Power Amplifier Behavioral Modeling by Neural Networks and their Implementation on FPGA

Roger Sandrin Ntouné Ntouné
Department of Engineering,
Université du Québec à Rimouski,
300, allée des Ursulines,
Rimouski, Qc, Canada, G5L 3A1.
Email: ntor0001@uqar.qc.ca

Mohammed Bahoura
Department of Engineering,
Université du Québec à Rimouski,
300, allée des Ursulines,
Rimouski, Qc, Canada, G5L 3A1.
Email: Mohammed_Bahoura@uqar.qc.ca

Chan-Wang Park
Department of Engineering,
Université du Québec à Rimouski,
300, allée des Ursulines,
Rimouski, Qc, Canada, G5L 3A1.
Email: Chan-Wang_Park@uqar.qc.ca

***Abstract*— In this paper, field programmable gate array (FPGA) implementation of two power amplifier (PA) dynamic behavioral modeling approaches with real-valued time-delay neural network (RVTDNN) and real-valued recurrent neural network (RVRNN) architectures are presented. The proposed PA models are based on the multilayer perceptron (MLP) neural networks with delayed inputs to take into account nonlinearity and memory effects of the PA. The synoptic weights of these neural networks are dynamically updated in order to prevent any eventual change in the PA's characteristics. Both architectures have been optimized to include only six hidden neurons and implemented on FPGA using Xilinx system generator. The FPGA is preferred to the digital signal processor (DSP) because it allows parallel computation tasks and software like flexibility. The modeling performances of these architectures are compared using 16-QAM modulated test signal. The mean square error (MSE) between the desired and the actual outputs of these models are also compared.**

I. Introduction

The power amplifier (PA), which amplifies a signal before it can be transmitted over a wireless communication channel, is a component that presents important nonlinearities in its saturation zone. These nonlinearities and the memory effects of PAs introduce distortions in the transmitted signal. The memory effects can be classified into two categories: the electro-thermal memory effects and the electric memory effects that are attributed to the non-constant spectral response around the carrier frequency [1]. These distortions create spectral regrowth in the adjacent channel (ACPR : adjacent channel power ratio) as well as deformations of the modulated signal constellation (EVM : error vector magnitude) [2] in the bandwidth of modern broadband wireless communication systems, such as code division multiple access (cdma2000), wideband code division multiple access (WCDMA) [1], and long term evolution (LTE).

In order to examine the history of the input signal's envelope, it is important to consider the nonlinearities and memory effects modeled by AM/AM and AM/PM characteristics. The easiest method to solve this problem of nonlinearity is to reduce the power level (Back-Off). It achieves high linearity, but it greatly reduces the power added efficiency. To increase efficiency and minimize distortions, many linearization techniques have been proposed in literature such as feedback [3], feedforward [3], linear amplification with nonlinear components (LINC) [4], digital pre-distortion (DPD) that uses Volterra series [2], [5], memory polynomials [6], lookup table (LUT) [7] and neural networks models.

Several approaches based on neural networks for PA modeling and linearizing have been published in the last decade. Liu *et al.* [8] proposed a real-valued time-delay neural network (RVTDNN) suitable for dynamic modeling of the baseband nonlinear behaviors of third-generation (3G) base-station PAs. Through an optimization procedure, an RVTDNN having 15 neurons and 5 tapped delay lines was found to be appropriate for the LDMOS PA driven by 3G signals [8]. Rawat *et al.* [9] proposed an adaptive predistortion technique based on a real-valued focused time-delay neural network (RVFTDNN) for the linearization of 3G PAs. They compared its performances to those of a real-valued recurrent neural network (RVRNN). In fact, the RVFTDNN and RVRNN [9] contained two hidden layers instead of one hidden layer for RVTDNN [8]. By optimization, the numbers of neurons in the two hidden layers were 7 and 15 for the RVFTDNN and 10 and 17 for the RVRNN. For both models (RVFTDNN and RVRNN), 3 tapped delay lines are added in their two baseband inputs [9].

