

A Novel Technique to create Behavioral Models of Differential Oscillators in VHDL-AMS

M. Kraemer¹, D. Dragomirescu^{1,2}, R. Plana^{1,3}

¹ LAAS-CNRS, Université de Toulouse, 7 Avenue de Colonel Roche, 31077 Toulouse, France

Email: {mkraemer, daniela, plana}@laas.fr

² Université de Toulouse, INSA

³ Université de Toulouse, UPS

Abstract — This paper describes an approach to model the transient and steady state behavior of differential microwave oscillators at the system level. Furthermore, phase noise characteristics can be emulated by the resulting models. The mathematical structure consist of a set of algebraic and differential equations that are solved by a VHDL-AMS interpreter. The nonlinearity occurring in the oscillator is described by multi-layer perceptron artificial neural networks (ANNs). In particular, it is shown how to model accurately the two mutually dependent outputs of differential oscillators without having an input signal as (phase) reference. The presented methodology can be employed independently of the used technology and the initial complexity of the oscillators.

1. INTRODUCTION

Behavioral modeling is indispensable, if the simulation of a large System on Chip is desired and a transistor level simulation would be too complex. Especially if the considered circuits are of mixed signal type, the hardware description language VHDL-AMS [1] is perfectly suited for this task, since it contains both analog and digital modeling capabilities. The goal of this kind of model is the order-reduction of an existing transistor level model rather than the model extraction from measurement results.

In [2] and [3] methodologies to model the input-output relationship of nonlinear microwave circuits using ANNs are described. However, this is done in the context of large signal network analysis, where the output of the circuit is related to a well known signal exciting the input. The modeling of noise and an implementation in a description language like VHDL-AMS is not considered there.

Reference [4] describes how the behavior of a single-ended microwave oscillator can be accurately modeled using VHDL-AMS. The presented approach uses a nonlinear differential equation similar to the Van-der-Pol equation, but employs an ANN instead of the polynomial to describe the nonlinearity. This works well for single-ended oscillators. It fails however, if mutually dependent outputs have to be modeled (like in the case of a differential oscillator), because the model doesn't take into account their internal dependency if extended to several outputs.

The approach presented in this paper overcomes the limitations of the previously mentioned work by using a system state that is different from the output and its derivatives, but can be expressed as a linear combination of them. This state is mapped to each one of the two outputs, yielding not only the differential mode but also the common mode behavior of the oscillator.

2. MODEL EQUATIONS

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

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

and an output equation

$$\mathbf{y}(t) = \Psi(\mathbf{x}(t)), \quad (2)$$

together constituting the state space representation of the system [e.g. 5]. Hereby, $\mathbf{u}(t)$ is a vector containing the inputs of the system and $\mathbf{y}(t)$ is a vector containing its outputs at each time instant t . The vector function $\Phi(\cdot)$ in (1) contains any nonlinearity present in the dynamic evolution of the system states. $\Psi(\cdot)$ is a nonlinear function mapping these states to the outputs of the system. The order N of the original system is given by the number of its states, physically represented by the number of independent energy storages.

Building a black box model implies that the internal physical states of a system are either not known or not of interest. Using less system states than physically present reduces the complexity of the model. This is the main interest in behavioral modeling. While in [4] the two system equations were merged to one, here the distinction between the states and outputs of the system, i.e. between equations (1) and (2), is kept. This is due to the fact that in the case of differential oscillators the dominant system states $x_n(t)$ are not equivalent to the outputs or their derivatives.

When considering the simplified equivalent circuit of the differential oscillator in figure 1, two dominating energy storages, constituting the LC tank, are identified. Thus, the state vector $\mathbf{x}(t)$ contains two elements:

$$\mathbf{x}(t) = \begin{bmatrix} v_y(t) - v_x(t) \\ \dot{v}_y(t) - \dot{v}_x(t) \end{bmatrix} = \begin{bmatrix} v_{\text{diff}}(t) \\ \dot{v}_{\text{diff}}(t) \end{bmatrix} \quad (3)$$

