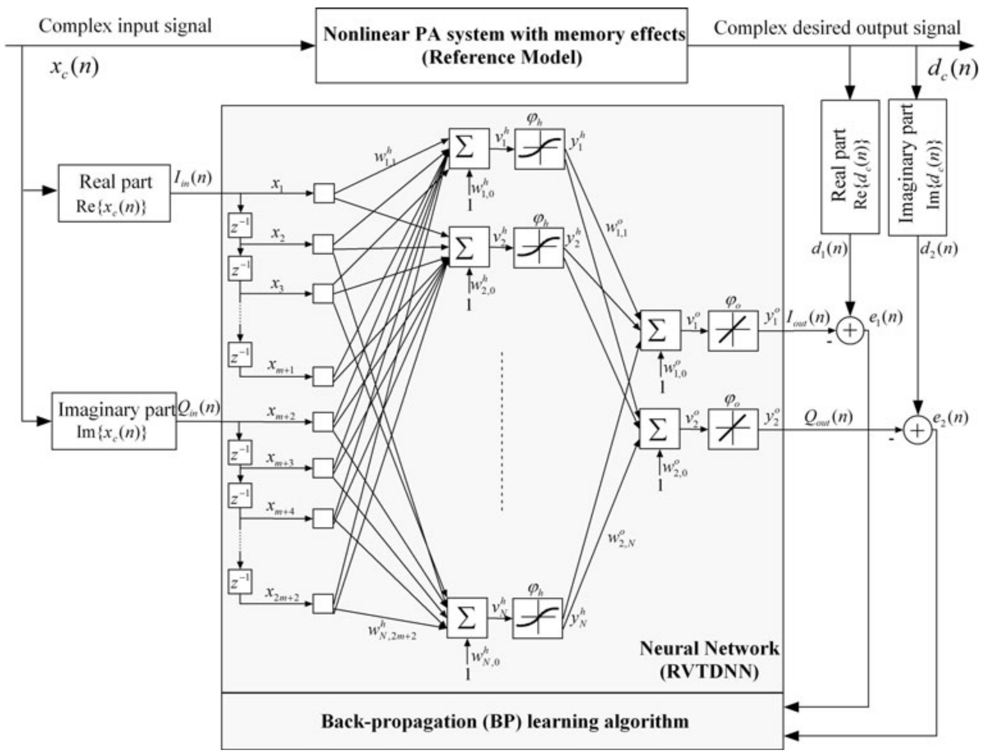
Complex input signal
$x_c(n)$
Nonlinear PA system with memory effects (Reference Model)
Complex desired output signal
$d_c(n)$
Real part $\mathrm{Re}\{x_c(n)\}$
$I_{in}(n)$
Imaginary part $\mathrm{Im}\{x_c(n)\}$
$Q_{in}(n)$
z^{-1}
x_1 x_2 x_3 x_{m+1} x_{m+2} x_{m+3} x_{m+4} x_{2m+2}
$w^h_{1,1}$ $w^h_{N,2m+2}$
Σ φ_h v^h_1 y^h_1 $w^h_{1,0}$ v^h_2 y^h_2 $w^h_{2,0}$ v^h_N y^h_N $w^h_{N,0}$
$w^o_{1,1}$ $w^o_{2,N}$
φ_o v^o_1 y^o_1 $w^o_{1,0}$ v^o_2 y^o_2 $w^o_{2,0}$
$I_{out}(n)$ $Q_{out}(n)$
Real part $\mathrm{Re}\{d_c(n)\}$
Imaginary part $\mathrm{Im}\{d_c(n)\}$
$d_1(n)$ $d_2(n)$ $e_1(n)$ $e_2(n)$
Neural Network (RVTDNN)
Back-propagation (BP) learning algorithm

Fig. 1 Diagram of dynamic two-layer RVTDNN PA behavioral modeling

paper, we use an hyperbolic tangent function $\varphi_h(x) = (1 - e^{-2x})/(1 + e^{-2x})$ for the hidden layer and a linear function $\varphi_o(x) = x$ for the output layer that it is a common choice for power amplifier modeling [4–6]. The hyperbolic tangent is preferred to the logistic sigmoid because of its bipolarity. The connection weights are determined using the backpropagation learning algorithm.

2.2 Back-propagation algorithm

The backpropagation algorithm performs the gradient descent on error. It can be implemented in two different ways: sequential mode and batch mode [13]. In sequential mode, the weights are updated after each training example (pattern) is applied to the network. In batch mode, all the training examples, that constitute an epoch, are applied to the network before the weights are updated. It is obvious that for a real-time applications, the sequential learning must be chosen.

At the *n*th iteration (i.e. presentation of the *n*th training example), the error signal at the output of neurone *k* is defined by [13]

$$e_k(n) = d_k(n) - y_k^o(n) \tag{6}$$

where $d_k(n)$ and $y_j^h(n)$ are the desired and the actual output of this neuron, respectively.