The digital predistortion is in the way of becoming one of the most important linearization techniques due to the availability of faster digital signal processing hardware, such as DSPs, FPGAs and ASICs (Application Specific Integrated Circuits). Manufacturers of chipsets, PA rack systems, and base stations propose different types of DPD solutions [7]. Before pre-distorting the signal, the PA has to be characterized with appropriate model as neural networks. Taking advantage of hardware parallelism, the FPGAs exceed the computing power of DSPs. Also, they allow software like flexibility for reconfigurable architectures. The FPGA has become one of main choices for implementing baseband digital PA modeling [5].

This paper presents RVTDNN and RVRNN models having only six neurons in the hidden layer to minimize the required hardware resources. These models and their back-propagation (BP) learning algorithms were implemented on FPGA using Xilinx System Generator and the Virtex-6 FPGA ML605

978-1-4673-1881-5/12/$31.00 ©2012 IEEE

Evaluation Kit. In Section II, the proposed neural network PA models are briefly introduced and their equations for propagation and back-propagation algorithms are developed. Section III presents the reference PA model. The FPGA implementation is presented in Section IV. The experimental results verifing the proposed architectures are presented in Section V. Conclusions are finally given in Section VI.

II. Modeling by Neural Networks

The RVRNN architecture is directly represented in Fig. 1. However, the RVTDNN is represented in this same figure without shaded boxes. As shown in Fig. 1, the RVTDNN model is based on a multi-layer perceptron (MLP) by adding delays in its input. However, the RVRNN model is based on the RVTDNN by adding delayed feedback from its outputs. For a given time n [10], the baseband signal in the input of each neural networks is defined as a vector $\mathbf{x}$ of $2m+2$ components for the RVTDNN and $2m+4$ components for the RVRNN, where m is memory depth.

- *RVTDNN*

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

- *RVRNN*

$$\begin{aligned}\mathbf{x} = [&I_{in}(n), \ldots, I_{in}(n-m), Q_{in}(n),\\ &\ldots, Q_{in}(n-m), I_{out}(n-1), Q_{out}(n-1)]\end{aligned} \tag{2}$$

where $I_{in}(n)$ and $Q_{in}(n)$ are the in-phase input and the quadrature input, respectively. $I_{out}(n)$ and $Q_{out}(n)$ are the in-phase output and the quadrature output, respectively. Each of these neural networks is a collection of neurons arranged together in layers in feed-forward manner [10]. Each neuron performs a weighted sum of its inputs followed by an activation function. For the PA modeling task, the activation function of the hidden neurons is an hyperbolic tangent function [8], [9]: $\varphi_h(x) = (1-e^{-2x})/(1+e^{-2x})$ and the activation function of output neurons is linear: $\varphi_o(x) = x$.

The construction of the RVTDNN and RVRNN models requires the determination of their connection weights and biases. To do that, weights and biases are randomly initialized using the *randn* function of MATLAB. Then, they are iteratively adjusted by the standard back-propagation training algorithm in order to converge to their optimal values. In this paper, the sequential learning mode is used instead of the batch mode used in the MATLAB-based implementations [8], [9].

A. Propagation

The output of a neuron j, in hidden layer, is given by

$$y_j^h(n) = \varphi_h\left(v_j^h(n)\right) \qquad j = 1, \ldots, N \tag{3}$$

$$v_j^h(n) = \sum_{i=1}^{N_0} w_{j,i}^h(n) x_i(n) + w_{j,0}^h(n) \tag{4}$$

where $w_{j,0}^h(n)$ is the bias of this neuron and $w_{j,i}^h(n)$ represents the connection weight from neuron i of the input layer. N is the number of neurons in the hidden layer ($N = 6$) and N_0 is the size of the input vector $\mathbf{x}$, which is $2m+2$ for RVTDNN and $2m+4$ for RVRNN. Based on the reference model, the memory depth is chosen to be $m = 2$.

