Fixed-point Arithmetic Architecture of a Physically-meaningful Perceptron for Digital Pre-distorters

Juliana C. L. Pereira, Felipe A. Schoulten, Caio P. Mizerkowski, Sibilla B. L. França, Eduardo G. Lima
Grupo de Circuitos e Sistemas Integrados (GICS) – Departamento de Engenharia Elétrica
Universidade Federal do Paraná, Curitiba, Brazil

Abstract—**Three-layer perceptrons (TLPs) are one of the approaches to model the inverse transfer characteristics of power amplifiers (PAs) for digital baseband pre-distortion (DPD). When a set of signals composed of amplitudes and trigonometric functions of phase differences are applied to the TLP inputs, only useful contributions for DPD purposes are generated. In literature, such physically-meaningful TLP has been investigated only in floating-point arithmetic. This work contribution is to present an architecture suitable for fixed-point arithmetic, in which complex operations are replaced by linearly interpolated look-up tables (LUTs). In Matlab software using floating-point arithmetic, a model was first trained to mimic input and output data measured on a LDMOS class AB PA stimulated by a WCDMA envelope. The TLP was then converted to fixed-point arithmetic and translated to VHDL language. Xilinx ISE post-place and route simulations targeting the Xilinx Artix7 XC7A200T FPGA shows that the synthesized code uses 3095 slice LUTs and 3362 flip-flops to provide a modeling accuracy of -37.9 dB in normalized mean square error.**

Keywords—power amplifier, neural network, VHDL.

I. Introduction

The power amplifier (PA) is one of the most important devices in wireless communication systems [1]. In such applications, the PA needs to transmit as much power as possible, to behave linearly and to be efficient. In many situations, the PA nonlinear characteristics cause undesired distortions. A viable work-around can be found through the use of a digital pre-distortion (DPD) technique [2], which consists of a digital system cascaded with the PA that purposefully distorts its input signal. When the distortion caused by the DPD is inversely related to the distortion caused by the PA, the cascade system tends to become linear.

The DPD scheme requires a model able to represent the inverse transfer characteristic of a PA. Three-layer perceptrons (TLPs) are suitable for that purpose [3]. In the customized TLP-based model for DPD purposes presented in [4], the signals used as input were based on the sine and cosine of the difference between two consecutive phase components. However, only results in floating-point arithmetic are reported in [4].

In a practical DPD application, the model must be synthesized in a field programmable gate array (FPGA) using fixed-point arithmetic. In doing that, several practical aspects must be addressed. One of them is how to perform complex mathematical operations, such as square root or division. Besides, when working with binary numbers, the result of a multiplication doubles the quantity of bits in a message. To keep unchanged the number of bits, a mechanism to reduce by half the number of bits after each multiplication is required. Since the application needs to be operational in a frequency on the order of tens of MHz, the time available for the execution of every operation cannot exceed a few tens of ns. In order to make it possible, it is necessary to use the inherent parallel characteristics of FPGAs by breaking the hardware description language code into independent operational blocks that are processed simultaneously. The goal of this work is to present an architecture suitable for the real-time processing in fixed-point arithmetic of such TLP-based model.

This paper will be presented in the following structure. In Section II, the floating-point TLP model will be detailed. In Section III, the fixed-point architecture will be presented. Results from a case study will be reported in Section IV and conclusions will be discussed in Section V.

II. Floating-point TLP-based Architecture

Artificial neural networks (ANNs) are used in many fields of the science and technology and that is a result of their capacities to approximate continuous functions in an arbitrary interval, which has been demonstrated, for the case of the multi-layer perceptron (MLP), in [5]. In the ANN addressed in here, the perceptron, a mathematical concept based in the neuron, is the fundamental structure. The perceptron is a linear separator, which receives an *n*-dimensional input and returns in which side of a hyperplane, defined by the coefficients of the perceptron, the input is. The decision of the perceptron is a result of the sum of the values of the inputs, weighted by a set of coefficients, known as weights, with the addition of an extra input named bias. The result of that operation is transferred to an activation function, normally the Heaviside function in the case of a single perceptron, and the result is a value that indicates in what side of the hyperplane the input is.

With the appropriate training, a perceptron is capable of slicing any linearly separable set of points, but for more complex tasks, as the problem addressed in this work, a more sophisticated tool is necessary. One of these tools is the MLP, which is an extension of the perceptron by the addition of multiple layers composed by many perceptrons. In each of these layers, the input is processed by the sum and the use of an activation function, and passed for the next layer. Additionally to these layers, known as hidden layers, the MLP has an input layer, which receives the input and whose outcome is transferred to the first hidden layer, and the output layer, which is also composed by perceptrons and the final layer of the network.