The instantaneous value of the total error energy at the output layer is defined as

$$E(n) = \frac{1}{2}\sum_{k=1}^{2} e_k^2(n) \tag{7}$$

This error can be reduced by updating the weights using the gradient descent method:

$$\Delta w_{k,j}(n) = -\eta \frac{\partial E(n)}{\partial w_{k,j}(n)} \tag{8}$$

where $(0 < \eta < 1)$ is the *learning rate*. The next value of $w_{k,j}(n)$ is then given by:

$$w_{k,j}(n+1) = w_{k,j}(n) + \Delta w_{k,j}(n) \tag{9}$$

2.2.1 *Output layer*

The connection weights in the output layer are updated using [13]

$$\Delta w_{k,j}^o(n) = \eta \delta_k^o(n) y_j^h(n) \tag{10}$$

where the local gradient $\delta_k^o(n)$ is defined by

$$\delta_k^o(n) = e_k(n)\varphi_o'(v_k^o(n)) \tag{11}$$

As a *linear* function is used, $\varphi_o'(v_k^o(n)) = 1$, Eq. 11 can be simplified to

$$\delta_k^o(n) = e_k(n) \tag{12}$$

2.2.2 *Hidden layer*

The connection weights in the hidden layer are updated using [13]

$$\Delta w_{j,i}^h(n) = \eta \delta_j^h(n) x_i(n) \tag{13}$$

where the local gradient $\delta_j^h(n)$ is defined by

$$\delta_j^h(n) = \varphi_h'(v_j^h(n)) \sum_{k=1}^{2} \delta_k^o(n) w_{k,j}^o(n) \tag{14}$$

As an *hyperbolic tangent* function is used, $\varphi_h'(v_j^h(n)) = 1 - \varphi_h^2(v_j^h(n))$, Eq. 14 can be simplified to

$$\delta_j^h(n) = (1 - \varphi_h^2(v_j^h(n))) \sum_{k=1}^{2} \delta_k^o(n) w_{k,j}^o(n) \tag{15}$$

In fact, dynamic RVTDNN-based models are continuously trained using the sequential mode of the backpropagation algorithm. After each input is applied to the network, the connection weights are updated on FPGA chip using (9), (10), and (13).

2.3 Reference model

To illustrate the performance of the RVTDNN model, we use a reference PA system based on the Wiener model (Fig. 2). We used the most popular memoryless nonlinear PA model proposed by Saleh [14], which is defined by the following AM/AM and AM/PM characteristics

$$A(r(t)) = \frac{\alpha_A r(t)}{1 + \beta_A r^2(t)} \tag{16}$$

$$\Phi(r(t)) = \frac{\alpha_\Phi r(t)}{1 + \beta_\Phi r^2(t)} \tag{17}$$

where *r(t)* stands for the envelope of the applied input signal. Typical parameter values are: $\alpha_A = 2.1587, \beta_A = 1.1517, \alpha_\Phi = 4.0033$ and $\beta_\Phi = 9.1040$.

The memory effects are modeled by a 2nd order finite impulse response (FIR) filter defined by

$$H(z) = 1 + 0.5z^{-2} \tag{18}$$

3 FPGA implementation

The proposed models are implemented on FPGA using Xilinx system generator (XSG) and the Virtex-6 FPGA ML605 evaluation kit. As shown in Fig. 3, the top-level

