A Nonlinear Order-Reducing Behavioral Modeling Approach for Microwave Oscillators

Michael Kraemer, Daniela Dragomirescu, Robert Plana

A Nonlinear Order-Reducing Behavioral Modeling Approach for Microwave Oscillators

Michael Kraemer, *Student Member, IEEE,* Daniela Dragomirescu, *Member, IEEE,* and Robert Plana, *Senior Member, IEEE*

***Abstract*—This paper describes a novel technique to model the transient, steady state and phase noise behavior of microwave oscillators in the hardware description language VHDL-AMS. It can be applied to a large variety of both single-ended and differential voltage-controlled oscillators independently of their architecture. The model is derived from data obtained by a more complex circuit-level model.**

As opposed to input-output models of a microwave two-port, where the output follows more or less the applied input signal, the output of an oscillator depends mainly on its former state. Thus, approaches developed for input-output modeling cannot be applied. The technique proposed in this paper approximates the dynamics of the oscillator by a system of two first-order ordinary differential equations. The oscillator's nonlinear characteristics are reproduced by a multilayer perceptron neural network. In addition to reproducing the oscillator's large-signal waveform, its phase noise characteristic in the $1/f^2$ and flat region is emulated. Finally, a VHDL-AMS implementation of the model is proposed and associated issues are addressed. The suitability of the model for oscillators at millimeter waves is demonstrated by examples working at 60 GHz.

***Index Terms*—nonlinear, behavioral modeling, VHDL-AMS, VCO, multilayer perceptron**

I. Introduction

In the design process of integrated microwave circuits like oscillators, highly sophisticated transistor level models are used. Furthermore, parasitic extraction adds additional capacitances to the schematic that multiply the order of the circuit model. Thus, when it comes to the simulation of more complex circuits like phase locked loops (PLL) or even entire mixed-signal integrated systems, these circuit models need to be replaced by behavioral models with reduced complexity. A hardware description language that represents a suitable tool for this task is VHDL-AMS [1]. It incorporates both digital and analog modeling capabilities. Using VHDL-AMS, it becomes feasible to simulate the behavior of a whole heterogeneous system on chip using the same modeling language, while taking into account interactions between the digital and analog part.

There exist a multitude of different approaches to model nonlinear microwave circuits at system level, introduced e.g. in [2], [3] or [4]. Depending on the approach, either time or frequency domain descriptions are utilized. However, the behavior of digital circuits is almost exclusively described in the time domain in VHDL, so to well interface to the digital circuit parts, VHDL-AMS also uses a time domain description. The oscillator model thus needs to be a time domain model. For this reason frequency domain models are not further taken into consideration here.

The behavioral time-domain models of oscillators can be divided into two principal categories: Either, the output voltage is expressed by a known, usually sinusoidal function that is evaluated at each time step (e.g. [5], [6]). However, in this case neither the transients and dynamics of the oscillator nor its nonlinear characteristic are correctly reproduced. Thus, the use of such models is limited to very simplified considerations. Or, secondly, the oscillator is described by an equation solved during simulation. The prototype for a model of this category is the Van der Pol-equation: Here, a second order nonlinear ordinary differential equation (ODE) describes the output voltage of a triode oscillator, using a polynomial to approximate its nonlinear characteristic [7]. An extension of this polynomial in order to model solid state oscillators is not trivial, because in addition to the function describing the device current, nonlinear capacitances need to be taken into account. Furthermore, neither the operating point and common mode behavior of differential oscillators nor its phase noise characteristics are incorporated in a model based on the Van der Pol-equation.

In the context of large signal network analysis, different nonlinear modeling techniques of the second type are investigated in literature. They approximate the nonlinearity of the circuit by radial basis functions, multivariate polynomials, or artificial neural networks, the latter showing the most accurate results [8]. The dynamics are either contained by using delays to maintain the notion of time [9], or in form of an ODE that is solved by the simulator [10].

However, all these methodologies yield *input-output* models of microwave multiports and thus cannot be directly applied to oscillators, which only exhibit *one* microwave output. Nevertheless, some of the basic concepts used to develop these methodologies are adopted in the presented oscillator model.

The approach proposed in the present paper describes the oscillator's behavior in state space, i.e. by a system of first-order ODEs. The nonlinear relationship that is contained in these equations is represented by an artificial neural network (ANN), similar to [8]. It is thus a non-physical black box model that is capable of incorporating the strong nonlinearities

Manuscript received August 1st, 2008; revised 22.12.2008. Work supported by the French National Research Agency ANR, under project RadioSoC (No JC05-60832)

M. Kraemer, D. Dragomirescu and R. Plana are with CNRS; LAAS; 7 avenue du colonel Roche, F-31077 Toulouse, France and Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France. e-mail: mkraemer@laas.fr, daniela@laas.fr; plana@laas.fr

DOI: 12365467983

occurring in integrated microwave oscillators as well as its dynamics.

The novel modeling technique introduced in this paper is very general: By reducing the order of the system to the minimum of two that is necessary to describe an oscillation, the evolution of the system state can be described in a two-dimensional state space. Depending on the architecture of the oscillator, the system state is then mapped either to a single-ended or differential output. This mapping also adds bias points and common mode behavior, which are faithfully reproduced by the model. The use of multilayer perceptron ANNs with two hidden layers to reflect the nonlinear characteristic of the oscillator allows for an accurate and easy applicable modeling of rather complex oscillators. This provides the capability to model oscillators where the system response is modified by a control voltage (model of a voltage controlled oscillator, VCO).

To include phase noise in the model and to start the oscillation in a well defined manner, a random signal is injected to an artificial noise port of the oscillator.

In order to describe this novel approach in sufficient detail, the paper is structured as follows: Section II introduces the theory of nonlinear oscillations, multilayer perceptron neural networks and the state space representation of nonlinear systems. Section III details the different aspects of the novel modeling technique, notably the structure of the model, the way in which phase noise is injected and the generation of data that is well suited to make the neural network represent the nonlinear behavior of the oscillator. Furthermore, the modeling flow is presented and some source code examples are given. Section VI contains results that show the capabilities and performance of the model. A conclusion is drawn in section VII.

II. Theoretical Background

A. System Description in State Space

A nonlinear, time-invariant, deterministic system can be mathematically described by a state equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\phi}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \tag{1}$$

that characterizes the dynamics of the system state $\boldsymbol{x}(t)$ and an output equation

$$\boldsymbol{y}(t) = \boldsymbol{\psi}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \tag{2}$$

that maps the state vector $\boldsymbol{x}(t)$ to the system's output vector $\boldsymbol{y}(t)$ of size N_y. The vector $\boldsymbol{u}(t)$ contains the N_u scalar inputs of the system. The N_x-dimensional space on which $\boldsymbol{x}(t)$ is defined is called the system's *state space*. The vector function $\boldsymbol{\phi}()$ describes the nonlinear relationship between the system state and its derivative. Because $\dot{\boldsymbol{x}}(t)$ describes the dynamical evolution of the states in direction and magnitude, it is also called the *velocity vector*. The vector function $\boldsymbol{\psi}()$ contains the relationship between system state and system output.

If a system's state is known at one point in time, all its future states and outputs are defined by the state space equation and the future inputs of the system. While in an input-output system (like a mixer or an amplifier) the influence of the inputs on the system state is dominant, the evolution of an oscillator's state depends primarily on its former states. An input is only needed to start the oscillation by deviating the system from its singular point at $\dot{\boldsymbol{x}}(t) = \boldsymbol{0}$.

The state space representation lends itself to the use in the oscillator model, because its dynamics are described as a system of mutually coupled ODEs, and thus it is straight-forward to implement in VHDL-AMS. Furthermore, it provides an intuitive perspective on how to correctly model an oscillation and allows the visualization of the system behavior by plotting its trajectories in state space. These trajectories are an ideal aid when training the neural networks used to represent the circuit's nonlinearities (cf. section IV-A).

B. Nonlinear Oscillators

To describe the behavior of an electrical oscillator, nonlinear autonomous ODEs can be employed. To illustrate the behavior of the oscillator, it is instructive to consider its trajectory in the phase plane, where the oscillator voltage is plotted on the abscissa and its derivative on the ordinate. The phase plane corresponds to the two-dimensional state space of an oscillator. The evolution of the states is represented by the tangent vector on the trajectory and thus equivalent to the velocity vector $\dot{\boldsymbol{x}}(\boldsymbol{t})$. Due to the autonomous nature of the considered ODEs, the trajectories are independent of time.

The phase space trajectories of electrical oscillations exhibit two characteristic equilibrium states: The first one is an unstable singular point at $\boldsymbol{x}_{\mathbf{start}} = \boldsymbol{x}(t_0)$, where the velocity vector $\dot{\boldsymbol{x}}(t_0)$ is $\boldsymbol{0}$. This point corresponds to the bias point of the oscillator, with $x_{\mathrm{start},1} = v_{\mathrm{bias}}$. Due to the instability of this point, any small deviation will lead to the start of the oscillation. The trajectory of the start-up has a spiral form, where the mean distance of $\boldsymbol{x}(t)$ from the singular point is increasing with time (cf. Fig. 1).