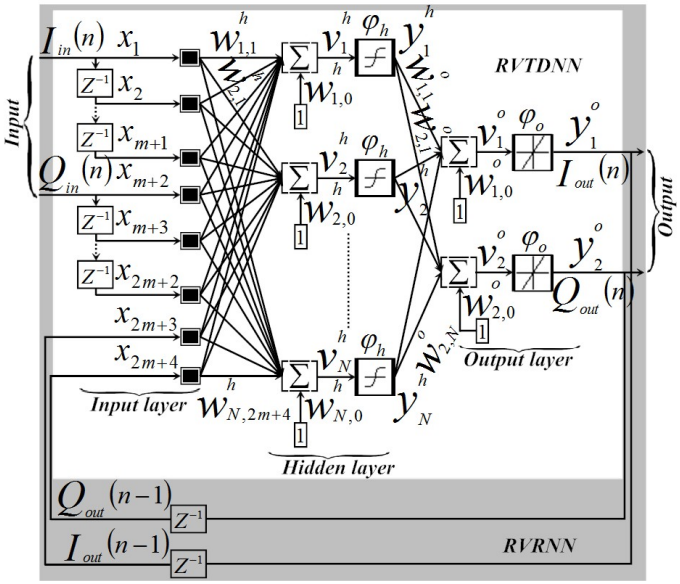

Fig. 1. Structure of the two-layer RVTDNN and RVRNN PAs models.

The output of a neuron k, in output layer, is given by

$$y_k^o(n) = \varphi_o\left(v_k^o(n)\right) \qquad k = 1, 2 \tag{5}$$

$$v_k^o(n) = \sum_{j=1}^{N} w_{k,j}^o(n) y_j^h(n) + w_{k,0}^o(n) \tag{6}$$

where $w_{k,0}^o(n)$ is the bias of this neuron and $w_{k,j}^o(n)$ represents the connection weight from neuron j of the hidden layer.

B. Back-propagation

The error signal $e_k(n)$ is the difference between the desired output and the actual neural network's output [11]:

$$e_k(n) = d_k(n) - y_k(n) \tag{7}$$

The mean square error (MSE) is defined by

$$\text{MSE} = \frac{1}{M} \sum_{n=0}^{M-1} E(n) \tag{8}$$

where $E(n) = \frac{1}{2}(e_1^2(n) + e_2^2(n))$ is the total error energy, and M is the number of samples in each epoch.

1) Output Layer: The correction $\Delta w_{k,j}^o(n)$ applied to the output neuron weights $w_{k,j}^o(n)$ is defined by

$$\Delta w_{k,j}^o(n) = \eta \delta_k^o(n) y_j^h(n) \tag{9}$$

where $\eta = 0.01$ is the *learning rate*. The local gradient is defined by

$$\delta_k^o(n) = e_k(n)\varphi_o'(v_k^o(n)) \tag{10}$$

where $\varphi_o'(v_k^o(n)) = 1$, a linear function is used.
Adjustment $w_{k,j}^o(n+1)$ of the weights is :

$$w_{k,j}^o(n+1) = w_{k,j}^o(n) + \Delta w_{k,j}^o(n) \tag{11}$$

The correction $\Delta w_{k,0}^o(n)$ applied to biases is a particular case of the weight correction $\Delta w_{k,j}^o(n)$ and is given by

$$\Delta w_{k,0}^o(n) = \eta \delta_k^o(n) \tag{12}$$

2) Hidden Layer: The correction $\Delta w_{j,i}^h(n)$ applied to the hidden neuron weights $w_{j,i}^h(n)$ is defined by

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

where the local gradient is given by

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

where $\varphi_h'(v_j^h(n)) = 1 - \varphi_h^2(v_j^h(n))$, an hyperbolic tangent function is used.