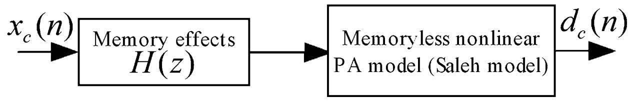

Fig. 2 Wiener model for nonlinear PA considering memory effects

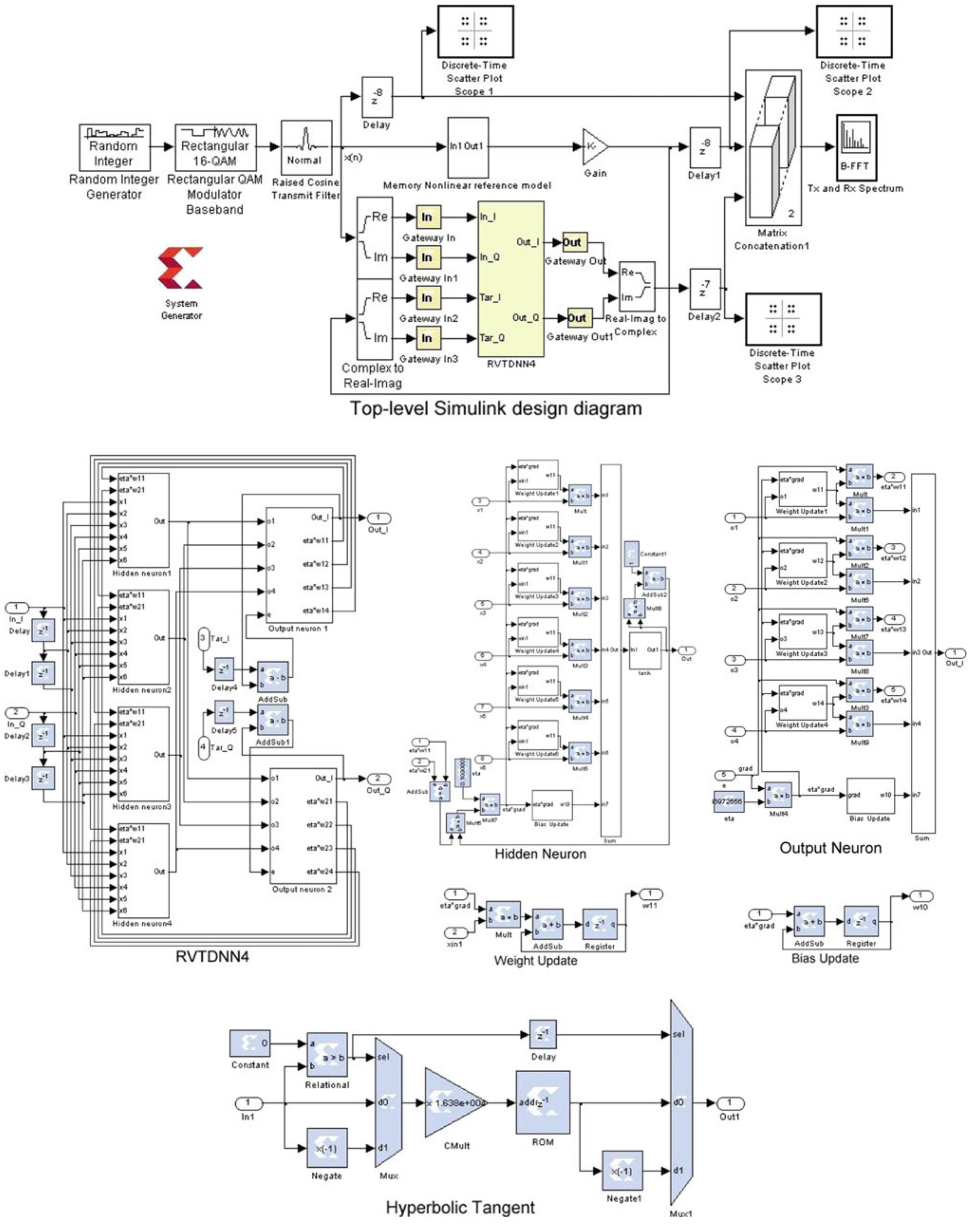

Fig. 3 Neural network architecture based on Xilinx system generator blockset for RF power amplifier modeling

design diagram includes SIMULINK blocks and XSG-based subsystem (RVTDNN4). The Simulink blocks communicate with XSG-based subsystem through "*Gateway In*" and "*Gateway Out*" blocks. To simplify illustration, the RVTDNN4 subsystem is described by a small neural network of four hidden neurons. The hyperbolic tangent function is implemented using a look-up table (LUT) schema that takes advantage from the symmetric characteristic of this function [15]. To reduce the critical path, the connection weights are updated using $w(n+1) = w(n) + \Delta w(n)$ instead of $w(n) = w(n-1) + \Delta w(n)$ used previously [11].

It is obvious that the modeling ability of the network depends on both the number of neurons that can be implemented and the precision of the fixed-point arithmetic operations. Because the hardware resources of a given FPGA are limited, an increase in the number of hidden neurons will be done at the cost of the number of bits used in the fixed-point format and vice versa. Our approach to implement a functional RVTDNN-based model is to identify the minimum number of hidden neurons that allows successful modeling when using high precision data (64 bits) and progressively reduce the number of quantization bits so that the network can be loaded into the FPGA chip. Simulations results show that a model of only six hidden neurons (RVTDNN6) can efficiently approximate the PA output. Except memory address width are unsigned, data in all other XSG block are 2's complement signed format. For a given fixed-point format of N_F bits, 4 bits are reserved for the sign and the integer part and the remaining $(N_F - 4)$ bits for the fractional part.

3.1 Fixed-point format of 36 bits (FIX36TW)

Initially [11], we have opted for the default *Quantization* and *Overflow* options because they have the lower cost in logic hardware.

- *Quantization*: Truncate,
- *Overflow*: Wrap.

In this case, the number of bits (N_F) that allows functional FPGA implementation of the smallest functional model (RVTDNN6) can not be reduced to less than 36. This fixed-point format is refereed by FIX36TW.

Table 1 gives in detail the resource requirement and the maximum operating frequency for different architectures. These architectures include the same number of neurons in hidden layer ($N = 6$) and the same fixed-point format (FIX36TW) but differ in the address bus size of the ROM memory used to implement the hyperbolic tangent function. For example, RVDTNN6_36_16 and RVDTNN6_36_11 use of ROM memories of 2^{16} words and 2^{11} words, respectively. With the used FPGA, the number of the hidden neurons can not be increased because the available RAMB36E1 and DSP48E1 blocks have almost all been used. However, the percentage of the required logic components (Flip-Flops and LUTs) remains negligible. As can be verified, a RVDTNN6 composed of 128 multipliers requires 640 DSP48E1s. In fact, each DSP48E1 slice contains a 25×18 multiplier, an adder and an accumulator. So, a multiplication of two numbers coded with FIX136TW format requires 5 DSP48E1s. This table shows also that the maximum operating frequency is approximately constant for these architectures and slightly improved compared to the previous architecture [11]. In fact, the connection weights are now updated using $w(n+1) = w(n) + \Delta w(n)$ instead of $w(n) = w(n-1) + \Delta w(n)$.

3.2 Fixed-point format of 24 bits (FIX24RS)

In this paper, we have opted for advanced *Quantization* and *Overflow* options because they insert logic to perform arithmetic operations.

- *Quantization*: Round,
- *Overflow*: Saturate.

Table 1 Resource utilization and maximum operating frequency for various architectures using the FIX36TW format

Architecture	RVTDNN6_36_16	RVTDNN6_36_15	RVTDNN6_36_14	RVTDNN6_36_13	RVTDNN6_36_12	RVTDNN6_36_11
Resource utilization						
Flip Flops (301,440)	2,262 (0.7 %)	2,256 (0.7 %)	2,250 (0.7 %)	2,352 (0.8 %)	2,238 (0.7 %)	2,238 (0.7 %)
LUTs (150,720)	6,542 (4.3 %)	6,076 (4.0 %)	5,630 (3.7 %)	5,604 (3.7 %)	5,589 (3.7 %)	5,572 (3.7 %)
Bonded IOs (600)	217 (36.1 %)	217 (36.1 %)	217 (36.1 %)	217 (36.1 %)	217 (36.1 %)	217 (36.1 %)
RAMB36E1s (416)	384 (92.3 %)	192 (46.1 %)	96 (23.1 %)	48 (11.5 %)	24 (5.7 %)	12 (2.8 %)
DSP48E1s (768)	640 (83.3 %)	640 (83.3 %)	640 (83.3 %)	640 (83.3 %)	640 (83.3 %)	640 (83.3 %)
Max. Opera. Freq. (MHz)	18.725	18.793	18.883	18.954	19.686	19.686