The second equilibrium state is an attractive limit cycle, resulting from the limitation of the oscillation amplitude. This is a periodic orbit which all trajectories approach for $t \to \infty$. The basin of attraction, i.e. the area in state space that leads to oscillations ending on the limit cycle, contains the whole inside of the limit cycle, and at least the vicinity of the outside of the limit cycle that can be reached due to external influence, like noise. Otherwise, small deviations from the limit cycle that lead to an increased amplitude of oscillation would yield the system state to leave the limit cycle and be attracted by an other equilibrium state. While this behavior is not critical in real systems, it can occur in improper models. Thus it is important to pay attention to this effect (cf. section IV-D).

Note that the above explication applies only in the case of soft startup conditions. For further details on the theory of nonlinear microwave oscillators, refer to [11],[12].

C. The Van der Pol-Oscillator

To illustrate the theory of nonlinear electrical oscillations, the equation proposed by Van der Pol [7] to describe triode oscillators is used. The classical formulation is

$$\ddot{v}(t) = \alpha\left(1 - v(t)^2\right)\dot{v}(t) - \omega^2 v(t). \tag{3}$$

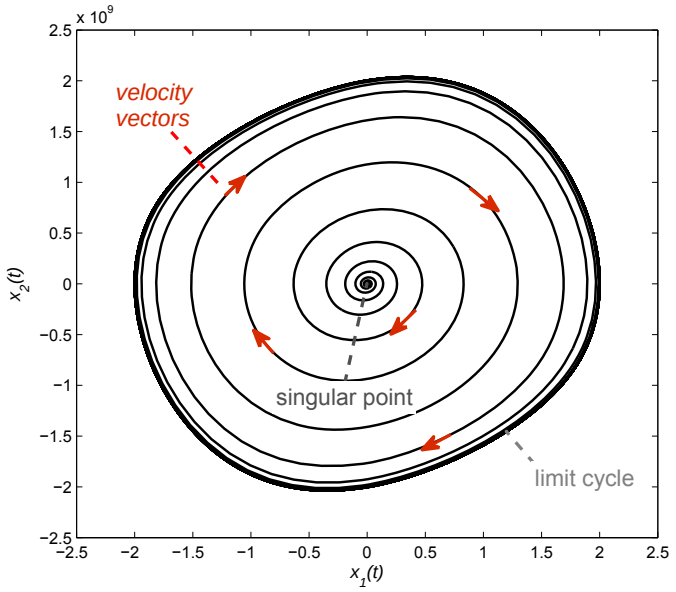


Fig. 1. Plot of the trajectory of a Van der Pol-oscillator, simulated in VHDL-AMS. Singular point $\dot{\boldsymbol{x}}(t) = \mathbf{0}$ for $\boldsymbol{x}(t) = \mathbf{0}$.

By defining the state vector of the Van der Pol-oscillator as

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} v(t) \\ \dot{v}(t) \end{bmatrix}, \tag{4}$$

equation (3) can be rearranged to yield the velocity vector

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ \alpha\left(1 - x_1(t)^2\right) x_2(t) - \omega^2 x_1(t) \end{bmatrix}. \tag{5}$$

Because the system output is equivalent to the first state variable $x_1(t)$, the output equation reduces to

$$[y_1(t)] = [x_1(t)]. \tag{6}$$

From (3)-(6) it is clear that the relation between the system's states, inputs and outputs is completely described by the function $f(t) = \alpha\left(1 - x_1(t)^2\right) x_2(t) - \omega^2 x_1(t)$ in case of the triode oscillator.

To illustrate the two equilibrium states of the Van der Pol-oscillator, the trajectory of one solution (calculated by a simple VHDL-AMS model) is given in figure 1.

D. Model-Order Reduction

In the case of the Van der Pol-oscillator, the differential equation is based on a polynomial approximation of the triode's anode current. This association of the circuit's complete nonlinearity to a single component is not feasible for an integrated microwave circuit, where the influence of a multitude of components (i.e. nonlinear capacitances, etc.) create the nonlinearity.

The complete state space representation of an integrated oscillator can be obtained from its schematic. To represent the circuit's behavior at microwaves correctly, this schematic has to include the parasitic elements extracted from layout. The order N of such a circuit is given by the number of its independent LC energy storages. The model of a complex oscillator taking into account all its parasitics therefore exhibits a large order ($N >> 100$). When using this kind of state space model, all currents and voltages inside the circuit are known, which is essential during circuit design.

However, the internal states of the system are not important for system level simulations, as long as their influence on the output is taken into account by the simplified model. Thus, even when maintaining the same accuracy at the system outputs, a great order reduction is possible by removing the relationship between system states and energy storage elements.

The discussion of the Van der Pol-equation in section II-C shows that the trajectory of an oscillation can be described in a two-dimensional state space. This can be illustrated by the fact that the oscillation physically consists of the periodic exchange of energy between two dominant energy storages. This reasoning leads to the conclusion that the system order of an oscillator can in general be reduced to two. Thus, an embedding approach to find the intrinsic system order as described in [8] is usually not necessary.

Note, that the modeling technique described in this paper can easily be extended to yield higher order models, if for a particular oscillator it turns out that an order reduction to three is more adequate. However, for a properly designed oscillator a reduction to two should be possible.

As the second order dynamics of the oscillator are included in the output and its derivative, it seems sensible to use them as system states as in the case of the Van der Pol-equation. However, when dealing with two related outputs as for differential oscillators, the system state needs to be defined in a way that it can be mapped to the two outputs equally well. If each output and its first derivative were defined as individual system states, the modeling of the interaction between the outputs would be hardly possible.

To avoid this problem, the system states of the behavioral model are based on the differential mode of the oscillator: The differential mode voltage $v_{\mathrm{y}}(t) - v_{\mathrm{x}}(t)$ and its derivative are selected to represent the system state according to

$$\begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} v_{\mathrm{y}}(t) - v_{\mathrm{x}}(t) \\ \dot{v}_{\mathrm{y}}(t) - \dot{v}_{\mathrm{x}}(t) \end{bmatrix}. \tag{7}$$

The advantage of this choice is the observability of the system's state from the output port of the oscillator. Furthermore, this state is related to both of the outputs in the same way. The output voltages $v_{\mathrm{y}}(t)$ and $v_{\mathrm{x}}(t)$ are illustrated in figure 2 by the VCO taken as example throughout this paper.

Mathematically, the order reduction can be seen as a projection of the oscillator's trajectory from the initial, N dimensional state space to the two-dimension state space of the behavioral model. By this projection, the relationship between system states and circuit components is removed and information about the initial states of the oscillator is lost. Plotting the two newly defined states of the oscillator helps to verify that the new state space still contains a trajectory correctly describing the oscillation. If the trajectory is not intersecting with itself, and still shows the typical form of figure 1, the order reduction is valid.

The dynamics of the system state $\boldsymbol{x}(t)$ is then given by the

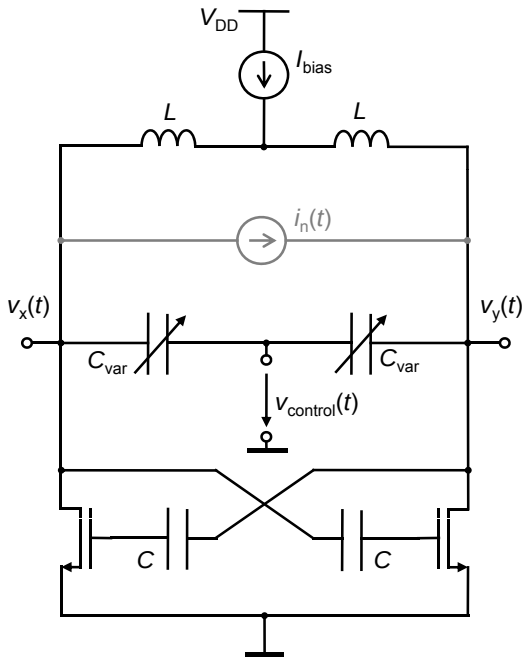


Fig. 2. Simplified schematic of a typical integrated state-of-the-art VCO

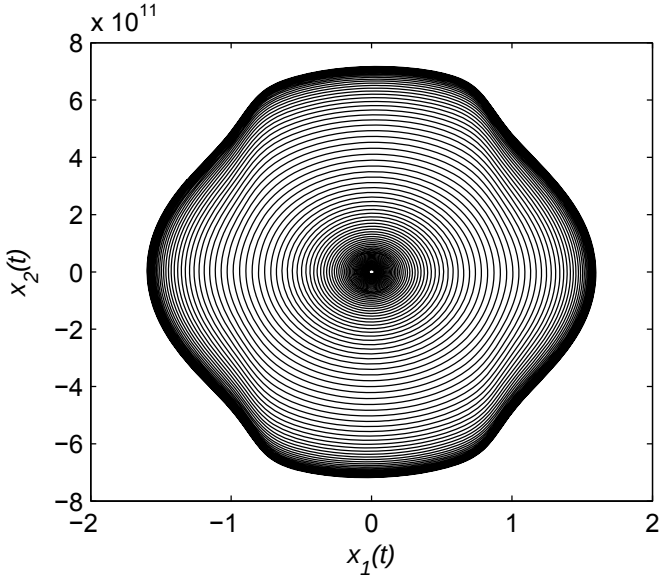


Fig. 3. Trajectory of a 60 GHz differential Colpitts oscillator in the two-dimensional state space as defined in (7)

velocity vector

$$\begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ f\left(x_1(t), x_2(t), \boldsymbol{u}(t)\right) \end{bmatrix} \tag{8}$$

that contains a scalar function $f()$. This function describes the nonlinearity of the oscillator and takes as arguments both the system states and the input vector $\boldsymbol{u}(t)$ that contains the oscillator's inputs like the noise current $i_{\mathrm{n}}(t)$ and the control voltage $v_{\mathrm{control}}(t)$ (cf. section III-A).