The connection weight update at the hidden layer is :

$$w_{j,i}^h(n+1) = w_{j,i}^h(n) + \Delta w_{j,i}^h(n) \tag{15}$$

The correction $\Delta w_{j,i}^h(n)$ applied to biases is:

$$\Delta w_{j,0}^h(n) = \eta \delta_k^h(n) \tag{16}$$

III. Reference Model

The PA reference model that includes nonlinearities and memory effects is based on the Wiener model. It is composed of a 2nd order finite impulse response (FIR) filter followed by nonlinear memoryless Saleh model's [12]. The 2nd order FIR filter is used to model the memory effects, explaining the fact that only a memory depth of 2 is sufficient in this experimentation. Saleh model is defined by the following amplitude-phase and quadrature nonlinear models of a traveling-wave tube (TWT) amplifier. The PA model using Saleh [12] is defined by AM/AM and AM/PM characteristics.

$$A(r(t)) = \frac{\alpha_A r(t)}{1 + \beta_A r^2(t)} \tag{17}$$

$$\Phi(r(t)) = \frac{\alpha_\Phi r(t)}{1 + \beta_\Phi r^2(t)} \tag{18}$$

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 $\beta_\Phi = 9.1040$ [12]. The 2nd order FIR filter defined by [10].

$$H(z) = 1 + 0.5z^{-2} \tag{19}$$

IV. FPGA Implementation

The RVTDNN and RVRNN models and their learning algorithms were implemented on FPGA using Xilinx System Generator for DSP and the Virtex-6 FPGA ML605 Evaluation Kit. Fig. 2 presents the RVRNN architecture of six hidden neurons implemented using Xilinx blockset. After successful simulation of these models, the corresponding bitstream files are automatically created. Table I gives the required resources and the maximum operating frequency obtained with data quantized using 2's complement signed 36-bit fixed point format having 32 fractional bits. However, only 1-bits format is used to represent respectively the constant values of 0 and 1 used to compute the hyperbolic tangent and its derivative. LUT (of 2^{16} samples) is used to implement the hyperbolic tangent function taking advantage from its symmetric characteristic. A delay is added to compensate the one introduced by the ROM block. Not like [10], the RVTDNN-based architectures use a 6-bit right shifter to implement multiplication by the learning rate ($\eta = 0.015625$). These modifications lead to the decreasing of the number of DSP48E1s from 640 to 600 and the increasing of the maximum operating frequency from 13.585 to 21.327 MHz. The operating frequency improvement is also attributed to the critical path reduction in the weight update schema. In the present paper, 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 in [10]. It is noted that the maximum operating frequency of the RVTDNN (21.327 MHz) is slightly higher than that of the RVRNN (21.311 MHz). The RVRNN uses more DSP48E1s (720) than RVTDNN (600). The RVTDNN architecture requires 2,262 Flip-Flops, while the RVRNN architecture uses 2,766. Similarly, the RVRNN uses more LUTs (6,850) than RVTDNN (5,981). However, these two architectures have the same number of Bonded IOBs and RAMBE1s.

V. Experiment and Results

The dynamic behavioral modeling performances of the RVTDNN and RVRNN architectures are compared to the Wiener reference model that includes nonlinearities and memory effects. The 16-QAM modulated test signal is generated using the communication toolbox of MATLAB. As shown in Fig. 3, AM/AM and AM/PM characteristics of the RVRNN is better than those of the RVTDNN. Fig. 4 shows that the RVTDNN power spectrum density provides good accuracy, but, as pointed in [9], the RVRNN in-band data has good modeling capability. The MSE, computed by averaging the results obtained by 10 different learning tests (Fig. 5), shows that the RVTDNN learns slightly faster than RVRNN. Each learning test is done through 90 frames of 8192 samples. These tests differ by the initial values of the synaptic weights. Weights and biases are randomly initialized, using the *randn* function of MATLAB, to show that each algorithm converges regardless the initial parameters. It has been found that using only 6 neurons in the hidden layer allows these models to be successfully implemented in Virtex-6 FPGA ML605