A special case of the MLP is the three-layer perceptron (TLP), which has the input and output layers, but only one hidden layer. The TLP is the base of the ANN architecture used in this work and its unique hidden layer is sufficient to approximate any continuous function as demonstrated by the universal approximation theorem [5]. The network hidden and output layers have equal characteristics, in particular, they both use the hyperbolic tangent sigmoid as the activation function.

However, the traditional TLP network has a problem to work with complex-valued signals, as is the case of the modeling addressed in this work. A more powerful architecture is used to solve that problem by a preprocessing of the signal and the separation of the output in its real and imaginary parts [4]. The separation is made by the use of two TLP networks, which receive the same set of preprocessed inputs, but the training of the coefficients are made using different parts of the output: one network is trained to approximate the real part and the other to approximate the imaginary part. In both cases, the output phase is modified by the subtraction of the input phase.

The preprocessed input is defined as a vector with the absolute value of the input in the current (*n*) and previous (*n-m*) instants, and the cosine and sine of phase difference between two consecutive instants. The amount of previous instants used in this work is arbitrarily set to 2, which results in 7 inputs to each TLP. However, the implementation of a larger memory length is straightforward. The total amount of coefficients can then be determined as a function of the number of perceptrons (*N*) in the hidden layer, for *N* greater than 1, which results in 9*N*+1 coefficients for each TLP.

III. Fixed-point TLP-based Architecture

Targeting a simpler FPGA implementation, the first step is to convert signals from floating-point into fixed-point arithmetic. This process is done in the following manner. Initially every complex number, representing PA input and output signals, is multiplied by a constant real gain in order to guarantee that all real and imaginary parts lie in the range -1 to 1. It is then assumed that the decimal 1 is represented by 2 to the power of the number of resolution bits. Although this limitation exists on the input and output signals, it is possible that numbers with magnitude greater than one appear at some point inside the neural network model. To represent these higher values, extra bits are added to the left in order to avoid overflow (for example, if 10 is considered to be the highest number, four extra bits are needed, since $2^4 > 10$ and $2^3 < 10$). So the number of bits in fixed-point representation consists of the number of resolution bits plus the extra bits used to avoid overflow plus one sign bit.

Every mathematical operation is done using multipliers, adders or LUTs, with the former two consisting of two inputs and one output. There exist other mathematical functions throughout the neural network (more specifically, the reciprocal, square root and hyperbolic tangent sigmoid functions). These are calculated using LUTs and linear interpolation. Since creating a LUT that addresses every possibility is considerably arduous work that would waste too much processing time in an application that is very time sensitive, especially in a case with signals represented by a high number of bits, the number of bits used to address the LUT is reduced. For every address there is two correspondent coefficients, *a* (angular) and *b* (linear), that are used to compute a linear function in the form of *au* + *b*., where *u* is the input value. Therefore, the calculation of each of these functions is performed by doing a reading of two values stored in a LUT, one multiplication and one sum.

When a binary multiplication is done, the number of bits is doubled. To keep the system consistent, this result needs to have *N* bits instead of *2N*. Recalling that each number is formed by a sign bit, resolution bits and extra bits to avoid overflow, and also that every number lies between -1 and 1, the solution to keep each number with *N* bits is to round them by discarding some of the least significant resolution bits and some of the most significant overflow bits.

The execution of each of these operations takes time. There is not much that can be done in a single clock cycle of a few tens of ns. Only one operation can be executed in a clock cycle, which means they all need to be executed at the same time. This justifies the use of a dedicated hardware design described in VHDL. This allows the logic to be divided into many blocks, each of them responsible for one operation, which consequently lets for all of them to be processed at the same time. Given that this separation is made, it becomes possible to create a structure that can operate at high sampling frequencies.

The block diagram that represents the introduced fixed-point arithmetic architecture for the TLP-based model is shown in Fig. 1, where the complex-valued input and output signals are indicated by *x* and *s*, respectively, the present instant is [*n*], and the past instants are [*n*-1] and [*n*-2].

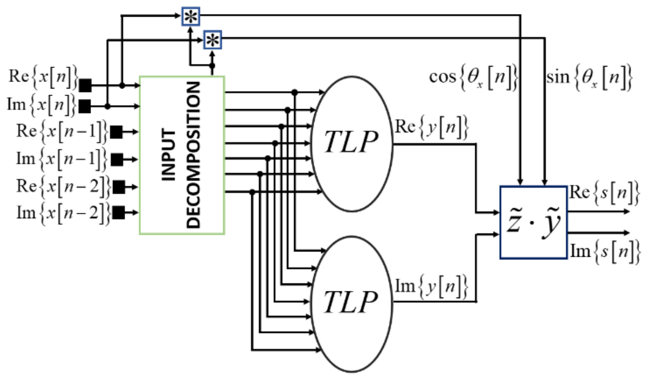

Fig. 1. TLP-based block diagram.