The output equation for this reduced order oscillator maps the two states to the oscillator's outputs, while taking into account the parameters and inputs from $\boldsymbol{u(t)}$ according to

$$\boldsymbol{y}(t) = \boldsymbol{\psi}\left(x_1(t), x_2(t), \boldsymbol{u}(t)\right). \tag{9}$$

The system's output vector contains the nodes of the oscillator observable at the output of the behavioral model. In the case of a differential VCO like the one of figure 2, the elements of $\boldsymbol{y}(t)$ are the voltages at the x and y output of the oscillator according to

$$\boldsymbol{y}(t) = \begin{bmatrix} v_{\mathrm{x}}(t) \\ v_{\mathrm{y}}(t) \end{bmatrix}. \tag{10}$$

For the single ended case, this vector reduces to a scalar like for the Van der Pol-oscillator.

An example for a trajectory obtained from simulations using the microwave circuit simulator ADS[1] is given in figure 3. The oscillator is a differential Colpitts VCO having the simplified schematic given in figure 2. The reduced states of the oscillator are calculated according to (7) from the ADS output voltages. The higher complexity of this oscillator is reflected by the form of its limit cycle: It shows a more irregular shape compared to the limit cycle of the Van der Pol oscillator displayed in figure 1 (which is due to the fact that the drain-source saturation voltage of the transistors cannot fall below a certain value). Thus, the trajectory exhibits a dent at each side of the rectangle.

Even the behavior of highly nonlinear oscillators can be described in a two-dimensional state space, however, the form of its trajectory becomes more difficult to approximate and depends heavily on parameters like the control voltage. For this task, the simple second-order polynomial of the Van der Pol-equation is insufficient. In the proposed model it is replaced by a function that contains the relevant nonlinearities of the employed transistors and varactors, including effects like the transition from one operating region to another or nonlinear capacitances.

E. Artificial Neural Networks

Because of the ease of finding the parameters and the resulting accuracy, the model proposed in this paper uses artificial neural networks (ANNs) to describe the oscillator's nonlinear behavior, represented by the functions $f()$ in (8) and $\boldsymbol{\psi}()$ in (9). This section introduces them briefly.

1) Multilayer Perceptrons: An ANN is a structure that resembles the human brain in containing neurons (i.e. nodes) interconnected by synapses (i.e. connections) in order to build a complex structure that is capable of reproducing sophisticated relationships.

A class of artificial neural networks that can be employed to perform a nonlinear input-output mapping of general nature is a multilayer perceptron (MLP) [13]. It can thus approximate the nonlinear vector functions $\boldsymbol{\phi}()$ (reduced to the scalar function $f()$ by (8)) and $\boldsymbol{\psi}()$ of equations (1) and (2).

The MLP's input consists of source nodes that correspond to the arguments of the function $f()$ it approximates. One or more hidden layers create the computational power of the network. Furthermore, one or more output nodes constitute the

[1]ADS 2005A, copyright 2005 by Agilent Technologies

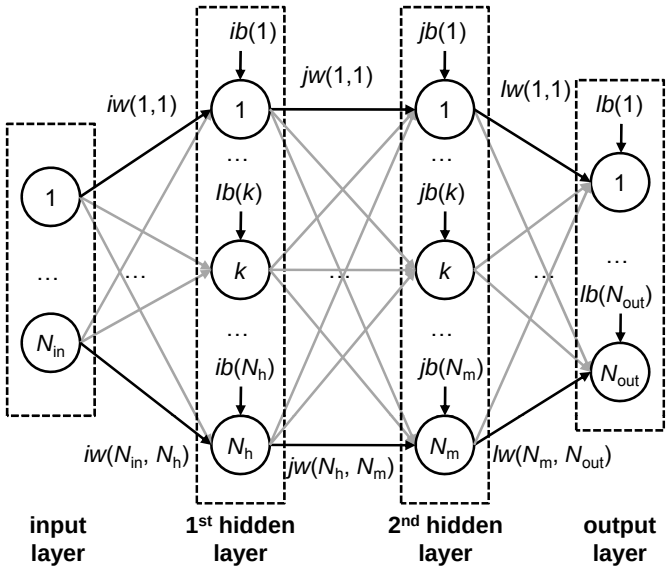


Fig. 4. Schematic of a multi-layer perceptron with two hidden layers.

output layer and correspond to the function value(s) of $f()$. The three distinctive properties of an MLP are [13]:

- It contains a nonlinear activation function to calculate the neuron values. This function introduces the nonlinearity of the MLP and needs to be smooth in order to create a network that can be trained by the error back-propagation algorithm (cf. section II-F). A common choice is the hyperbolic tangent $\tanh()$. It exhibits a smooth transition from -1 to $+1$. As a consequence, the outputs of the MLP need to be normalized to this range. Furthermore, it is convenient to also normalize the input.
- It possesses one or more hidden layers, that are not directly accessible from the outside.
- It contains a high degree of connectivity. Each node is connected to every node in both the preceding and the following layer by weighted synapses.

The structure of a multilayer perceptron network with two hidden layers is sketched in Fig. 4. The neuron values within this structure are calculated by

$$n(k) = \tanh\left[\sum_{j=1}^{M}\left(n_{\mathrm{in}}(j)w(j,k)\right) + b(k)\right] \qquad (11)$$

with $n_{\mathrm{in}}(j)$ being the jth neuron of the precedent layer containing M neurons, $w(j,k)$ being the weight assigned to the path from this neuron to the current one, and $b(k)$ its bias value. Knowing its weights and biases and as well as, if applicable, the normalization factors completely characterizes the MLP.

2) Neural Networks to Approximate Nonlinear Functions: To approximate an arbitrary continuous function it is sufficient to use a MLP with only one hidden layer, provided its number of neurons is sufficiently large. This is the essence of the universal approximation theorem for a nonlinear input-output mapping [14] applied to MLPs. However, it is only an existence theorem and does not guarantee that a solution with one hidden layer is the most practical or even optimum one. In fact, when using only one hidden layer the neurons of the MLP tend to interact with each other globally. Thus, when approximating functions like the one present in an oscillator, the improvement of one point in the approximation will typically lead to the worsening of another one [13].
To avoid these kind of problems, two hidden layers are advantageous. Here, the first hidden layer typically reproduces the local behavior, while the second hidden layer contains global features of the function to be approximated. Thus, curve-fitting becomes more manageable.
That is why MLPs with two hidden layers are recommended to approximate the function $\phi()$ and $f()$, respectively. Only for the weakly nonlinear function $\psi()$ that maps the system state to the system output, MLPs with a single hidden layer are advantageous to avoid increased complexity.
The necessary number of neurons N_h in the first and N_m in the second hidden layer depend much on the pecularities of the oscillator to represent and are thus determined empirically: In the training process, these numbers are decreased iteratively until the maximum desired mean squared error (MSE) is exceeded. This is necessary because the complexity of the function to approximate is not easily quantified and converted to the number of required neurons.

F. Training by Error Back-Propagation

The training of the neural network may be viewed as a curve fitting problem [13]: During this process, the weights and biases of the ANN are adjusted in order to enable the ANN to perform a desired input-output mapping (e.g. represented by the function $f()$ from (8)). A training data set containing input-output pairs is presented to the ANN for this purpose. In addition to approximating the mapping for each input-output pair, the ANN will also generalize the training data and thus be able to map inputs never presented in the training process to their appropriate outputs.
A popular method to train an MLP is error back-propagation. The general algorithm works as follows: First, the weights ($iw(k)$, $jw(k)$ and $lw(k)$) and biases ($ib(k)$, $jb(k)$ and $lb(k)$) are initialized by random values. Next, the output of the MLP is computed based on the input part of the training data. Then, the error between the output from the training data set and the calculated output is determined.
Next, it needs to be determined to what degree this error can be assigned to which weight or bias-values. To be able to do this, local gradients are introduced that contain the influence of each precedent node to the error at the actual node. Because the error is known directly only at the output layer, it has to propagate backwards into the network to determine the error at the neurons in the hidden layers. After the influence of each weight and bias to the error is determined, they are adjusted accordingly. The error back-propagation process is repeated until a given criteria (e.g. MSE below a certain bound, gradient below a certain bound, maximum iteration count, etc.) is reached. Further details on this method are given in [13].
Note that the design of the training data set is an essential part

of the training process. The MLP can only approximate and generalize a behavior it is trained to before. In section IV this issue is discussed with respect to the oscillator model.
While the calculation of the output values of the MLP as a function of the inputs and net parameters, sometimes referred to as forward propagation, constitutes an integral part of the VHDL-AMS model, the training procedure has only to be done once to build the oscillator model. The convenient tools of the Matlab Neural Networks Toolbox[2] have been employed to train the MLP using the Levenberg-Marquardt back-propagation algorithm.