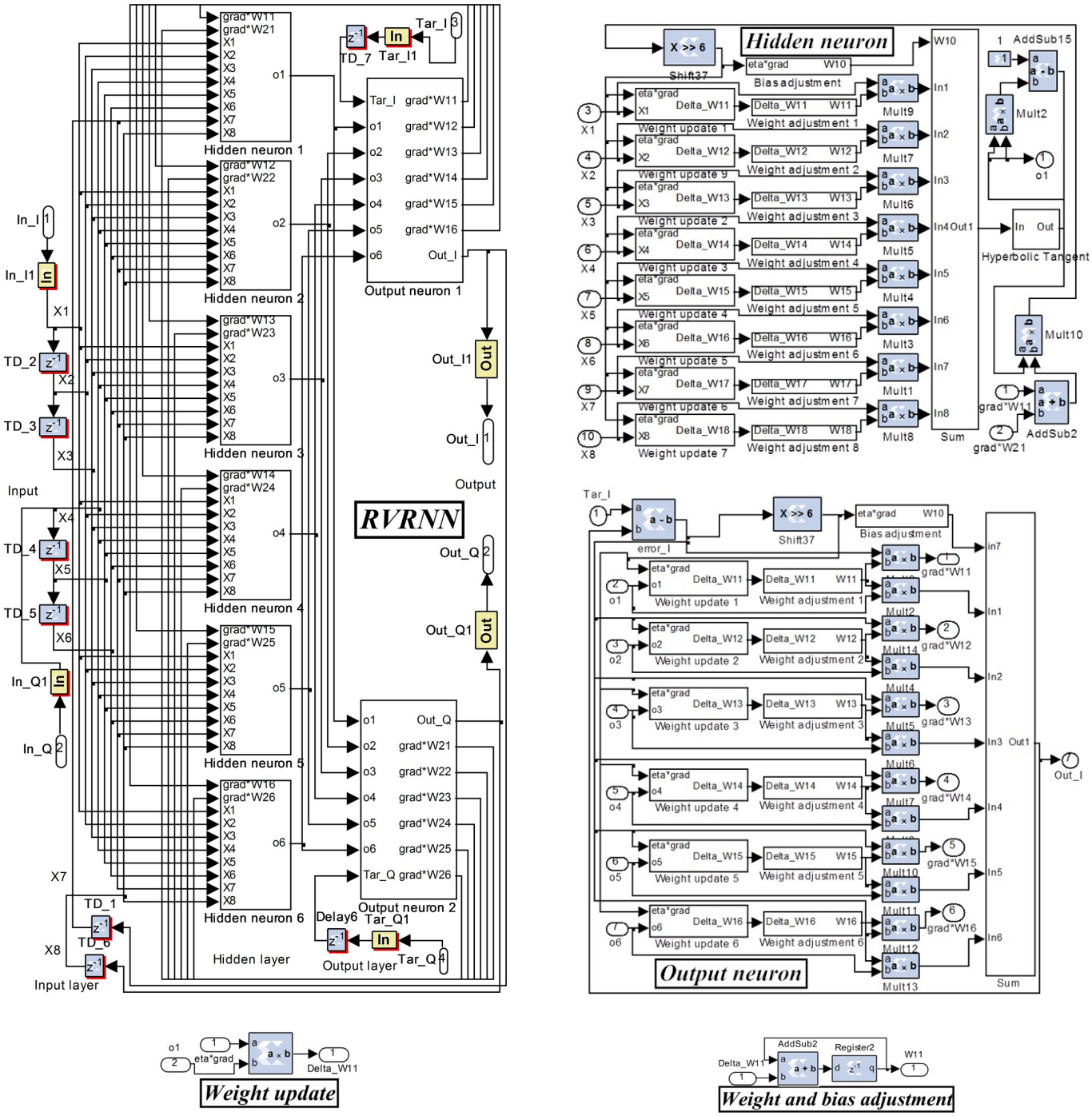

Fig. 2. RVRNN architecture based on Xilinx system generator blockset for RF Power amplifier modeling. Details of other blocks can be found in [10].

