Nonlinear behavioral modeling of oscillators in VHDL-AMS using artificial neural networks

Michael Kraemer, Daniela Dragomirescu, Robert Plana

Nonlinear Behavioral Modeling of Oscillators in VHDL-AMS using Artificial Neural Networks

M. Kraemer, D. Dragomirescu, R. Plana

LAAS-CNRS, University of Toulouse, 8 Avenue de Colonel Roche, 31077 Toulouse, France
Email: {mkraemer, daniela, plana}@laas.fr

***Abstract* — In this paper an approach to behavioral modeling of microwave oscillators is described. The presented model takes into account start-up, steady state behavior and phase noise. To describe the nonlinearities, an Artificial Neural Network (ANN) is employed. The dynamic behavior of the oscillator is described by a system of differential equations that are solved in VHDL-AMS. As opposed to input-output models of microwave devices, this paper presents a self sustaining oscillation, which starts from a small injected excitation (e.g. noise) and ends in a stable limit cycle. Additionally, the phase noise characteristics of the oscillator in the $1/f^2$ and flat region are emulated.**

***Index Terms* —VHDL-AMS, behavioral modeling, oscillator, artificial neural network, black box model.**

I. Introduction

In the design process of analog and mixed-signal integrated circuits, behavioral modeling becomes indispensable. A hardware description language that permits the simulation of entire systems is VHDL-AMS [1]. It incorporates both digital and analog modeling capabilities. Our goal is to create accurate VHDL-AMS models of all components of a 60 GHz transceiver front-end to be capable to simulate a whole System on Chip.

There exist a multitude of different approaches to model nonlinear components on the system level (e.g. [2]). Due to the use of VHDL-AMS, our models need to operate in the time domain. In order to be able to model not only weak nonlinearities (e.g. amplifiers), but also strongly nonlinear devices (e.g. oscillators and mixers), a black box approach that describes the output waveforms for different states and inputs seems sensible.

In the context of large signal network analysis (LSNA), black box modeling using neural networks is proposed. The dynamics of the circuit are either contained by using delays [3], or in form of a differential equation [4]. For LSNA, modeling of the noise characteristics is of no interest, yet it is indispensable for simulating a transceiver system and is thus considered in the presented model.

This paper extends the methods described in [5] to model an oscillator in VHDL-AMS. The nonlinearity is reproduced by an ANN. The system of differential equations implicitly contains the feedback inside the oscillator. The oscillation starts up from a random signal injected into an artificial input port. Furthermore, this random signal is used to generate phase noise in the $1/f^2$ region.

II. Theoretical Background

A. State Space Representation

A nonlinear, time invariant dynamical system can be represented by the state equation

$$\dot{\mathbf{x}}(t) = \Phi\big(\mathbf{x}(t), \mathbf{u}(t)\big) \tag{1}$$

and the input equation

$$\mathbf{y}(t) = \Psi\big(\mathbf{x}(t)\big) \tag{2}$$

where the vector $\mathbf{x}(t)$ of size N_x represents the state of the system. The N_x-dimensional space on which $x(t)$ is defined is called the systems state space (e.g. [6]). The input is represented by the vector $\mathbf{u}(t)$ of size N_u, the output by $\mathbf{y}(t)$ of size N_y. N_x represents the number of independent energy storages and is usually much larger than N_y. $\Phi(\,)$ and $\Psi(\,)$ are generally nonlinear functions.

To get a black box model, the internal states of the system are not of interest. We need to know the behavior of the inputs $\boldsymbol{u}(t)$ over time, described by the input signals and their derivatives to determine the outputs $\boldsymbol{y}(t)$. The derivatives of the output $\boldsymbol{y}(t)$ are also necessary to embody the input-independent notion of memory. In systems with feedback (like oscillators) the dependence of $\boldsymbol{y}(t)$ on its own can be embodied by an implicit formulation of the system equations.