G. Drawbacks

The neural network architecture used to represent the non-linearities works perfectly fine with respect to accuracy of approximation and manageability. Nevertheless, it exhibits two weaknesses: The first one originates from the fact that the parameters are determined by a training process: To get a good approximation by the neural network, several training runs starting from random initial values have to be compared. Based on this, the network having the minimum MSE is picked. Thus, the optimal solution is not found in a deterministic way, and though the MSE of this solution is small, one can't be sure where this error will occur and how well the network will generalize behavior it is trained on. This is not the case for a physical model.
Second, the solution of a system of equation that contains perceptron neural networks with hyperbolic tangent basis functions is more computationally intensive then for example an approximation that contains polynomials or radial basis functions. So for less stringent accuracy requirements, an approximation using this kind of functions may be the better choice.

III. The Novel Modeling-Approach

Based on the theoretical background presented in the previous section, a behavioral modeling methodology is developed. It is applicable to a large variety of LC-oscillators. To illustrate it by a representative example, the circuit sketched in figure 2 is used throughout this paper. It exhibits all important characteristics of a state-of-the-art VCO: It is fully differential, with outputs $v_\mathrm{x}(t)$ and $v_\mathrm{y}(t)$. The tank is comprised of symmetrical inductors L and varactors C_var, whose capacitance values are controlled by $v_\mathrm{control}(t)$. A cross-coupled Colpitts architecture is used, recognizable by capacitive voltage dividers comprising the capacitors C and the gate source capacitances C_GS of the MOSFETs. The VCO is designed to exhibit an oscillation frequency around 60 GHz.
The differential current source $i_\mathrm{n}(t)$ is not present in the original schematic. It is added as a means to inject a small, noise-like signal into the tank. This signal serves several purposes: It starts the oscillation in a more controlled way then numerical inaccuracy would, it can be used to generate more robust training data by varying the trajectory of the oscillator (cf. section IV), and it creates a port at which to inject noise

[2]Matlab R2007a, copyright 1984-2007 by The MathWorks

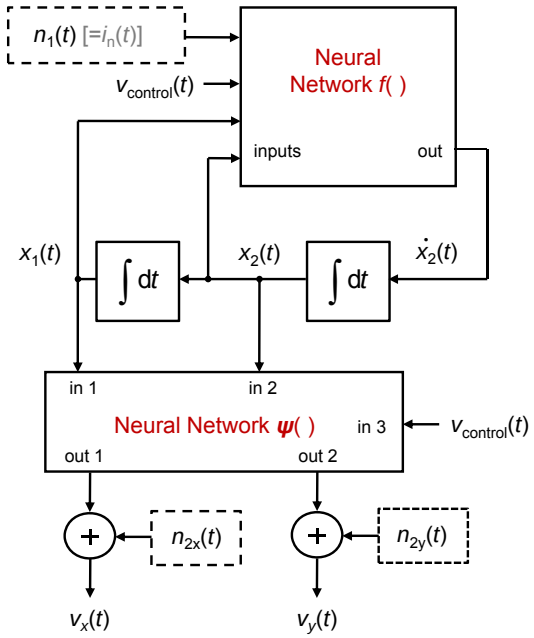


Fig. 5. Block diagram of the oscillator model for the case of differential outputs ($v_\mathrm{x}(t), v_\mathrm{y}(t)$) and one control input ($v_\mathrm{control}(t)$). Blocks with dashed margin represent white Gaussian noise generators.

to the tank in order to generate phase noise in the $1/f^2$ region (cf. section III-B).
Note that a complete schematic of this circuit that is used for low level simulation (e.g. in ADS or SPICE) would include a more complete biasing network, possible output buffers, inductors and/or transmission lines for matching and all the parasitics being extracted from layout. The novel model is created to exhibit the behavior of this entire circuit.

A. Model Structure

The structure of the proposed model is presented in figure 5. The diagram given there is adapted to differential VCOs like the one of figure 2. The state equation (i.e. (8)) is represented by the upper part of the diagram: The output of the ANN is fed back to its inputs using two integrator blocks that embody the derivatives of the differential equation. The neural network employed in this part incorporates the strong nonlinearity, which is mathematically contained in the function $f()$. In addition to the two state variables, the ANN has two other inputs, represented by the input vector

$$\boldsymbol{u}(t) = \begin{bmatrix} v_\mathrm{control}(t) \\ i_\mathrm{n}(t) \end{bmatrix}. \tag{12}$$

The input $v_\mathrm{control}(t)$ is the control voltage applied to change the capacitance of the varactors. The model will be capable of representing the influence of this voltage not only on the oscillation frequency, but also on any other properties of interest (e.g. the tank amplitude or the startup time). It is the only external input in this example, but similar inputs are possible to represent model parameters like the bias currents of the circuit. However, the number of inputs should be limited to those present in the fabricated circuit. Otherwise the reduction

of computation time with respect to a circuit level model is low, which reduces the interest of employing the behavioral model.

Input $n_1(t)$ corresponds to the artificial current source $i_\mathrm{n}(t)$ inserted into the schematic. In the VHDL-AMS model this input of the ANN is connected to a random number generator that creates white Gaussian noise of standard deviation σ_1.

The nonlinear function $\boldsymbol{\psi}()$ used for the output mapping of (8) is realized by the second neural network. The block diagram for this part does not contain feedback: The states $x_1(t)$ and $x_2(t)$ are simply mapped to the output voltages, which results in less stringent accuracy constraints for the ANN employed. The third argument of $\boldsymbol{\psi}()$ represents any influence of the control voltage on this projection. It is equivalent to the input of $f()$ of the same name.

The blocks with dashed outlines in figure 5 contain the noise signals $n_{2\mathrm{x}}(t)$ and $n_{2\mathrm{y}}(t)$. They add white Gaussian noise of standard deviation $\sigma_{2\mathrm{x}} = \sigma_{2\mathrm{y}} = \sigma_2$ to the differential output voltages in order to emulate the noise floor (cf. section III-B). The central role of the states $x_1(t)$ and $x_2(t)$ becomes obvious from this diagram: They are connected to both of the neural networks and the integrator blocks.

Note that in single ended oscillators the above discussed architecture simplifies considerably: Only one output is present, which usually is identical or closely related to the tank voltage. The state variables can be directly assigned to the output and its derivatives as in the case of the Van der Pol-oscillator rendering the second neural network obsolete.

B. Phase Noise Emulation

Independently of the physical explanation that is employed, most theories about phase noise in oscillators assume that the phase noise spectrum can be divided into three regions as introduced by Leeson [15]. More precisely, a flat, $1/f^2$ and $1/f^3$ region are distinguished, corresponding to the asymptotic slope of their characteristic in the double logarithmic phase noise spectrum.

The present behavioral oscillator model is designed to emulate phase noise in the flat and $1/f^2$ region. At this stage, the model does not take into account the contributions creating the $1/f^3$ region, which would translate into increased complexity.

For sake of simplicity, physical processes are not considered when building this part of the model. Rather, the original, more complex circuit model, or measurements of a realized oscillator (if available) are used do determine the actual phase noise characteristics. (In the former case depending on the phase noise model used in the circuit simulation, more or less physical noise mechanisms are taken into account).

From the phase noise spectrum obtained this way the different regions are identified, and the positions of the asymptotes characterizing them are extracted. Based on this knowledge, the phase noise part of the behavioral model is parametrized.

To inject noise that is transposed to phase noise in the $1/f^2$ region, a current source i_n is inserted in parallel to the tank of the oscillator, as illustrated in figure 2. If this source has the mean-square spectral density $\overline{i_\mathrm{n}^2}/\Delta f$ and the tank impedance is $Z(\omega)$, this current source will yield a voltage v_n with mean-square spectral density

$$\frac{\overline{v_\mathrm{n}^2}}{\Delta f} = \frac{\overline{i_\mathrm{n}^2}}{\Delta f}\left|Z(\omega)\right|^2 \tag{13}$$

that is superimposed on the tank voltage v_tank.

Assuming that the tank impedance consists of the total tank inductance $2L$ in parallel with the capacitance $C_\mathrm{tot}/2$ (C_tot contains the two varactors in series as well as all other parasitic capacitances), and any losses are exactly compensated by the active part of the circuit, the resulting tank impedance is given by

$$Z(\omega) = \frac{j\omega 2L}{1-\omega^2 LC} = \frac{j\omega 2L}{1-\left(\frac{\omega}{\omega_0}\right)^2} \tag{14}$$

with $\omega_0 = 1/\sqrt{LC}$ being the resonance frequency. When considering phase noise, the frequencies of interest are close to the carrier, so $\omega = \omega_0 + \Delta\omega$, where $\Delta\omega$ represents the small deviation from the carrier frequency. Thus, $Z(\omega)$ may be approximated by

$$Z(\omega) \approx \frac{j\omega_0^2 L}{\Delta\omega}, \tag{15}$$

neglecting all small and higher order components.