Evaluation kit. Popular on-line learning algorithm by Least-mean-square (LMS) is used to calculate the error during the continuous update of the synaptic weights of the BP algorithm.

VI. Conclusion

RVTDNN and RVRNN neural networks for dynamic PA modeling have been successfully implemented on Virtex-6 XC6VLX240T FPGA chip. Despite the fact that the power spectral density of the RVTDNN is closer to the reference model than the RVRNN's one, the AM/AM and AM/PM characteristics of the RVRNN model are more accurate than those of the RVTDNN. The RVRNN requires more resources (Flip-Flops, LUTs, and DSP48E1s) than RVTDNN because of its two additional inputs and, also, presents slightly slower convergence speed than the RVTDNN. However, these two architectures have the same number of Bonded IOBs and RAMBE1s.

TABLE I

RESOURCE UTILIZATION AND MAXIMUM OPERATING FREQUENCY OF THE VIRTEX-6 XC6VLX240T CHIP. RESOURCES AVAILABILITY ARE GIVEN BETWEEN BRACKETS

Architecture	RVTDNN	RVRNN
Resource utilization		
Flip-Flops (301,440)	2,262	2,766
LUTs (150,720)	5,981	6,850
Bonded IOBs (600)	217	217
RAMB36E1s (416)	384	384
DSP48E1s (768)	600	720
Maximum Operating Frequency (MHz)	21.327	21.311

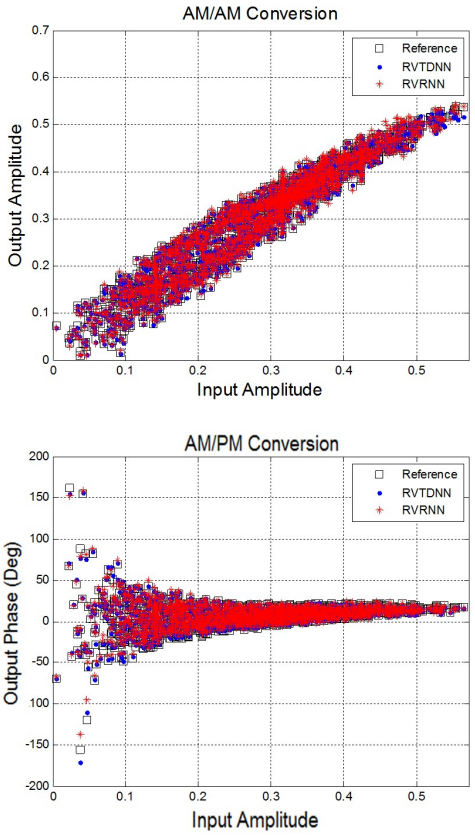

Fig. 3. AM/AM and AM/PM characteristics for the reference, RVTDNN and RVRNN models.

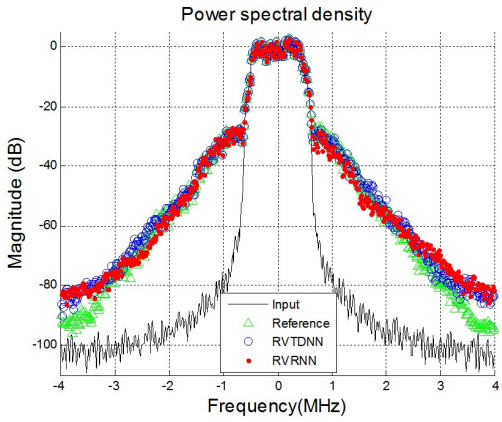

Fig. 4. Power spectral density of input and output signals for the reference, RVTDNN and RVRNN models.