The block diagram of Fig. 1 can be divided into three segments. The first one calculates the seven signals that are used as inputs to both TLPs. The second one computes the outputs of the TLPs themselves, which correspond to the real and imaginary parts of the complex number *y*[*n*]. Finally, the third one calculates the real and imaginary parts of the output signal.

In what concerns the first segment, the real and imaginary parts of the complex-valued inputs *x*[*n*], *x*[*n*-1] and *x*[*n*-2] are used to calculate the seven inputs for the TLPs, according to Fig. 2.

The process starts, in the first clock cycle, by calculating the squares of each input (both real and imaginary parts) using multipliers. In the second clock, the sums of real and imaginary parts are calculated. To compute the square roots, as mentioned before, linearly interpolated LUTs are used. In the third clock cycle, line coefficients *a* and *b* previously stored in the LUTs are read. The fourth clock cycle is responsible for the multiplications of the angular coefficients (*a*) and their respective LUT inputs and the fifth clock cycle for the additions of the linear coefficients (*b*) to their respective results of the previous cycle.

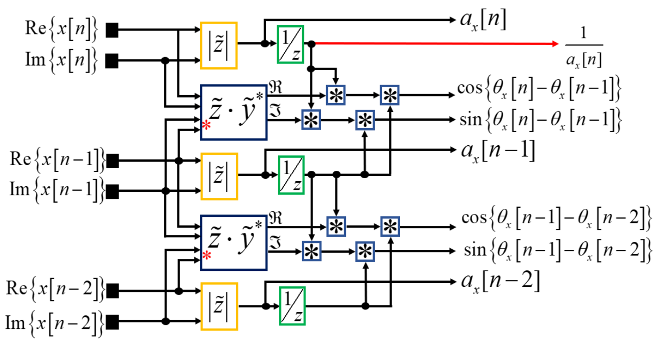

Fig. 2. Detailed diagram of the block called input decomposition in Fig. 1.

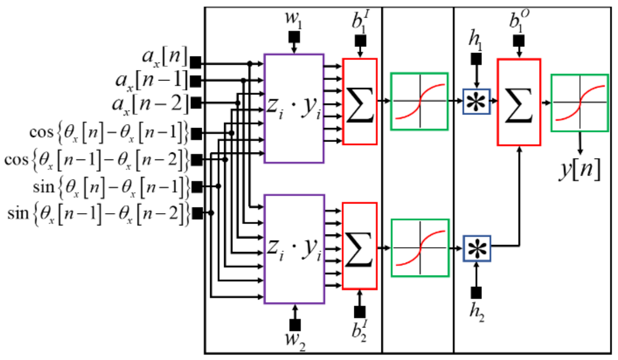

Fig. 3. Detailed diagram of the TLP structure.

With the absolute values calculated, the sines and cosines could be obtained. Recall that these are trigonometric functions of an angle difference. Euler's Formula defines a complex number as

$$e^{j\theta n} = \cos(\theta n) + j\sin(\theta n). \quad (1)$$

Using exponential properties, it can be shown that

$$e^{j\theta n}.\, e^{-j\theta n-1} = e^{j(\theta n-\theta n-1)}$$
$$e^{j\theta n}.\, e^{-j\theta n-1} = \cos(\theta_n - \theta_{n-1}) + j\sin(\theta_n - \theta_{n-1}). \quad (2)$$

Because either exponential or trigonometric functions are not synthesizable (i.e. they cannot be directly translated into a digital logic), the expression $e^{j\theta n}.\, e^{-j\theta n-1}$ needed to be described using basic logics, such as adders and multipliers, in a way to achieve the same functionalities. By definition, it is given that

$$\frac{x(n)}{|x(n)|} = e^{j\theta n}, \quad (3)$$

where $x(n)$ is a complex number. Replacing (3) in (2),

$$e^{j\theta n}.\, e^{-j\theta n-1} = \frac{x(n)}{|x(n)|}.\frac{x^*(n-1)}{|x(n-1)|} = \frac{x(n).x^*(n-1)}{|x(n)|.|x(n-1)|}, \quad (4)$$

where ()* is the conjugate operator. The multiplication between two complex numbers is equivalent to four multiplications and two additions between two real numbers.