As a result, the mean-square spectral density of the tank voltage due to the injected current is

$$\frac{\overline{v_\mathrm{n}^2}}{\Delta f} = \frac{\overline{i_\mathrm{n}^2}}{\Delta f}\left(\frac{\omega_0^2 L}{\Delta\omega}\right)^2. \tag{16}$$

Due to the equipartition theorem of thermodynamics, half of this power density appears as phase noise. Because $\overline{v_\mathrm{n}^2}/\Delta f$ is proportional to $1/\Delta\omega^2$, the spectrum generated by the considered signal can be used to emulate the $1/f^2$ region. The initially flat noise current density is shaped by the filtering due to the LC-tank [16].

To generate the current density $\overline{i_\mathrm{n}^2}/\Delta f$, a random number generator is used. It generates a normal distributed random variable with zero mean and standard deviation $\sigma_1^2 = \overline{i_\mathrm{n}^2}$. To correctly fix the position of the asymptote for the phase noise in this region, it is thus sufficient to fix the standard deviation σ_1.

A VHDL-AMS implementation of a random number generator for normal distributions can be found e.g. in [17]. It makes use of the `UNIFORM` - function provided by VHDL-AMS that can generate only uniformly distributed random variables, and converts them to normally distributed ones.

To add a noise floor to the oscillator output, the same type of random number generator with standard deviations $\sigma_{2\mathrm{x}}$ and $\sigma_{2\mathrm{y}}$, one for each of the differential outputs, is used. Because the shape of the noise spectrum is already correct for them, no special injection is necessary. Thus, the noise can simply be added to the outputs of the oscillator as indicated in figure 5. The equipartition theorem holds here as well, which means that half of the added noise power is actually phase noise.

C. Modeling Flow

The modeling flow presented in this section assumes that the structure of the particular model has already been established, i.e. that the inputs, outputs and states of the reduced

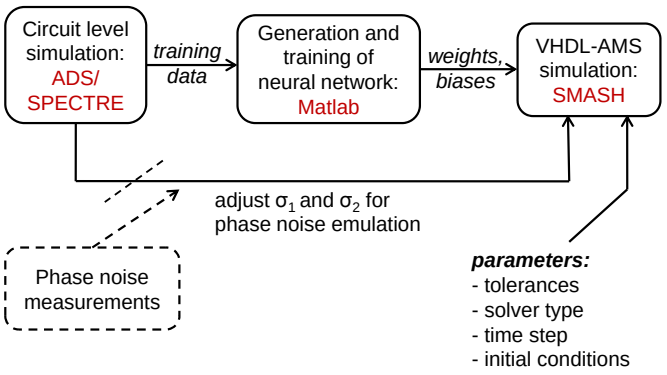


Fig. 6. Modeling flow: from circuit level simulation to behavioral model

order model are defined and the number and kinds of neural networks are fixed. To create a model according to such a structure, several steps are necessary, as illustrated in figure 6. The model is based on an accurate, but rather complex circuit level model. This model is used to generate a dataset able to characterize the oscillator in all relevant states according to section IV. The dataset is imported to Matlab, where it is used to train the neural networks that constitute the nonlinearities of the model using the backpropagation algorithm previously described. Then the parameters of these neural networks are written to a datafile in an appropriate format. This datafile is then imported by a subroutine of the VHDL-AMS model, which uses the contained data to rebuild the mathematical representation of the neural networks according to (11).
To determine the standard deviations of the noise sources, either a phase noise simulation or, if available, phase noise measurements can be used. These standard deviations are entered directly as parameters to the VHDL-AMS model and will be used to create the random numbers according to III-B. Before running the simulation based on this model, important parameters of the simulation need to be set. (cf. chapter V).

IV. Parametrization of the Model

To match the model to a given circuit, the parameters of the neural networks representing $f()$ and $\psi()$ have to be found. This is done by fitting their response to a data set generated by a circuit level simulation. The theoretical background of this training process is given in chapter II-F. Another crucial part to get a working model is to properly design the training data. This section gives guidelines to training data design and presents the training results for the ANNs employed in the example of a millimeter wave oscillator.

A. State-Space as Training Aid

To approximate a function by a neural network, the ANN must yield the function value as output for all inputs of interest. These relevant inputs are located in a specific, limited part of the function's input space. The dimension of the input space is the number of scalar inputs of the function.
In the case of the present oscillator model, the states $x_1(t)$ and $x_2(t)$ are inputs of both employed neural networks, as displayed in figure 5. The state space of the oscillator is thus a subspace of the neural networks' input spaces. That is why the state space representation of the oscillator's behavior is very well suited for training data design. Previous considerations show that all practical inputs of the neural networks are located inside or on the limit cycle of the oscillator's trajectory.
Besides the states $x_1(t)$ and $x_2(t)$, the neural networks can have other inputs that act as parameters. The only parameter in the present example is v_{control}. Thus, for all valid control voltages (here from 0 V to 1 V) the function must yield correct values in the above mentioned part of the oscillator's state space.
If several parameters are used, all valid combinations of them have to be taken into account for training. Generating a complete training data set thus becomes more difficult for models with increasing number of parameter inputs.
The artificial noise input can be treated less strictly: Because the excitations on this port are small compared to the states but have a similar influence on the output, it is not necessary to consider this input as a separate dimension. Additionally, some small noise-like signals should be injected at this port to match the responses of circuit and neural network on these kind of excitations.

B. Interdependence of Inputs

The straight-forward way to generate the necessary training data is to sweep the inputs to get a coverage of the whole valid input space and calculate the outputs given by the original circuit level model based on them. However, as obvious from figure 5, the inputs (and in the case of function $f()$ the output as well) are connected outside the neural network and thus not sweepable independently. This fundamental difference compared to a neural network approximating simple input-output relationships necessitates a different approach to training data design. Furthermore, the inputs of the neural networks are not identical to the inputs of the system (they are rather states, which are chosen to be related to the output), so they are not accessible in the circuit simulator without changing the circuit response.
Independent input parameters like v_{control}, however, can be swept to generate training data for the whole range of valid values. With reasonable spacing between two discrete values, the whole parameter range can be covered by a reasonable number of simulation runs. In the example of v_{control}, generation of training data for the six values $0.0\,\mathrm{V}, 0.2\,\mathrm{V} \ldots 1.0\,\mathrm{V}$ yields an accurate model for all valid control voltages as can be confirmed by looking at the outputs of the model for $v_{\mathrm{control}} = 0.5\,V$ in chapter VI.

C. Training by Varying Trajectories

A typical startup trajectory of an oscillator covers more or less densely the inside and the rim of the limit cycle, as shown in figure 7. However, due to the high gain of the transistors used in the particular circuit producing this trajectory, large regions of the state space remain uncovered by training data. While for small gaps the neural network interpolates the behavior of the oscillator correctly, based on the provided training data, this is not true for large gaps:

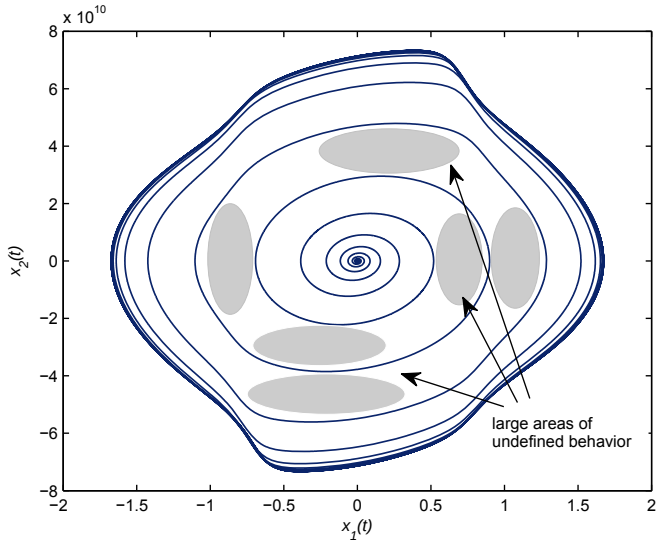


Fig. 7. Trajectory of an oscillator with fast startup: large uncovered areas in the input space yielding neural networks poorly approximating the desired function.