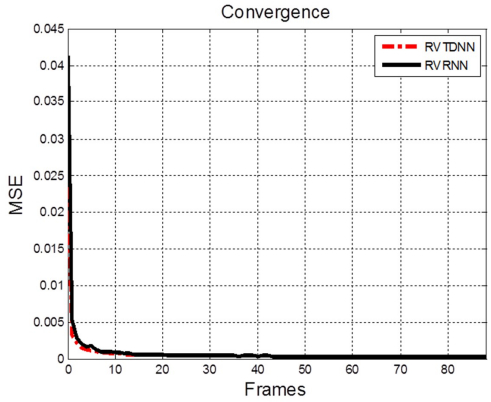

Fig. 5. MSE of the RVTDNN and RVRNN models obtained by averaging the results of 10 different tests. Each learning test is done through 90 frames of 8192 samples.

REFERENCES

[1] T. Liu, Y. Ye, X. Zeng, and F. M. Ghannouchi, "Memory Effect Modeling of Wideband Wireless Transmitters Using Neural Networks," in *proc. 4th IEEE Int. Conf. Circuits and Systems for Communications*, Shanghai, China, 26-28 May 2008, pp. 703–707.

[2] J. Liszewski, B. Schubert, W. Keusgen, and A. Kortke, "Low-complexity FPGA implementation of Volterra predistorters for power amplifiers," in *proc. IEEE Topical Conf. Power Amplifiers for Wireless and Radio Applications*, Phoenix, Arizona, USA, 16-19 Jan. 2011, pp. 41–44.

[3] H. H. Boo, S. Chung, and J. L. Dawson, "Digitally Assisted Feedforward Compensation of Cartesian-Feedback Power-Amplifier Systems," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 58, no. 8, pp. 457–461, Aug. 2011.

[4] M. Helaoui and F. M. Ghannouchi, "Linearization of Power Amplifiers Using the Reverse MM-LINC Technique," *IEEE Trans. Circuits and Systems II: Express Briefs*, vol. 57, no. 1, pp. 6–10, Jan. 2010.

[5] L. Guan and A. Zhu, "Low-Cost FPGA Implementation of Volterra Series-Based Digital Predistorter for RF Power Amplifiers," *IEEE Trans. Microwave Theory and Techniques*, vol. 58, no. 4, pp. 866–872, Apr. 2010.

[6] B. Fehri and S. Boumaiza, "Systematic Estimation of Memory Effects Parameters in Power Amplifiers' Behavioral Models," in *proc. IEEE Int. Microwave Symp. Digest*, Baltimore, USA, 5-10 Jun. 2011, pp. 1–4.

[7] P. L. Gilabert, A. Cesari, G. Montoro, E. Bertran, and J.-M. Dilhac, "Multi-Lookup Table FPGA Implementation of an Adaptive Digital Predistorter for Linearizing RF Power Amplifiers With Memory Effects," *IEEE Trans. Microwave Theory and Techniques*, vol. 56, no. 2, pp. 372–384, Feb. 2008.

[8] T. Liu, S. Boumaiza, and F. M. Ghannouchi, "Dynamic Behavioral Modeling of 3G Power Amplifiers Using Real-Valued Time-Delay Neural Networks," *IEEE Trans. Microwave Theory and Techniques*, vol. 52, no. 3, pp. 1025–1033, Mar. 2004.

[9] M. Rawat, K. Rawat, and F. M. Ghannouchi, "Adaptive Digital Predistortion of Wireless Power Amplifiers/Transmitters Using Dynamic Real-Valued Focused Time-Delay Line Neural Networks," *IEEE Trans. Microwave Theory and Techniques*, vol. 58, no. 1, pp. 95–104, Jan. 2010.

[10] M. Bahoura and C.-W. Park, "FPGA-Implementation of an Adaptive Neural Network for RF Power Amplifier Modeling," in *proc. 9th IEEE Int. Conf. New Circuits and Systems*, Bordeaux, France, 26-29 June 2011, pp. 29–32.

[11] S. Haykin, *Neural Networks and Learning Machines*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 2009.

[12] A. Saleh, "Frequency-Independent and Frequency-Dependent Nonlinear Models of TWT Amplifiers," *IEEE Trans. Communications*, vol. 29, no. 11, pp. 1715–1720, Nov. 1981.