The state variables of a black box model of (reduced) order N can then be defined by

$$\begin{bmatrix} \mathbf{x}_1(t) \\ \mathbf{x}_2(t) \\ \vdots \\ \mathbf{x}_N(t) \end{bmatrix} = \begin{bmatrix} \mathbf{y}(t) \\ \dot{\mathbf{y}}(t) \\ \vdots \\ \mathbf{y}^{(N-1)}(t) \end{bmatrix}, \tag{3}$$

where the output equation is already contained in the first line of (3). The notation $\mathrm{y}(t)^{(N)} = (\mathrm{d}t)^N/(\mathrm{d}y(t))^N$ is used. The state equation of the black box model is

$$\begin{bmatrix} \dot{\mathbf{x}}_1(t) \\ \vdots \\ \dot{\mathbf{x}}_N(t) \end{bmatrix} = \begin{bmatrix} \mathbf{x}_2(t) \\ \vdots \\ \mathbf{f}\left(\mathbf{x}_1(t) \ldots \mathbf{x}_N(t), \mathbf{u}(t) \ldots \mathbf{u}^{(M)}(t)\right) \end{bmatrix}. \quad (4)$$

The values N and M correspond to the number of derivatives of the out- and input signal, respectively. To obtain the values of N and M, the a priori knowledge about oscillators can be used.

The function $f(\ )$ describes the nonlinear behavior of the system. After defining the inputs, outputs, and the number of their derivatives, this function needs to be found for the particular system under observation.

B. The Van der Pol-Oscillator

To get a first impression of nonlinear oscillations described in state space, the Van der Pol-equation is considered. It describes the voltage $v(t)$ at the output of a triode oscillator in form of a second order differential equation ($N = 2$, $N_y = 1$). The nonlinear triode characteristic of this oscillator is approximated by a polynomial. The definition of the associated state variables is given by

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \dot{v}(t) \end{bmatrix}. \quad (5)$$

The explicit knowledge of $\ddot{v}(t)$ allows us to define

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ \alpha\left(1 - x_1(t)^2\right)x_2(t) - \omega^2 x_1(t) \end{bmatrix}. \quad (6)$$

Because $N_u = 0$, equations (5) and (6) do not describe an input-output relationship, but an oscillation starting at the unstable singular point $\dot{\mathbf{x}}(t) = \mathbf{0}$. If a small disturbance occurs in this state, the oscillator will leave the singular point and follow a trajectory into a stable, attractive limit cycle (cf. Fig. 1). The disturbance starting the oscillator can be numerical inaccuracy, or noise added to one of the components of x(t).

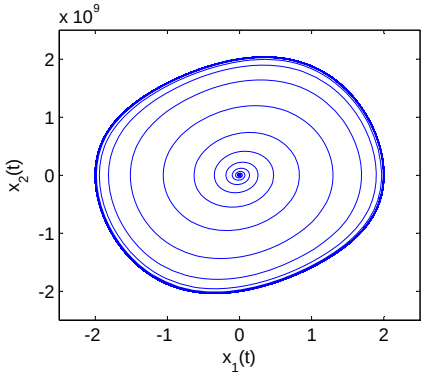

Fig.1. Plot of the trajectory of a Van der Pol-oscillator, simulated in VHDL-AMS. Singular point $\dot{\mathbf{x}}(t) = \mathbf{0}$ for $\mathbf{x}(t) = \mathbf{0}$.

C. Neural Networks

To obtain a flexible model, the function $f(\ )$ is described by a single layer perceptron ANN [6]. The approximation theorem states that such a network can approximate any nonlinear function $f(\ )$ arbitrarily well, provided the number of neurons (i.e. nodes) is large enough [6]. The perceptron consists of N_{in} input neurons, N_{h} neurons in a hidden layer, and N_{out} output neurons (cf. Fig. 2). While the value n_{in} of an input node represents the associated input signal at a certain time, the values of the nodes in the two other layers (n_{h}, n_{out}) are calculated according to Fig. 2 by

$$n_{\text{h}}(k) = \text{tansig}\left[\sum_{\text{j=1}}^{N_{\text{in}}} (\text{n}_{\text{in}}(j) iw(j,k)) + ib(k)\right] \quad (7)$$