For example, RVDTNN6_36_16 represents an architecture of 6 neurons in hidden layer that uses 36 bits for coding data and 16 bits for addressing the ROM memory. Resources availability of Xilinx Virtex-6 XC6VLX240T FPGA are given between *brackets* in the *left column*

Table 2 Resource utilization and maximum operating frequency for various architectures using the FIX24RS format

Architecture	RVTDNN6_24_16	RVTDNN6_24_15	RVTDNN8_24_16	RVTDNN8_24_15	RVTDNN10_24_15	RVTDNN15_24_15
Resource utilization						
Flip Flops (301,440)	1,524 (0.5 %)	1,518 (0.5 %)	1,968 (0.6 %)	1,960 (0.6 %)	2,392 (0.8 %)	3,507 (1.1 %)
LUTs (150,720)	11,672 (7.7 %)	11,422 (7.5 %)	15,494 (10.2 %)	15,163 (10.0 %)	18,602 (12.3 %)	28,167 (18.7 %)
Bonded IOs (600)	145 (24.1 %)	145 (24.1 %)	145 (24.1 %)	145 (24.1 %)	145 (24.1 %)	145 (24.1 %)
RAMB36E1s (416)	264 (63.5 %)	132 (31.7 %)	352 (84.6 %)	176 (42.3 %)	220 (52.9 %)	330 (79.3 %)
DSP48E1s (768)	256 (33.3 %)	256 (33.3 %)	340 (44.3 %)	340 (44.3 %)	424 (55.2 %)	634 (82.5 %)
Max. Opera. Freq. (MHz)	27.796	28.958	27.777	28.936	28.936	28.936

For example, RVDTNN10_24_15 represents an architecture of 10 neurons in hidden layer that uses 24 bits for coding data and 15 bits for addressing the ROM memory. Resources availability of Xilinx Virtex-6 XC6VLX240T FPGA are given between *brackets* in the *left column*

In this case, the number of bits (N_F) that allows functional FPGA implementation of the smallest functional model (RVTDNN6) can be reduced to 24. This fixed-point format is refereed by FIX24RS. Reducing the number of bits N_F can save considerably the DSP48E1 resources. A multiplication of two numbers coded with FIX24RS requires only 2 DSP48E1s rather than 5 with the previous format. Consequently, RVTDNN models with larger number of neurons in the hidden layer can be implemented on the used FPGA. Table 2 gives the resource requirement and the maximum operating frequency for various architectures which differ in the number of the hidden neurons and the address bus size of the ROM memories. For example, RVDTNN10_24_15 represents an architecture of 10 neurons in the hidden layer that uses 24 bits for coding data and 15 bits for addressing the ROM memories. It can be shown that FIX24RS format allows us to implement RVDTNN models with up to 15 neurons in the hidden layer. The maximum operating frequency is also increased with this format. This can be explained by the minimization of the critical path caused by the reduction of the number of multipliers (DSP48E1s) [16].

4 Experiment and results

As shown in the top-level design diagram of Fig. 3, the proposed architectures was evaluated with a 16-QAM modulated test signal generated using the communication toolbox of MATLAB/SIMULINK.

4.1 Setup

This subsection provides values used to set parameters in various Simulink blocks.

- *Random Integer Generator*
  - M-ary number: 16,
  - Sample time: 10^{-6},
  - Output data type: Integer.
- *Rectangular QAM Modulator Baseband*
  - M-ary number: 16,
  - Input data type: Integer,
  - Constellation ordering: Binary,
  - Normalization method: Average power,
  - Average power (watts): 0.1,
  - Output data type: Double.
- *Raised Cosine Transmit Filter*
  - Filter type: Normal,
  - Group delay: 8,
  - Rolloff factor: 0.3,
  - Upsampling factor: 8,
  - Filter gain: Normalized.
- *Memory Nonlinear Reference Model* The parameters of the PA reference model are defined in the Sect. 2.3.

Due to the up-sampling of 8 in the transmit filter, the sampling frequency of XSG blocks are 8 times greater than that of the SIMULINK blocks. The connection weights are randomly initialized.

For an effective comparison, the connection weights of the RVTDNN networks of the same size are initialized with the same random values. Also, the same input signal is applied to all the evaluated RVTDNN architectures.

4.2 Results and discussion

Figures 4, 5, and 6 show the time-domain cartesian components, constellation, AM/AM and AM/PM characteristics of the RVTDNN6 PA model for different ROM address bus sizes (12, 14 and 16), using the FIX36TW fixed-point format. It is obvious that both nonlinearity and memory effects are well modeled by the three RVTDNN6 models. However, Fig. 7 show that the power spectrum density can be greatly affected by the ROM memory size used to implement the hyperbolic tangent function. It is obvious that the best results are obtained by the RVTDNN6_36_16 model.