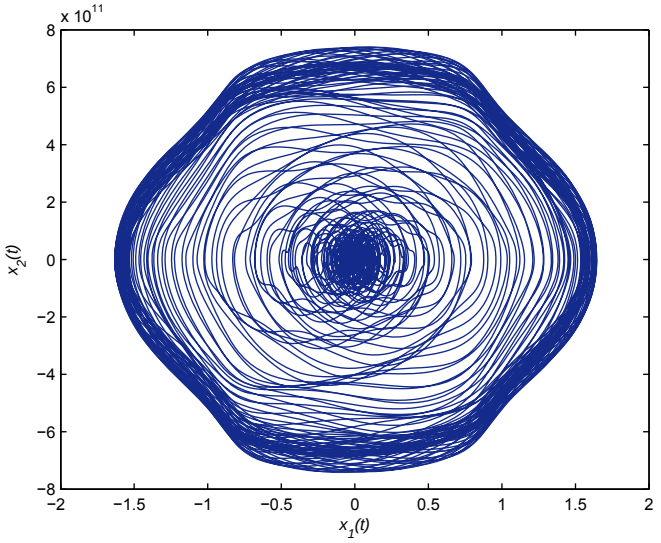


Fig. 8. Dense trajectory due to the use of the artificial current source for injection of a current that influences the trajectory

In the case of the second neural network, which is not part of a differential equation, this inaccuracy only increases the inaccuracy of the model output. The maximum error that can be tolerated depends on the application.

For the neural network approximating $f()$, dense coverage is crucial though: This ANN is part of a differential equation whose solution at one time step mainly depends on the solution of the previous time steps (and its derivatives). Although small errors yield only a slightly inaccurate solution at one time step, these errors sum up over time, leading to a deviation from the valid trajectory. If the trajectory leaves the well characterized inside of the limit cycle, this can finally result in a complete failure of the solution.

Note that for oscillators with a slow startup this problem is less pronounced, because one startup contains many periods and thus densely covers the input space as shown in figure 3. In order to generate dense training data from oscillators with fast transients, several startups are used. To change the conditions at and after startup in order to vary their trajectories, a high frequency, small amplitude signal can be injected to the artificial input port i_{n} (cf. figure 2). This signal must be differentiable, thus noise is not well suited. Either a sinusoid or a pulse train with smooth slopes may be used. As a result of this injection, a dense coverage of the input space is achieved and the neural network's response to excitations at this artificial port is matched to the circuit's response.

D. Accuracy Issues

Besides the dense coverage of the neural network's input space there are other important issues that have to be addressed during training in order to get an accurate, functional model. As indicated in section II-B, the behavior near the outside of the limit cycle has to be characterized as well. Otherwise, the differential equation can exhibit other, artificial attractors that are not present in the real oscillator. These attractors can render

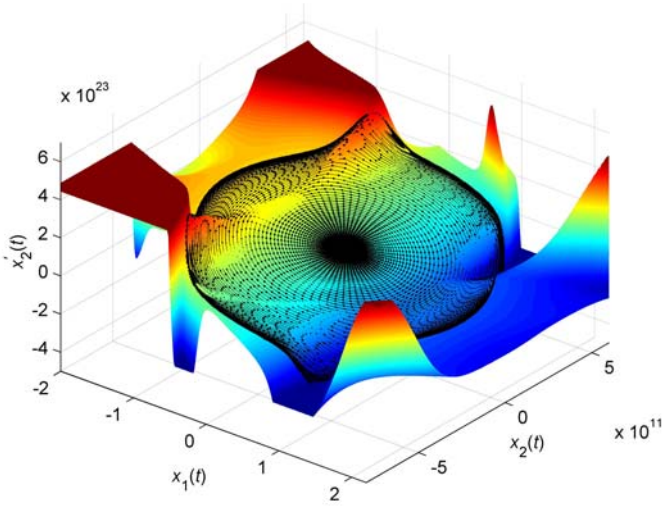


Fig. 9. Function $f()$ approximated by the neural network (colored plane) versus trajectory used to train it (solid line with dots to indicate actual values), $v_{\mathrm{control}} = 0\,V$.

the limit cycle unstable, in such a way that a large noise peak in steady state causes the oscillator to leave the limit cycle. To avoid this, the artificial input is used to generate steady state training data that deviates a certain amount from the ideal, noise-free trajectory. Thus, the function approximates the oscillator's behavior even in areas of the state space that are rarely encountered.

A potential source of inaccuracy is the error already present in the data used for training the ANN. All the more, because during the development of the oscillation an insignificantly small error amplifies and becomes considerable. To keep the error present in the training data set small, the circuit simulator needs to use extremely tiny time steps and small internal tolerances.

In this context, the accuracy of the derivatives that relate $x_2(t)$ to $x_1(t)$ and $\dot{x}_2(t)$ to $x_2(t)$ are of utmost importance. Because

a forward difference approach needs an impractically small time step to yield small errors, a central difference is used to compute the derivatives. Note that numerical derivatives calculated by higher order differences are an option to minimize this error. The advantage is that their use does not slow down the VHDL-AMS model while improving model accuracy. Simple forward difference calculations yield errors as large as some ten millivolts with time steps of only some femtoseconds in the 60GHz oscillator example, so they must be avoided.
Overtraining of the neural network also can become an issue: If the same set of training data is presented repeatedly during the training process, the neural network tends to specialize on this data, and thus looses its generalization property [13].

E. Training Results

The two networks of the model of the 60GHz differential VCO are trained using training data prepared according to the guidelines provided in the previous sections. The data is obtained from ADS simulations. The number of neurons is determined experimentally to get a mean squared training error (MSE) in the range of 10^{-5}. Note that this accuracy can be improved by increasing the number of nodes. However, the optimum parameter set is then more difficult to find, and the training process takes more resources. Furthermore, an even better accuracy conflicts with the goal to achieve complexity reduction by means of the behavioral model.

1) Neural Network to Represent $f()$: The neural network to represent $f()$ uses 15 neurons in the first and 10 neurons in the second hidden layer. It achieves an MSE of $1.25 \cdot 10^{-5}$ with respect to the training data presented. The training data consists of six startup trajectories with different control voltages and modulated artificial input current, plus one frequency sweep done by increasing v_{control} continuously from 0 V to 1 V.
Figure 9 compares the response of the neural network for $v_{\mathrm{control}} = 0\,\mathrm{V}$ to the trajectory of a startup that ends in steady state. They correspond very closely to each other, illustrated by the fact that the trajectory is in some places slightly above and in other places slightly below the plane that represents $f()$.
Figure 10 shows the output of the neural network versus the used training data over time for two different control voltages in steady state. Excellent agreement is observed, which confirms successful training for different control voltages. Higher maxima and a smaller period indicate the higher frequency of oscillation for $v_{\mathrm{control}} = 1.0\,V$

2) Neural Network to Represent $\psi()$: The neural network to represent $\psi()$, i.e. the mapping from the system states to the differential outputs, contains a single hidden layer of three neurons. The training data used characterizes the same states of the oscillator as the one for $f()$. An MSE of $5.54 \cdot 10^{-6}$ is achieved.
In figure 11 its steady state response is shown. (Similar accuracy is achieved for the startup trajectory. It is not shown here for the sake of clarity.) The 180° phase difference between the v_x and the v_y output is graphically represented by the fact that the two quasi-planes containing the trajectories only intersect

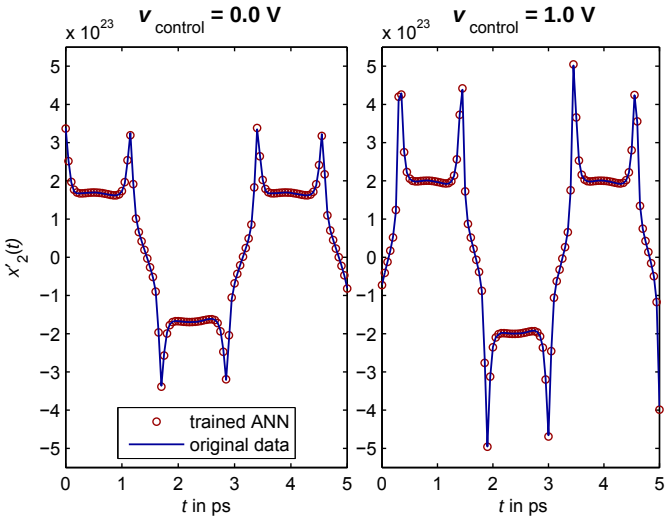


Fig. 10. Original data (blue solid line) versus output of trained neural network (red circles) for two different control voltages (left: 0 V, right: 1 V)

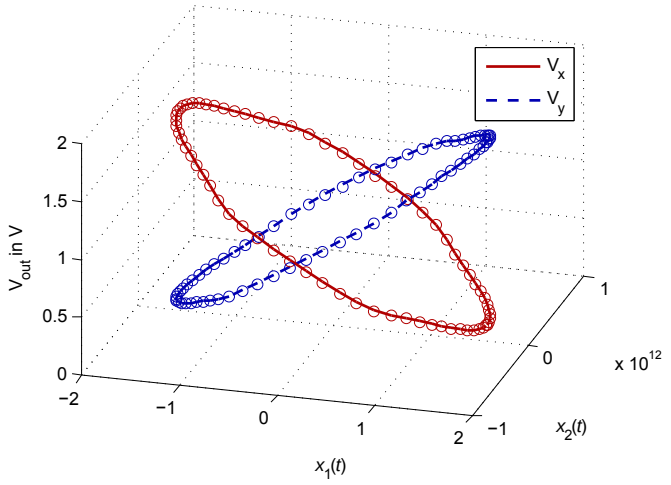


Fig. 11. Mapping from the states x_1 and x_2 of the system to its output voltages (function $\psi()$). dashed and solid line: original output, circles: output of neural network.

at the bias voltage $V_{\mathrm{bias}} = 920\,mV$ and exhibit a symmetry to the plane $x_1(t) = 0$.