and

$$n_{\text{out}}(k) = \text{tansig}\left[\sum_{\text{j=1}}^{N_{\text{h}}} (\text{n}_{\text{h}}(j) lw(j,k)) + lb(k)\right], \quad (8)$$

respectively. The weights iw, lw and biases ib, lb are the parameters of the ANN. The employed tansig( ) – function

$$\text{tansig}(x) = \frac{2}{1 + \exp(-2x)} - 1, \quad (9)$$

has an output range of [−1, +1]. Therefore the outputs of the ANN need to be normalized. Note that the inputs of the neural networks in our model correspond to the arguments of $f(\ )$ which do not necessarily coincide with the inputs of the system. The process of finding the weights and biases of the ANN is called training. To train, a set of input and output time series that are related by the unknown function f( ), is used to adjust the weights and biases to minimize the error between the calculated value of the output neurons and the given output time series.

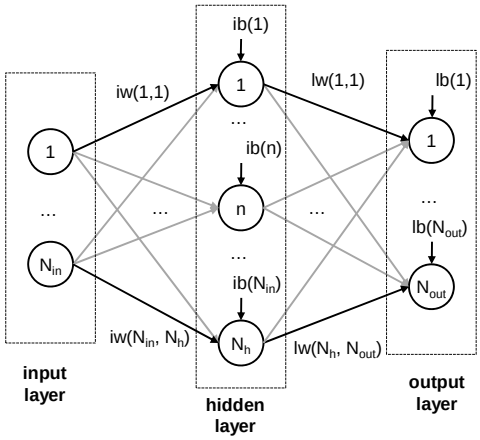

Fig. 2. Schema of a single layer perceptron. The neuron values are calculated according to (7) and (8).

III. Complete Model of the Oscillator

The first step in building the model of the oscillator is to define its order N as well as its number of inputs and outputs. From analogy with the Van der Pol-equation, which was sufficient to capture the dynamics of an oscillator, $N = 2$ is choosen. We consider a single ended oscillator, thus $N_y = 1$. To be able to start up the oscillator by noise, we define one input ($N_u = 1$). The last line of (4) in this particular model is thus given by

$$\dot{x}_2(t) = f\big(x_1(t), x_2(t), u(t)\big), \tag{10}$$

with the output voltage being $v(t) = x_1(t)$ and its derivative $\dot{v}(t) = x_2(t)$. The white gaussian noise for $u(t)$ is generated by the VHDL-AMS code presented in [7]. Because the noise is injected to the LC tank structure, it is converted into $1/f^2$ phase noise. Note that this noise source is virtual, however, its variance σ_1 can be used as a fitting parameter to model the overall effect of all noise sources that contribute to the phase noise in the $1/f^2$ region.

IV. Training the Neural Network

To make the ANN to represent the nonlinear function $f(\,)$, an appropriate set of training data needs to be generated. This is done by simulating the oscillator circuit in the device level simulator ADS. The ANN is trained by the Back Propagation Algorithm [6] using the Matlab Neural Network Toolbox. Note that the training procedure assumes perfect reproduction of the output signal by the ANN by using inputs $x_1(t)$ and $x_2(t)$ that are related to the training goal rather than to the actual output of the ANN. The key to a well trained ANN is a training data set which densely covers the whole input space. Such a data set is easy to generate for independent inputs, by varying them systematically over the desired range of values. However, in the oscillator model the input $x_2(t)$ of the ANN is both the derivative of $x_1(t)$ and the integral of $\dot{x}_2(t)$. Their values cannot be varied separately.

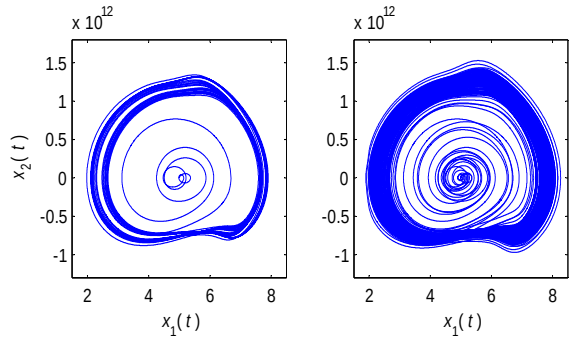

Fig. 3. Input space of training signal for sparse (left) versus dense (right) training data set.