Figures 8, 9, and 10 show the time-domain cartesian components, constellation, AM/AM and AM/PM characteristics of the RVTDNN6 PA model for different ROM address bus sizes (12, 14 and 16), using the FIX24RS fixed-point format. Both nonlinearity and memory effects are well modeled by the three RVTDNN model. As previously,

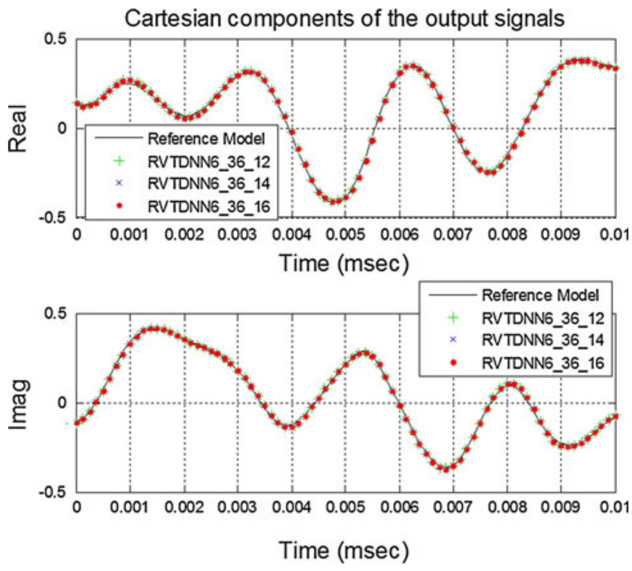

Fig. 4 Output signal in the time domain for the reference and RVTDNN6_36_12, RVTDNN6_36_14 and RVTDNN6_36_16 models

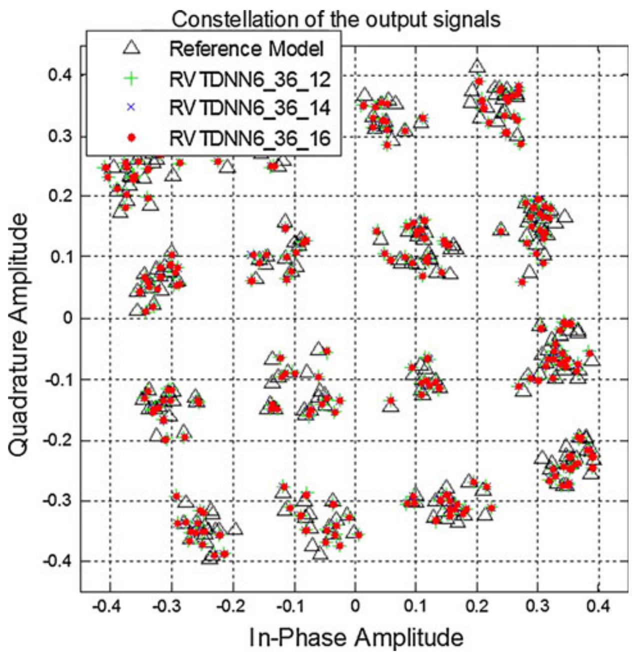

Fig. 5 Output signal constellation for the reference and RVTDNN6_36_12, RVTDNN6_36_14 and RVTDNN6_36_16 models

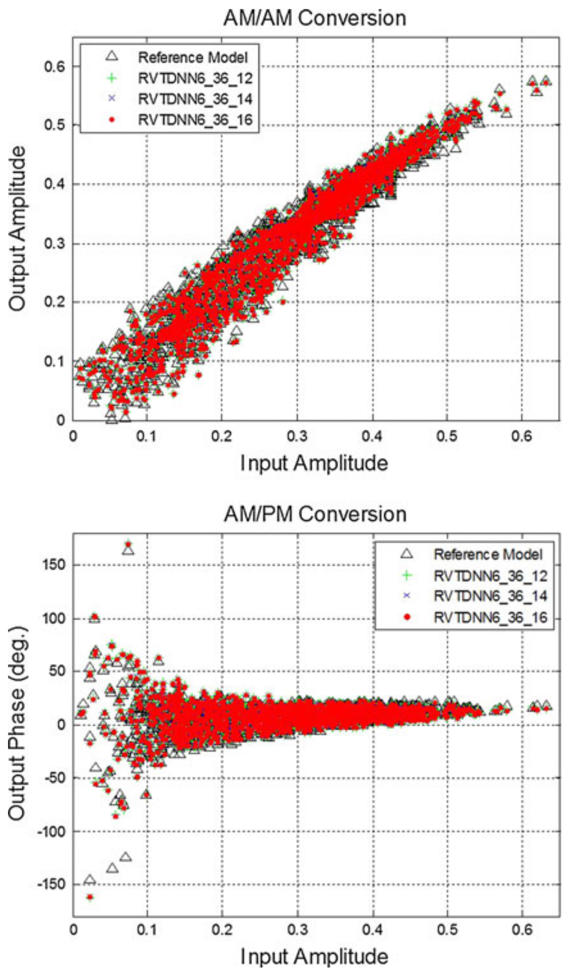

Fig. 6 AM/AM and AM/PM characteristics for the reference and RVTDNN6_36_12, RVTDNN6_36_14 and RVTDNN6_36_16 models

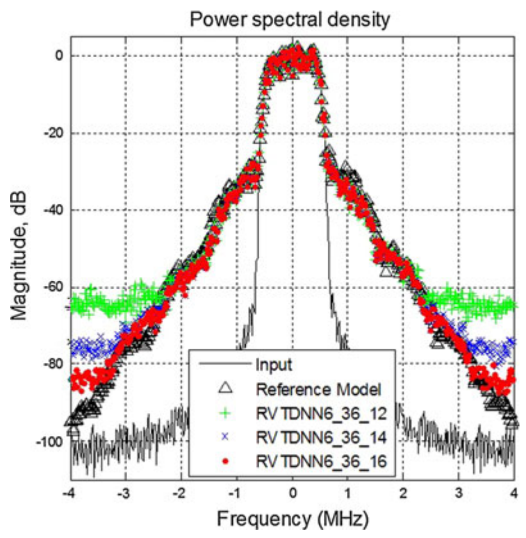

Fig. 7 Power spectral density of input and output signals for the reference and RVTDNN6_36_12, RVTDNN6_36_14 and RVTDNN6_36_16 models