Since the values of $x(n)$ and $x(n$-1) are applied as inputs, and hence do not depend on any value from previous clocks, the multiplications and additions involving them are done in the first and second clock cycles and stored to be used later. Since the next operation is a reciprocal function, a LUT is used to compute the result of $\frac{1}{|x(n)|}$ and $\frac{1}{|x(n-1)|}$. It went from the sixth to eighth clock cycles and it is done in the same way the square root was calculated. Then, the ninth clock cycle is necessary to calculate

$$v'(n) = x(n).x^*(n-1).\frac{1}{|x(n)|} \quad (5)$$

and the tenth for

$$v(n) = v'(n).\frac{1}{|x(n-1)|}. \quad (6)$$

All of these steps are repeated for the trigonometric functions for $\theta_{n-1} - \theta_{n-2}$.

Now that the TLP inputs are calculated, the second segment, detailed in Fig. 3, begins to be executed.

The perceptron is divided in three layers and has two neurons. The first layer is responsible for doing a linear combination with the weight vectors w_1 and w_2 (one corresponding to each input, both with 14 elements) and the biases b_1 and b_2 (one corresponding to each neuron). The operations involved here are 28 multiplications (one for each input) that are computed in the 11th clock cycle, and the sum of these values with the corresponding bias. Adding eight values (seven inputs plus bias) in a single clock cycle is not viable for the desired frequency, so this operation was divided into three clock cycles (12th through 14th). The hidden layer takes the result from the first layer and passes it through a hyperbolic tangent sigmoid function. Such as the square root and reciprocal functions, it is calculated using a LUT and linear interpolation, and this is done from the 15th through 17th clock cycles. The third and last layer takes the results from the second layer and multiplies each one by the weights h_1 and h_2 (18th clock cycle), and then adds the result to an extra bias (19th cycle). The perceptron output is done from 20th to 22nd clock cycles and is calculated from the hyperbolic tangent sigmoid function via linearly interpolated LUTs.

These operations are repeated for both perceptrons, and their outputs correspond to the real and imaginary parts of a complex number. What distinguishes one perceptron from the other is the numeric values of the weights and biases.

Now, with the outputs from the perceptron, it starts the third segment, which computes the output from the whole model. For it to be done, it is necessary to multiply the complex number formed by the outputs of the perceptrons by the sine and cosine from the time instant n.

From (1) and (3), the sine and cosine are given by

$$\cos(\theta_x[n]) = \frac{\mathrm{Re}\{x[n]\}}{|x[n]|} \quad (7)$$

$$\sin(\theta_x[n]) = \frac{\mathrm{Im}\{x[n]\}}{|x[n]|} \quad (8)$$

Since $\frac{1}{|x[n]|}$ was already calculated, there is no need to do it again. Because both sine and cosine depend only on the value of $|x[n]|$, they were previously calculated and stored.

Finally, the following complex multiplication is done

$$\begin{aligned} s[n] &= [\mathrm{Re}\{y[n]\} + j\mathrm{Im}\{y[n]\}][\cos(\theta_x[n]) + j\sin(\theta_x[n]] \\ &= [\mathrm{Re}\{y[n]\}.\cos(\theta_x[n]) - \mathrm{Im}\{y[n]\}.\sin(\theta_x[n])] + \\ & j[\mathrm{Im}\{y[n]\}.\cos(\theta_x[n]) + \mathrm{Re}\{y[n]\}.\sin(\theta_x[n])] \quad (9) \end{aligned}$$

The output $s[n]$ is then obtained in the 23rd cycle, where the multiplications are calculated, and 24th, where the result of the neural network is calculated from the additions.

IV. Simulation Results

The presented architecture is now applied to the inverse modeling of a class AB Si laterally diffused metal oxide semiconductor (LDMOS) PA. A 2 GHz carrier modulated by a 3.84 MHz 3GPP Wideband Code Division Multiple Access (WCDMA) envelope feeds the PA. An Agilent MXA N9020A vector signal analyzer (VSA) is used to collect PA output envelope sampled at 30.72 MHz. In Matlab software using floating-point arithmetic, several TLP-based models are trained using nonlinear least squares based on the Levenberg-Marquardt. Table I shows the normalized mean square error (NMSE) as a function of the number of hidden neurons. Only slightly improvements are observed for larger values of N. The NMSE for $N = 8$ is equal to -39.6 dB.