When using noise for the signal $u(t)$ during the training process, only the original trajectory of the system is accurately reproduced by the ANN. This is problematic, because then the region in the state space outside of the limit cycle is not well defined and the gaps between two lines of the limit cycle are too large. The result is a weak model, where $f(\,)$ is only known on the exact trajectory: Small errors could lead the simulator to part from the trajectory and end up in an undefined or unstable state. To avoid this, we inject a signal $u(t)$ of a non negligible amplitude (as opposed to the weak noise signal). By doing so, $f(\,)$ is known to the ANN for a much larger number of arguments (cf. Fig. 3). This yields more robust models.

The model under consideration exhibits a sufficiently small mean square training error, if about ten neurons are used in the hidden layer. Further increase of the number of neurons does not increase the quality of the ANN, since the task of finding the appropriate weights and biases becomes more difficult. The output an accurately trained net is compared to the training signal in Fig. 4.

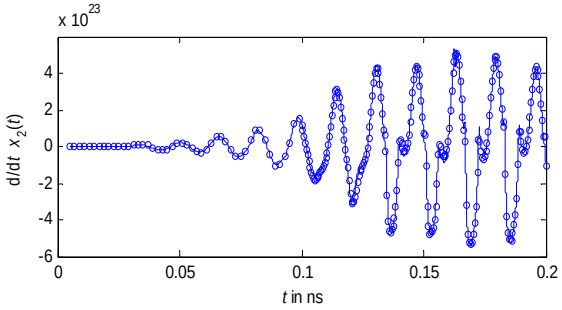

Fig. 4. Solid line: given training data. Marker "o": Output of trained ANN with ten neurons.

V. Implementation in VHDL-AMS

VHDL-AMS permits to enter differential equations on defined quantities (e.g. analog values like voltages or currents) using the `'dot` directive to indicate derivatives. To implement the model, the state space equations (3) and (4) with (10) as last line, as well as equations (7) and (8) for the neural network are entered. To enable the ANN to reflect the whole output range of $f(\,)$, the voltages and their derivatives need to be normalized to $[-1, 1]$. The voltage of the first noise source described in chapter III is assigned to $u(t)$, the one of the second noise source is added to $v(t)$.

Two special properties of an oscillator at millimeter wavelengths need to be taken into account: First, because of the two very different time constants of the model equations, they need to be solved by a numerically stable integration algorithm. The optimum stability is achieved by Gears algorithm of second order. Secondly, because

$\dot{v}(t)$ has a completely different order of magnitude than $v(t)$ for an oscillator at 60 GHz (cf. Fig. 3), two different tolerances have to be used for that quantity and its derivative. Only the VHDL-AMS simulator SMASH (by Dolphin Integration) is able to deal with this problem. It implements the VHDL-AMS directive `tolerance`, which enables the user to specify different tolerances for different equations.

Before simulating the oscillator model it is necessary to verify that the simulators initial value for the DC-part of v(t) is located inside the limit cycle. Outside this limit cycle the model differs from the original and yields additional (invalid) operating points.

VI. Modeling Results

To efficiently convert an ADS device level model to a behavioral model in VHDL-AMS, various software tools were created. The training data is generated by ADS and exported to Matlab, where the ANN is repeatedly trained starting from a large number of different, random initial conditions. The ANN yielding the lowest mean squared error is selected and saved. The VHDL-AMS model reads its weights and biases to parameterize the ANN-equations. In Fig. 5 the output waveform achieved by a VHDL-AMS model of a 60 GHz Colpitts oscillator using ten neurons is compared to the original ADS output. The DC operation point of 5.0 V corresponds exactly to the original. The two signals differ in the transient region, because the order N of the VHDL-AMS model is much smaller than the number of internal states N_x. If the transient is to be simulated more accurately, N needs to be increased. The steady state is reproduced very accurately: The frequency is 61.08 GHz instead of 61.22 GHz in the original circuit, and the voltage range is 2.32 V to 7.71 V instead of 2.30 V to 7.73 V. These results are obtained when using a maximum time step of 10 fs in SMASH. If the maximum time step is set to 100 fs, the amplitude accuracy diminishes, yielding a voltage range of 2.34 V to 7.64 V. The benefit is the decrease of simulation time for a signal of 2 ns from 6.8 s to 1.6 s.