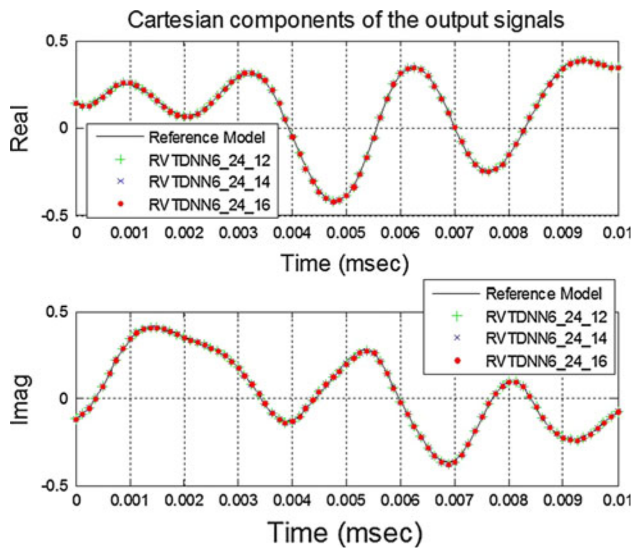

Fig. 8 Output signal in the time domain for the reference and RVTDNN6_24_12, RVTDNN6_24_14 and RVTDNN6_24_16 models

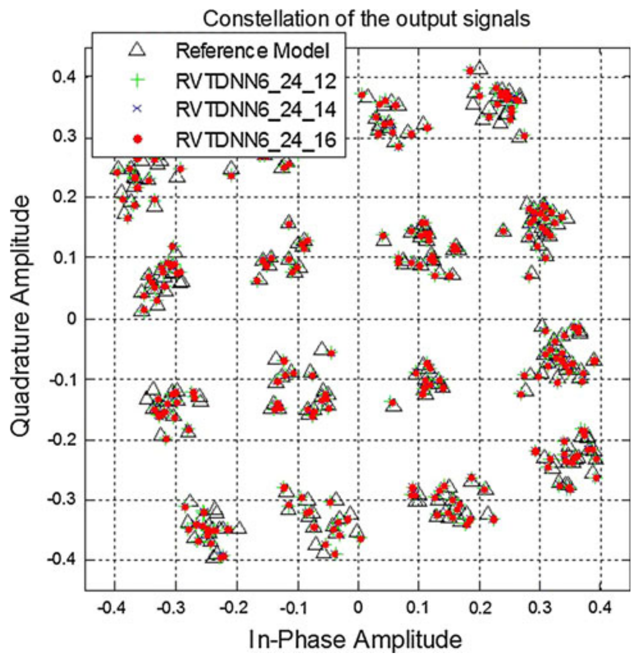

Fig. 9 Output signal constellation for the reference and RVTDNN6_24_12, RVTDNN6_24_14 and RVTDNN6_24_16 models

Fig. 11 show that the power spectrum density can be greatly affected by the ROM memory size. The best results are obtained by the RVTDNN6_24_16 model. In light of these results, it is clear that only the PSD curve is affected by the ROM memory size used to calculate the activation function.

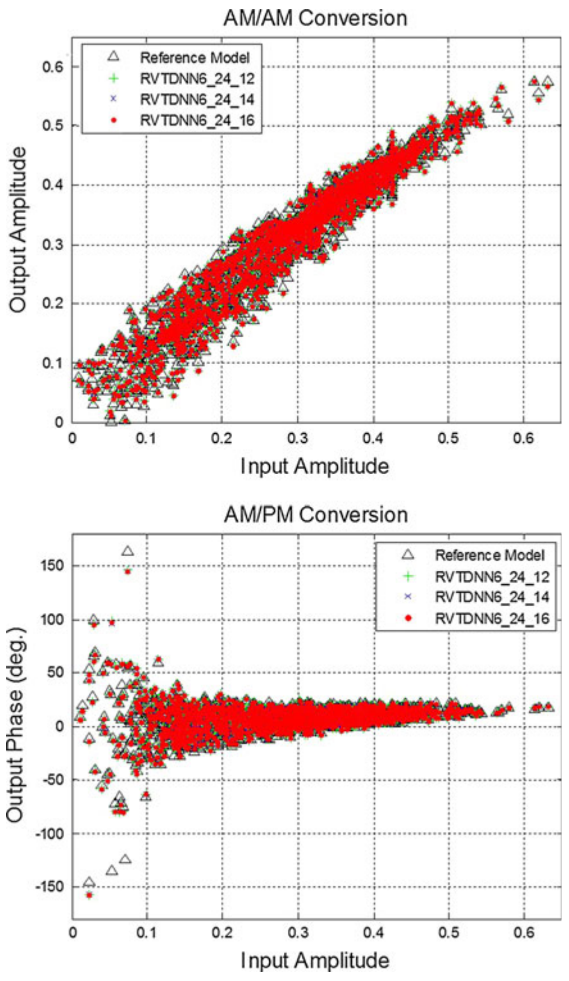

Fig. 10 AM/AM and AM/PM characteristics for the reference and RVTDNN6_24_12, RVTDNN6_24_14 and RVTDNN6_24_16 models

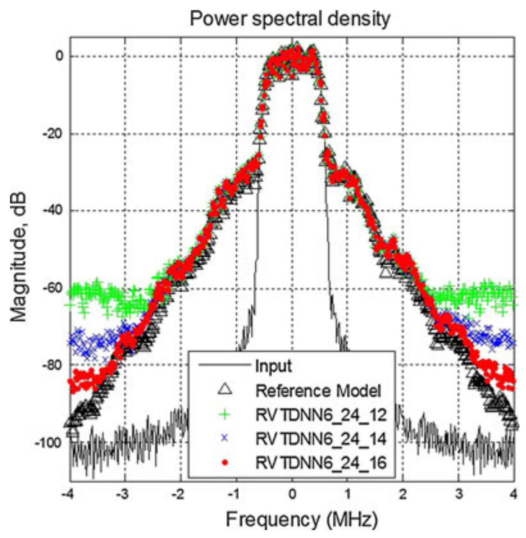

Fig. 11 Power spectral density of input and output signals for the reference and RVTDNN6_24_12, RVTDNN6_24_14 and RVTDNN6_24_16 models