Note that it is not sufficient to model only the differential voltage $v_{\text{diff}}(t)$, because the system outputs $v_x(t)$ and $v_y(t)$ contain important information like DC operating point and common mode oscillations that result from the transistors leaving their desired operating region. Thus, it is necessary that equation (2) maps the system state, represented by the vector $\mathbf{x}(t)$, to the output voltages of the oscillator (indicated in figure 1) which are combined to the output vector

$$\mathbf{y}(t) = \begin{bmatrix} v_x(t) \\ v_y(t) \end{bmatrix}. \quad (4)$$

As stated before, an oscillator is not a classical input-output system. In fact, the evolution in time of its outputs depends mainly on the system state. However, to start the oscillation it is necessary to deviate the system from its (unstable) singular point $\dot{\mathbf{x}}(t) = 0$. This can either happen due to mathematical inaccuracy, without defining any implicit input of the

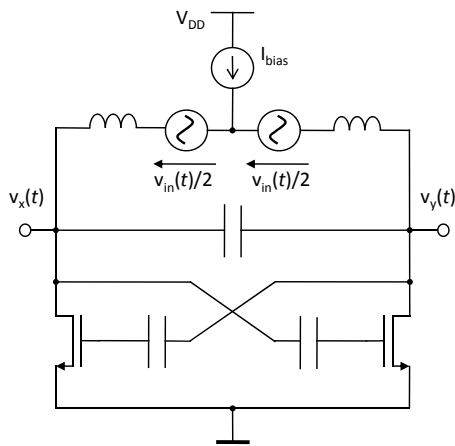


Figure 1: Simplified schematic of the differential Colpitts oscillator used to demonstrate the modeling approach.

system, or more controlled by adding an artificial input. The advantage of the latter is that this input, placed wisely, can be used to introduce and thus emulate phase noise in the $1/f^2$ region [4]. The differential voltage source v_{in} in figure 1 can be used as such an input. If a white Gaussian noise voltage is generated by this source, it is filtered by the LC tank and thus exhibits the characteristic slope of 20dB/decade. To include this voltage source in the proposed model, the input vector $\mathbf{u}(t)$ is defined by

$$\mathbf{u}(t) = [v_{in}(t)]. \quad (5)$$

Having defined the inputs, states and outputs of the model, the next step is to find a means to describe the nonlinear functions $\Phi(\cdot)$ and $\Psi(\cdot)$.

3. ARTIFICIAL NEURAL NETWORKS

Multi-layer perceptron neural networks with a single hidden layer are capable of approximating any nonlinear function arbitrary well, provided the number of neurons (i.e. nodes) of the ANN is large enough [5]. However, for functions exhibiting a strong nonlinearity and locally varying behavior, like $\Phi(\cdot)$ in the case of the considered oscillator, it is more efficient (in terms of nodes necessary to get a certain accuracy) to introduce a second hidden layer.

In figure 2 a perceptron with two hidden layers is sketched. The nonlinear behavior of the nodes is introduced by the *tansig*-function

$$\text{tansig}(x) = \frac{2}{1 + \exp(-2x)} - 1, \quad (6)$$

which exhibits a smooth transition from the lower limit -1 of its function value to the upper limit +1. The outputs of the ANN need to be normalized to their maxima in order not to violate these limits. Using the

tansig-function, the value of the k th neuron of a layer is calculated by

$$n(k) = \text{tansig} \left[\sum_{j=1}^N (n_{in}(j)w(j,k)) + b(k) \right], \quad (7)$$

with $n_{in}(j)$ being the j th neuron of the precedent layer of N neurons, $w(j,k)$ being the weight assigned to the path from this neuron to the current one, and $b(k)$ its bias value. The weights, biases and normalization factors completely characterize the ANN.

The process of finding the weights and biases in order to constitute a desired function is called training. To train the net, a data set that represents arguments and their corresponding function values at different times t is used. The goal is to minimize the error between the response of the ANN and the training time series by adjusting the weights and biases accordingly. The ANNs presented in this paper were trained by different back-propagation algorithms [5] already implemented in the Matlab Neural Networks Toolbox.