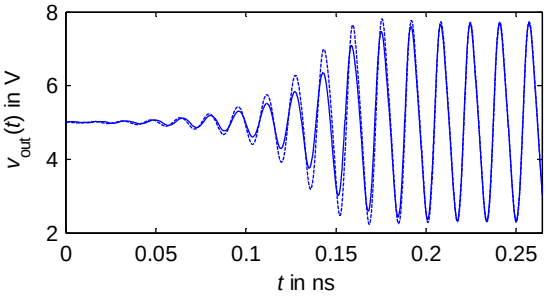

Fig. 5. Solid line: VHDL-AMS response, dashed line: ADS response (original circuit)

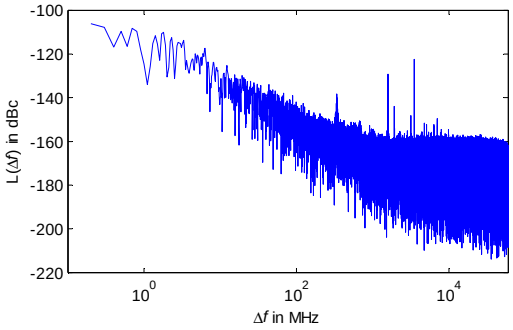

Fig. 6. Phase noise spectrum calculated from the model output $v_{out}(t)$ in steady state

The phase noise generated by the VHDL-AMS model is shown in Fig. 6. One can clearly distinguish between the flat region and the $1/f^2$ region. The two asymptotes of Fig. 6 can be shifted by adjusting the variances σ_1 and σ_2 of the two noise sources employed in the model.

VII. Conclusion

This paper presented a methodology to model a nonlinear oscillator in VHDL-AMS using an ANN, including the capability of simulating phase noise. The output reproduces faithfully the behavior of the original circuit. The presented example models a very basic oscillator, but can be enhanced by also taking into account other properties of the circuit like an external load resistance or the voltage controlled variation of the frequency.

Acknowledgement

This work was supported by the French National Research Agency ANR, under project RadioSoC (No JC05-60832).

References

[1] "IEEE standard VHDL analog and mixed-signal extensions: Std. 1076.1-1999," 1999.

[2] D. Schreurs, "Overview of RF large-signal modelling techniques based on time-domain large-signal measurements," GA 2002.

[3] Y. Fang, M. Yagoub, W. Fang, Q. Zhang: "A new macromodeling approach for nonlinear microwave circuits based on recurrent neural networks," IEEE Trans. Microwave Theory & Techn., Vol. 48, No. 12, pp. 2335–2344, December 2000.

[4] J. Xu, M. Yagoub, et al., "Neural-based dynamic modeling of non linear microwave circuits," IEEE Trans. Microwave Theory & Techn., Vol. 50, No. 12, pp. 2769–2780, December 2002.

[5] J. Wood, D. Root, and N. Tufillaro, "A behavioral modeling approach to nonlinear model-order reduction for RF/microwave ICs and systems," IEEE Trans. Microwave Theory & Techn., Vol. 52, pp. 2274–2284, 2004.

[6] S. Haykin, "Neural Networks - A Comprehensive Foundation", 2nd ed.Upper Saddle River, New Jersey: Prentice-Hall, Inc., 1999.

[7] E. Normark et al., "VHDL-AMS behavioral modeling and simulation of a π/4 DQPSK transceiver system", BMAS 2004, p.119–124.