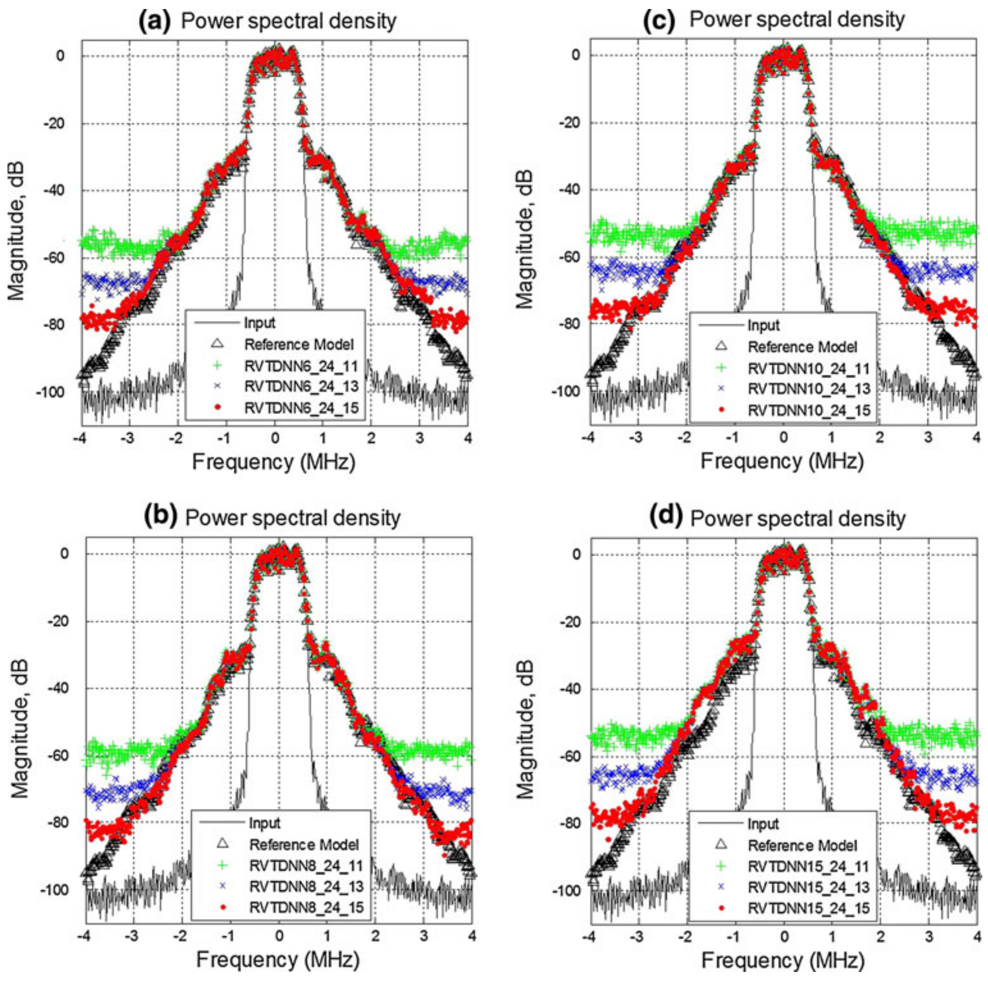

Fig. 12 Power spectral density of input and output signals for the reference and RVTDNN PA models for different numbers of hidden neurons and ROM memory bus sizes using FIX24RS format. For example, RVTDNN6_24_15 represent a model of 6 neurons in hidden layer that process data in fixed-point format of 24 bits and uses 15 bits to address the tanh ROM memory (e.g., size of 2^{15} words)

Figure 12 shows the power spectrum density of input and output signals for the reference and RVTDNN PA models for different numbers of hidden neurons and ROM memory sizes using FIX24RS format. For the same ROM memory size, the PSD curves obtained with RVDTNN models having different numbers of hidden neurons are compared with that obtained by the reference model. The closest PSD curve is obtained with the RVTDNN8_24_15 model that includes 8 neurons in the hidden layer and uses 15 bits to address the tanh ROM memory (Fig. 12b).

By analyzing the PSD curves obtained with the FIX24RS fixed-point format for different RVTDNN sizes (Figs. 11 and 12), it is clear that the best performances are obtained by RVTDNN6_24_16 and RVTDNN8_24_15 PA models. These models gives approximately the same PSD curves than that obtained with the RVTDNN6_36_16 model (Fig. 7). However, the RVTDNN6_24_16 and RVTDNN8_24_15 PA models require less hardware resources (DSP48E1 and RAM36E1) and present a higher maximum operating frequency than the RVTDNN6_36_16 (about 28.5 MHz rather 18.725 MHz). See Tables 1 and 2 for more details. Compared to RVTDNN8 _24_15, RVTDNN6_24_16 model uses less DSP48E1s (256 rather 340) because it includes less hidden neurons but it needs more RAM36E1s (264 rather 176) because it used one additional bit to address ROMs. The choice between the last two models will be determined by the resources (DSP48E1s and RAM36E1s) available on the used FPGA.