4. CREATING $\Phi(\cdot)$ AND $\Psi(\cdot)$

Equation (1) relates the values of $\mathbf{x}(t)$ and $\mathbf{u}(t)$ to $\dot{\mathbf{x}}(t)$ for each t , using the nonlinear vector function $\Phi(\cdot)$. While the derivative of the first state of the system, $\dot{x}_1(t)$, is equivalent to $x_2(t)$, the part of the function to calculate $\dot{x}_2(t) = \dot{v}_{diff}(t)$ is highly nonlinear and thus implemented by an ANN with two hidden layers. The input nodes of this ANN are assigned to $x_1(t)$, $x_2(t)$ and $\mathbf{u}(t)$.

The state $\mathbf{x}(t)$ of the system can be interpreted as the trajectory on which the differential mode oscillation develops. The direction in which this trajectory advances is given by $\dot{\mathbf{x}}(t)$. To assure that the model state develops on the trajectory traced by the original

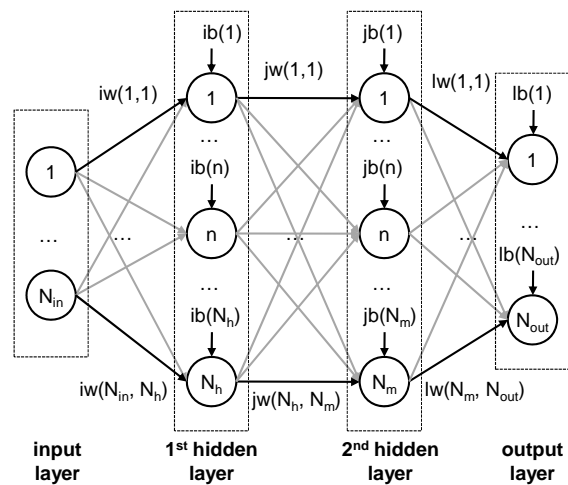


Figure 2: Schema of a perceptron artificial neural network with two hidden layers

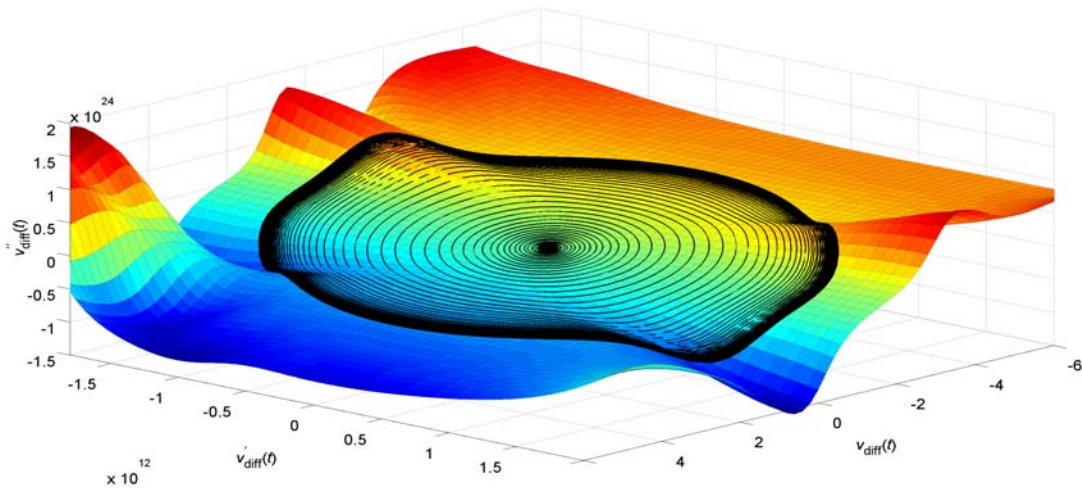


Figure 3: Trajectory from ADS-simulation used for training (black line) versus function represented by the ANN (surface)

system, it is important that the nonlinear function is approximated very well by the ANN at any relevant point $\mathbf{x}(t)$. Any small deviation of $\dot{\mathbf{x}}(t)$ that causes the system to leave this trajectory makes the model fail. That renders the training of this ANN much more important than in the case where an input-output relation dominates the system behavior, and the training accuracy only influences the accuracy of the model, but not its functionality.