TABLE I. Floating-Point Simulation Results

Number of hidden neurons	NMSE (dB)
1	-28.1
2	-38.1
3	-39.4

A TLP-based model with 2 neurons is chosen to be converted to fixed-point arithmetic. Table II reports the fixed-point NMSE results in relation of the number of resolution bits, which the sign bit is not included, and of LUT addressable bits for the square root, reciprocal and hyperbolic tangent sigmoid functions. From Table II, it is noted that its last row case presented the best compromise between accuracy and complexity, such that this case is chosen for a hardware description language realization. Figure 4 presents the measured and estimated output amplitude as function of the input amplitude. Figure 5 presents the measured and estimated output power spectral densities (PSDs). From Fig. 4, it is observed that the estimated and measured signals show a PA inverse behavior in which both are very similar to each other. From Fig. 5, it is noticed that outputs present small amount of distortions, although the estimated samples have greater levels.

TABLE II. Fixed-Point Simulation Results

Number of resolution bits	Number of LUT addressable bits			NMSE (dB)
	Square-root	Reciprocal	Hyperbolic tangent sigmoid	
22	10	9	8	-31.9
23	10	9	8	-36.5
24	10	9	7	-36.3
24	10	9	8	-37.8
25	9	9	8	-37.7
25	10	9	6	-28.6
25	9	8	8	-37.7
25	10	9	8	-37.9

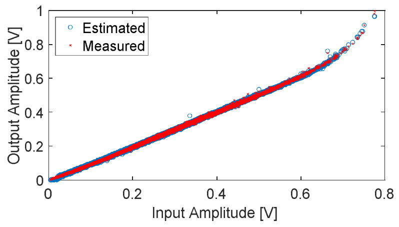

Fig. 4. Output amplitudes as a function of input amplitudes.

The fixed-point architecture is described using VHDL. To check the validity of the described circuit, a behavioral simulation is done on the ISE Software using the ISim Simulator to verify if the values obtained in the VHDL code matched the values obtained via MATLAB. All the results match, proving the correctness of the implemented VHDL code. A post-place and route simulation is also done, using the Xilinx Artix7 XC7A200T FPGA as target device. The natural delays of the operations are below one clock time. If a discrete-time sequence of length L with a sampling interval of 32.55 ns is applied at the input, then an output sequence with same length and sampling frequency will be available after a constant delay of 781.2 ns. Besides, the synthesized code used 3095 slice LUTs and 3362 flip-flops.

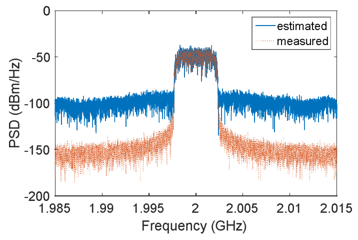

Fig. 5. PSDs of output signals.

V. Conclusions

This work has addressed important aspects related to the fixed-point arithmetic description of a TLP-based model suitable for PA linearization purposes. The generation of sine and cosine of phase differences has been performed using the amplitude information together with a reciprocal function and some additional multiplications and sums. Square root, reciprocal and hyperbolic tangent sigmoid functions were replaced by lines with distinct slopes, that were performed by reading two line coefficients previously stored in LUTs, followed by a multiplication and an addition. Considering the high number of successive multiplications demanded by the TLP-based model and that outcomes from two input multipliers doubles the quantity of bits, after each multiplication half of the bits were removed in a way to minimize the degradation in modeling accuracy. Each basic operation was implemented by an exclusive hardware and parallelism was fully exploited to guarantee the fulfillment of the real-time requirements. Simulation results from a case study show that a modeling accuracy of -37.9 dB in NMSE was achieved when the synthesized VHDL code requires 3095 slice LUTs and 3362 flip-flops.

Acknowledgment

The authors would like to acknowledge the financial support provided by Programa de Iniciação Científica da Universidade Federal do Paraná, UFPR/Tesouro Nacional, and by Pró-Reitoria de Assuntos Estudantis (PRAE-UFPR).

References

[1] S. Cripps, *RF Power Amplifiers for Wireless Communications*, 2nd edition. Norwood, MA: Artech House, 2006.

[2] P. B. Kenington, *High Linearity RF Amplifier Design*. Norwood, MA: Artech House, 2000.

[3] J. C. Pedro and S. A. Maas, "A comparative overview of microwave and wireless power-amplifier behavioral modeling approaches," *IEEE Trans. Microw. Theory Tech.*, vol. 53, no. 4, pp. 1150–1163, Apr. 2005.

[4] L. B. C. Freire, C. Franca, and E. G. Lima, "A Modified Real-Valued Feed-Forward Neural Network Low-Pass Equivalent Behavioral Model for RF Power Amplifiers," *Prog. Electromagn. Res. C*, vol. 57, 43–52, 2015.

[5] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.