5 Conclusion

Neural networks have been successfully implemented on Virtex-6 XC6VLX240T chip for dynamic PA behavioral modeling. The obtained simulation results with the 16-QAM modulated signal show the high performance are obtained with low-cost architectures. Compared to existing software-based models [3–6], the proposed hardware-based architectures guaranties comparable performances using fixed-point format of only 24 bits. Moreover, dynamic modeling is provided by continuous update of the synoptic weights to prevent any eventual change in PA's characteristics.

In the future, the maximum operating frequency of the prosed architectures will be improved by pipelining [17]. Therefore, they can be applied in larger baseband digital predistortion for dynamic linearization of PAs.

References

1. Zhu, A., Pedro, J., & Brazil, T. (2006). Dynamic deviation reduction-based Volterra behavioral modeling of RF power amplifiers. *IEEE Transactions on Microwave Theory and Techniques*, *54* (12), 4323–4332.
2. Ku, H., & Kenney, J.S. (2003). Behavioral modeling of nonlinear RF power amplifiers considering memory effects. *IEEE Transactions on Microwave Theory and Techniques*, *51*(12), 2495–2504.
3. Cao, Y., Chen, X., & Wang, G. (2009). Dynamic behavioral modeling of nonlinear microwave devices using real-time recurrent neural network. *IEEE Transactions on Electron Devices*, *56*(5), 1020–1026.
4. Liu, T., Boumaiza, S., & Ghannouchi, F. M. (2004). Dynamic behavioral modeling of 3G power amplifiers using real-valued time-delay neural networks. *IEEE Transactions on Microwave Theory and Techniques*, *52*(3), 1025–1033.
5. Mkadem, F., & Boumaiza, S. (2011). Physically inspired neural network model for RF power amplifier behavioral modeling and digital predistortion. *IEEE Transactions on Microwave Theory and Techniques*, *59*(4), 913–923.
6. Rawat, M., Rawat, K., & Ghannouchi, F. M. (2010) Adaptive digital predistortion of wireless power amplifiers/transmitters using dynamic real-valued focused time-delay line neural networks. *IEEE Transactions on Microwave Theory and Techniques*, *58*(1), 95–104.
7. Lee, K. C., & Gardner, P. (2006). Adaptive neuro-fuzzy inference system ANFIS digital predistorter for RF power amplifier linearization. *IEEE Transactions on Vehicular Technology*, *55* (1), 43–51.
8. Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *2*(4), 303–314.
9. Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.
10. Chen, Q., Chan, Y. W., & Worden, K. (2003). Structural fault diagnosis and isolation using neural networks based on response-only data. *Computers & Structures*, *81*(22–23), 2165–2172.
11. Bahoura, M., & Park, C.-W. (2011). FPGA-implementation of an adaptive neural network for RF power amplifier modeling. In: *9th IEEE Iiternational NEWCAS conference*, Bordeaux, France, June 2011 (pp. 29–32).
12. Cherubini, D., Fanni, A., Montisci, A., & Testoni, P. (2005). A fast algorithm for inversion of MLP networks in design problems. *COMPEL-The International Journal for Computation and Mathematics in Electrical and Electronic Engineering, 24*(3), 906–920.
13. Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd edn). Upper Saddle River, NJ: Prentice-Hall.
14. Saleh, A. A. M. (1981). Frequency-independent and frequency-dependent nonlinear models of TWT amplifiers. *IEEE Transactions on Communications, 29*(11), 1715–1720.
15. Bastos, J. L., Figueroa, H. P., & Monti, A. (2006). FPGA implementation of neural network-based controllers for power electronics applications. In *Twenty-first annual IEEE applied power electronics conf. and Exp.*, 2006. APEC '06 (pp. 1443–1448).
16. Bahoura, M., & Ezzaidi, H. (2011). FPGA-implementation of parallel and sequential architectures for adaptive noise cancelation. *Circuits, Systems, and Signal Processing*, *30*(6), 1521–1548.
17. Bahoura, M., & Park, C.-W. (2011). FPGA-implementation of high-speed MLP neural network. In *18th IEEE international conference on electronics, circuits, and systems, ICECS 2011*, Beirut, Lebanon, Dec 2011 (pp. 426–429).

Mohammed Bahoura received the B.Sc. degree in Electronic Engineering from the Université des Sciences et de la Technologie of Algiers, Algeria, in 1990, the M.Sc. degree in Instrumentation and Control from the Université de Rouen, France, in 1994 and the Ph.D. degree in Electrical Engineering, from the same University, in 1999. He was a Postdoctoral Fellow at the Université du Québec àChicoutimi, Canada, from 1999 to 2001. Since 2001, he has been with the Université du Québec à Rimouski, where he is a Professor of Electrical Engineering. His research interests focus on digital signal processing, biomedical engineering, speech enhancement, underwater acoustic, and hardware implementation on FPGAs.

Chan-Wang Park received the B.Sc. degree in Electronic and communication from Han Yang University, in Seoul, Korea in 1981 and M.Sc. degree from E.N.S.T. de Paris in France and Ph.D. degree from IEMN, Université des sciences et technologies de Lille 1 in France 1996 and 2001 respectively. From 1981 to 1985 he was in R&D center with Gold Star Electric (L.G. Telecom) at AnYang in Korea. From 1985 to 1990 he was in Training Centre in Saudi Telecom as a Technical Instructor. From 1990 to 1994 he developed three satellites at Satellite Technology Research Centre in KAIST, in Korea as a project manager. From 1994 to 1995 he was with IAS and Matra Marconi Space in France. From 2001 to 2002 he was with AmpliX and Mitec Telecom in Canada. In 2003, he joined the Université du Québec à Rimouski, where he is currently a professor. His research area is RF/microwave and millimeter wave components and system for wireless communication.