An ideal training consists of sweeping all relevant input constellations of the ANN by varying each input independently. However, this is not possible in the present case, because the inputs $x_1(t)$ and $x_2(t)$ are mutually dependent, and the output is the derivative of $x_2(t)$. To nevertheless create a proper training data set, several startups with different signals provided to the v_{in} input are simulated in a transistor level simulator (e.g. ADS, Spice). The signals $v_{in}(t)$, $x_1(t)$, $x_2(t)$, $\dot{x}_1(t) = x_2(t)$ and $\dot{x}_2(t)$ resulting from these simulations are recorded and used to train the ANN.

Since this training data set is the only information that is used in the process of creating the oscillator model, its accuracy is essential. The derivatives used for training have to be computed using a very small time step (about a tenth of the time step used for a normally accurate simulation), because every inaccuracy present here will result in an error inherent in the VHDL-AMS model even if the ANNs perfectly reproduce the training data. A small error in the derivative of $\mathbf{x}(t)$ would accumulate in a run of the final VHDL-AMS model, yielding a substantial error in the transient response and the steady state output voltage.

Used to approximate the state equation of the differential Colpitts oscillator sketched in figure 1, an ANN with 10 neurons in the first hidden layer and 5 neurons in the second hidden layer yields a very low mean square error (MSE) of 2.8E-6. In figure 3 both

the trajectory of the signal used to train the ANN and the function provided by the ANN is displayed. Excellent agreement is observed.

The task of generating the output function $\Psi(\cdot)$ is less critical: Because it isn't used in the differential equation of the oscillator, small inaccuracies won't accumulate. A perceptron with one hidden layer that maps $v_{diff}(t)$ and $\dot{v}_{diff}(t)$ to $v_x(t)$ and $v_y(t)$ is sufficient here. Nevertheless, excellent agreement, quantified by a MSE of 4.8E-6, can be achieved using only three nodes. This shows that the nonlinearity described by this ANN is much weaker than the one in $\Phi(\cdot)$.

5. VHDL-AMS IMPLEMENTATION

The state and output equations (both including ANNs) have to be entered according to the VHDL-AMS syntax when implementing the model. The program code consists of the following parts:

- Initialization: Load weights and biases as well as normalization constants of the two ANNs from Matlab data files.
- Definitions: Define states, inputs and outputs of the system as well as all nodes of the ANNs to constitute continuous-time VHDL-AMS quantities. The `generate` directive helps to facilitate this task.
- Neuron equations: Formulate simultaneous statements (using the `==` operator of VHDL-AMS) that define the neuron values for both ANNs according to equation (7)
- Differential equations: Relate the quantity $v_{diff}(t)$ and their time derivatives by the `'dot'` directive according to equation (1). Normalization factors need to be taken into account.
- Excitation at the v_{in} port: The oscillation can either be started by a short, transient pulse or a Gaussian noise signal (cf. [4]).

After implementing the VHDL-AMS code described above, the generated model has to be compiled and solved. This is done by the VHDL-AMS simulation tool SMASH (by Dolphin Integration). Two important issues have to be addressed in the solution process: First, for an oscillator working at microwave frequencies (or even at mm-waves like 60 GHz in the present example) the voltages have a largely different order of magnitude compared to their derivatives. It is thus necessary to specify their absolute tolerances independently. To the best of the author's knowledge this feature is only available within the SMASH simulation environment.

Secondly, it is important to use a stable and efficient numerical algorithm like Gear's [6]. Otherwise the model requires prohibitively small time steps or doesn't converge at all.

Additionally, it is necessary to limit the search domain for the DC operating point to the area of the state space that is covered by training data. Otherwise it's possible that the ANNs provoke additional, deceptive operating points.

6. VHDL-AMS SIMULATION OUTPUTS

To show the performance of the VHDL-AMS model, the approach presented in this paper is applied to a differential Colpitts oscillator, whose simplified structure is sketched in figure 1. Note that the schematic in this figure doesn't represent the real complexity that is contained in an integrated oscillator. This could contain output buffers and matching circuitry as well as a multitude of parasitic capacitors and resistors extracted from the integrated circuit layout.

From the complex, accurate transistor level ADS model of this oscillator, selected output data is recorded and exported to Matlab. It is subsequently used to train the ANNs employing the Matlab neural network toolbox. The parameters characterizing these ANNs are read by the VHDL-AMS code when being compiled by the SMASH simulation environment. Excited by a tiny pulse at the artificial input v_{in} , the model yields the differential outputs $v_x(t)$ and $v_y(t)$ in figure 4. The difference between VHDL-AMS model and reference data is so small that it can only be clearly seen in the zoomed view.