V. Implementation and Solver Issues

The model structure according to figure 5 is implemented in VHDL-AMS. The results in section VI are obtained from this implementation, which is detailed and illustrated by some code examples in the appendix. The simulation environment used is SMASH[3], which is chosen due to the particular requirements of the model (cf. section V-A). An implementation of the model in other description languages and simulation environments is possible, as long as they provide similar directives and solver options as VHDL-AMS and SMASH.

A. Solver Issues

The correct numerical evaluation of the system of differential-algebraic equations described in VHDL-AMS de-

[3]SMASH 5.10.0, copyright 1992-2007 by Dolphin Integration

mands several prerequisites. Otherwise, the simulation does not yield correct results despite perfect model equations.

First of all, the numerical algorithm employed has to be chosen wisely. While backward Euler or trapezoidal algorithms work only when using prohibitively small time steps, Gear's algorithm is stiffly stable [18] and works fine for a reasonable range of time steps (whose upper bound depends on the oscillation frequency). It is furthermore supported by the major simulation environments, among them SMASH.

Secondly, the operating point simulation needs to start from initial system states that lie inside the limit cycle. Only then it is assured that the correct singular point that corresponds to the bias point of the oscillator is found. The reason is that the neural network representing the function $f(\,)$ is not trained far outside the limit cycle, and thus can exhibit artificial singular points there. Manually setting reasonable initial values for the state quantities remedies this problem.

The third issue concerns the tolerances maintained during the solution process. Due to the very high frequency microwave oscillators are working at, the states and their derivatives have largely different orders of magnitudes. For the 60 GHz VCO under consideration, $x_1(t)$ is on the order of 1 while $\dot{x}_1(t)$ is on the order of 10^{12}. Thus, the tolerances that need to be imposed on a quantity and its derivative are of totally different order of magnitude. However, to the best of the author's knowledge, SMASH is the only simulator that allows to set the tolerance value of the quantities and their derivatives independently. This is an essential feature without which it is hardly possible to solve the model equations correctly for millimeter wave oscillators.

VI. Realized Modeling Examples

A. Single-Ended Free-Running Oscillator

A first validation of a model using a more basic form of the presented modeling methodology is provided in [19]. The oscillator is free-running and has one single-ended output. Due to the simplicity of this architecture, no parameter inputs are necessary. The oscillator's output voltage and its derivatives are used as system states. A single neural network with one hidden layer is sufficient to create a behavioral model that shows excellent agreement with circuit level simulations.

B. Differential Colpitts-VCO

The results presented in this section belong to the differential 60 GHz Colpitts VCO taken as example throughout the paper. The parameters of the employed neural networks correspond to the ones obtained by the training process documented in chapter IV-B. If not mentioned otherwise, $v_{\mathrm{control}} = 0.5\,\mathrm{V}$. This voltage has not been employed for the startups in training. The ADS results which the VHDL-AMS simulations are compared to are created exclusively for model verification. Thus, the simulation results in this section do not represent fitted curves, but rather show the excellent generalization properties of the model. To reduce the influence of the solver accuracy on the results, the maximum time step is chosen to be 50 fs for both VHDL-AMS and ADS.

Figure 12 compares oscillator startups generated by the VHDL-AMS model and ADS. The excellent agreement in steady state is demonstrated by the zoomed-in view in the same figure. The 180° phase difference between the two outputs is well maintained by the behavioral model. The minimum and maximum values are 0.147 V and 1.731 V (VHDL-AMS) versus 0.137 V and 1.745 V (ADS). The oscillation frequency is 61.16 GHz (VHDL-AMS) versus 61.15 GHz (ADS).

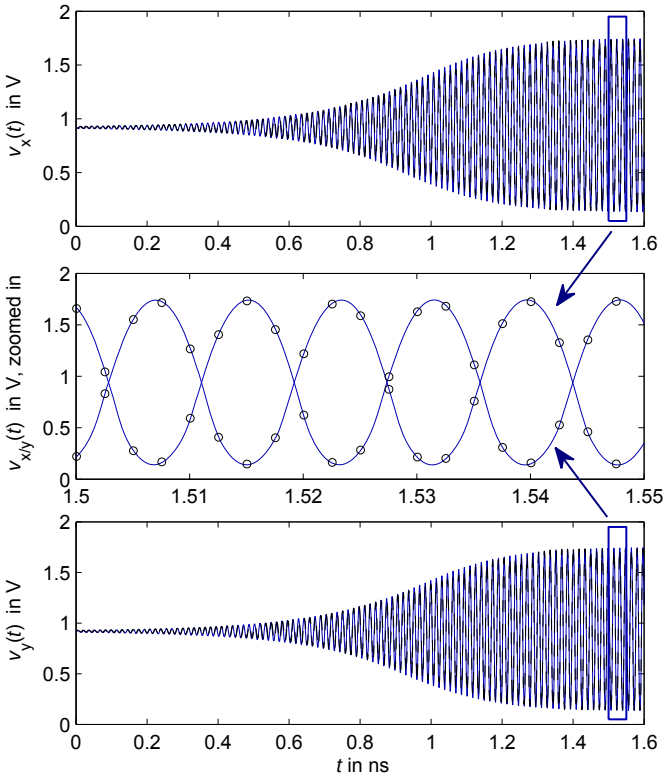


Fig. 12. Output of the VHDL-AMS model (dotted black line or markers) versus ADS simulation results (solid blue line) for $v_{\mathrm{control}} = 0.5\,\mathrm{V}$

The transient envelopes as a whole correspond very well. The amplitude difference is about 60 mV around $t = 1.05$ ns due to the fact that the behavioral model is only of order two and thus cannot represent all the dynamics necessary to exactly simulate the startup process. However, the achieved accuracy is more than sufficient for a behavioral model. This is also confirmed by the transient simulations of a real-world VCO discussed in section VI-C.

Figure 13 gives a plot of the even and odd mode behavior of the oscillator. The swing of the odd mode oscillation is 1.612 V_{pp} (VHDL-AMS) versus 1.585 V_{pp} (ADS). A small discrepancy during the transient is observed here as well.

The even mode plots show the very close agreement of the bias voltage (919 mV VHDL-AMS versus 921 mV ADS) and the response in steady state, even though the model is not optimized in this regard. In the transient part of the even mode plot the agreement is slightly degraded, for the reason discussed above. However, considering the mV scale of the ordinate, this error has small impact on the overall simulation.

Figure 14 shows the results of a frequency sweep when continuously increasing v_{control} from its minimum to its maximum value. The lowest frequency is 58.79 GHz (VHDL-

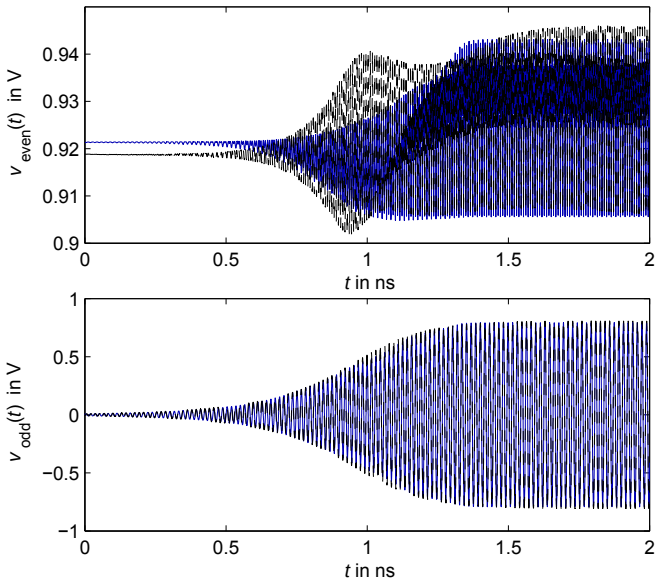


Fig. 13. Output of the VHDL-AMS model (dotted black line) versus ADS simulation results (solid blue line) in even mode ($v_{\mathrm{even}}(t) = (v_{\mathrm{x}}(t) + v_{\mathrm{y}})/2(t)$) and odd mode ($v_{\mathrm{odd}}(t) = (v_{\mathrm{y}}(t) - v_{\mathrm{x}}(t))/2$).

AMS) versus 58.72 GHz (ADS), the highest frequency is 63.91 GHz (VHDL-AMS) versus 63.79 GHz (ADS). The slight phase difference between the VHDL-AMS curve and the ADS-curve apparent from the zoomed-in view is due to this slight difference in frequencies. Compared to the dispersion due to process variations an error of this magnitude is negligible.

Note that during the change of v_{control} the two curves follow each other very closely. This reflects the fact that the behavioral model correctly represents the dynamics when changing v_{control}. The model can thus be used to simulate systems where these dynamics are important, as for example PLLs.