This shows the capability of the model to reproduce nonlinear transient and steady state in excellent agreement with the low level simulation data. The VHDL-AMS model exhibits an oscillation frequency of $f_{\text{VHDL-AMS}} = 67.47$ GHz versus $f_{\text{ADS}} = 67.52$ GHz obtained by the ADS simulation. The voltage swing is 0.99V to 4.82V for the VHDL-AMS model versus 0.98V to 4.86V for the ADS simulation. The DC operating point at 2.75 V was exactly reproduced. All

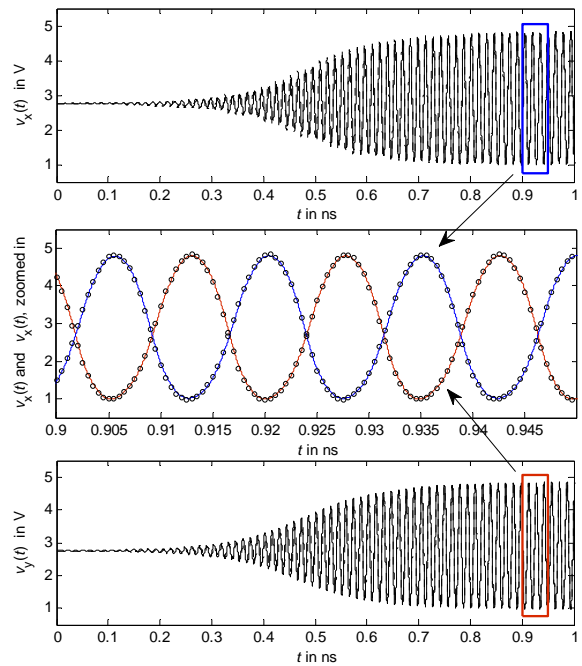


Figure 4: Output voltages $v_x(t)$ and $v_y(t)$ generated by the VHDL-AMS model (solid lines) versus original ADS data (full plot: dotted, zoomed plot: markers).

these results were obtained using Gear's algorithm and a fixed time step of 10fs in both the ADS and the VHDL-AMS simulations.

7. CONCLUSION

An approach for modeling differential oscillators in VHDL-AMS was presented. It faithfully reproduces the behavior of the original circuit in transient and steady state, and incorporates phase noise and common mode of the oscillator. Thus it is ideally suited for simulations that require good accuracy, but cannot afford the computational complexity of circuit level models.

The model is based on the reduced state space representation of the circuit, employing artificial neural networks to reproduce its nonlinear behavior. ANNs using a second hidden layer proved useful in describing the strongly nonlinear behavior of an oscillator exhibiting a nonzero common mode component.

Acknowledgement

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

Thanks also go to Dolphin Integration for providing access to an academic license of the VHDL-AMS simulation environment SMASH.

References

- [1] "IEEE standard VHDL analog and mixed-signal extensions: Std. 1076.1-1999," 1999.
- [2] J. Xu, M. Yagoub, et al., "Neural-based dynamic modeling of non linear microwave circuits," IEEE Trans. Microwave Theory & Techniques, Vol. 50, No. 12, pp. 2769–2780, December 2002.
- [3] J. Wood, D. Root, and N. Tuffiaro, "A behavioral modeling approach to nonlinear model-order reduction for RF/microwave ICs and systems," IEEE Trans. Microwave Theory & Techniques, Vol. 52, pp. 2274–2284, 2004.
- [4] M. Kraemer, D. Dragomirescu, R. Plana, "Nonlinear Behavioral Modeling of Oscillators in VHDL-AMS using Artificial Neural Networks", IEEE Radio Frequency Integrated Circuits Symposium, June 2008, pp. 689 – 692.
- [5] S. Haykin, "Neural Networks - A Comprehensive Foundation", 2nd ed., Upper Saddle River, New Jersey: Prentice-Hall, Inc., 1999.
- [6] C. Gear, "Simultaneous Numerical Solution of Differential-Algebraic Equations", IEEE Transactions on Circuits and Systems, 1971, vol. 18, p. 89-95