Figure 15 shows the phase noise plot created by the VHDL-AMS simulation in SMASH. The asymptotes can be placed by setting the standard deviations σ_1 and $\sigma_{2x/y}$ in order to closely resemble the original phase noise plot. The greater density at higher frequencies, accompanied by stronger deviations from the asymptotes, are due to the fact that the simulation is done on a linear scale, but the plot is logarithmic. The slight decrease of the noise level in the flat region at highest frequencies is due to the fact that the additive Gaussian noise generated is not actually white, but its power density decreases when approaching the sampling frequency.

C. Integrated 60 GHz VCO in 65 nm CMOS including Buffers

In order to demonstrate the applicability of the proposed modeling technique to real oscillators fabricated in a submicron CMOS semiconductor process, it is applied to a 60 GHz VCO in fabrication in the 65 nm CMOS technology of ST Microelectronics. The simplified schematic (with neither bias circuitry nor extracted parasitics) is shown in figure 16. The Colpitts VCO uses differentially tuned accumulation-MOS varactors, which are represented during simulation by BSIM4 models. The BSIM4 model is also used for the transistors. The inductors are characterized by $2-\pi$ models extracted from electromagnetic field simulations. Source-follower buffers are employed to minimize loading of the oscillator core. All component values and device widths are annotated in the corresponding schematic in figure 16. The artificial current source $i_{\mathrm{n}}(t)$ is added to the schematic after the design phase to generate the training data as described in section IV-C.

The design and layout of the oscillator was done using

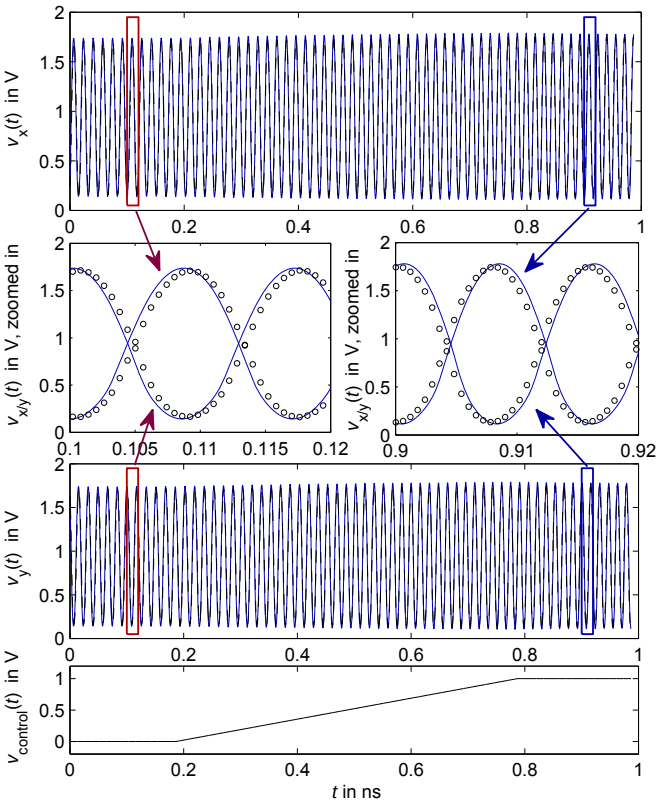


Fig. 14. Output of the VHDL-AMS model (dotted black line or markers) versus ADS simulation results (solid blue line) for control voltage sweep

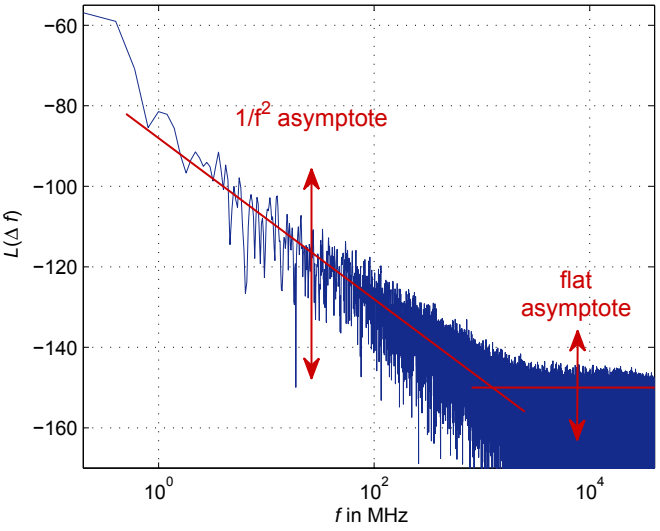


Fig. 15. Phase noise characteristics shown by the VHDL-AMS model. The height of the asymptotes is set by σ_1 and σ_{2x}/σ_{2y}.

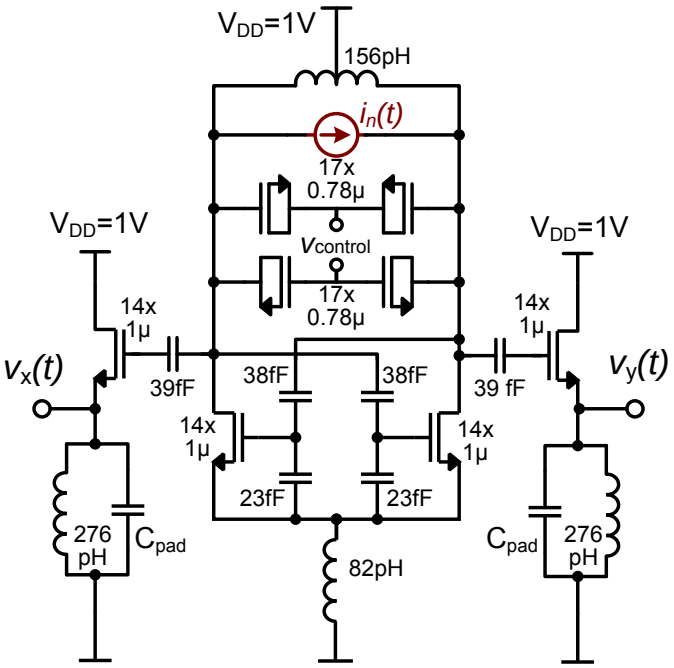


Fig. 16. Schematic of the 65 nm CMOS 60 GHz VCO

the Cadence IC Design Tools, Version 5.10.41[4]. The circuit simulator SPECTRE, which is part of the Cadence framework, is used to calculate the circuit behavior based on the extracted schematic. To give an impression of the complexity of the oscillator, table I summarizes the circuit inventory established during SPECTRE simulation. This entire circuit is taken into account during model generation, and the evaluation of the behavioral VHDL-AMS model is done with respect to SPECTRE simulations of this circuit. As before, the state

nodes	938
equations	2560
bsim4	140
capacitor	2166
inductor	16
resistor	2318

TABLE I
CIRCUIT INVENTORY OF THE SPECTRE MODEL

vector of the behavioral model comprises the difference of the output voltages, $v_y(t) - v_x(t)$, as first state variable and the derivative thereof as second state variable. Note that the output buffers are included inside the VHDL-AMS model, so $v_y(t) - v_x(t)$ does not correspond to a voltage observed across a circuit element.

The structure of the VHDL-AMS model is the one of figure 5. The ANN used to approximate $f()$ contains 18 neurons in the first and 13 neurons in the second hidden layer. The ANN used to represent $\psi()$ employs 3 neurons in a single hidden layer. The training employs startup waveforms with v_{control} varying from -0.7 V to 0.7 V in steps of 0.1 V and yields an MSE of 2.30E-5 for $f()$ and an MSE of 1.59E-4 for $\psi()$.

[4]Copyright 1992-2005 Cadence Design Systems, Inc.

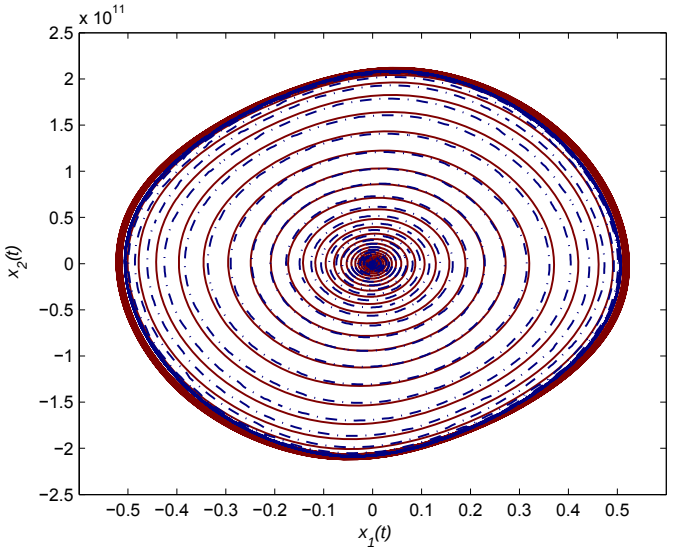


Fig. 17. Trajectory obtained from SPECTRE-simulation (red, solid line) versus VHDL-AMS model (blue, dotted line)

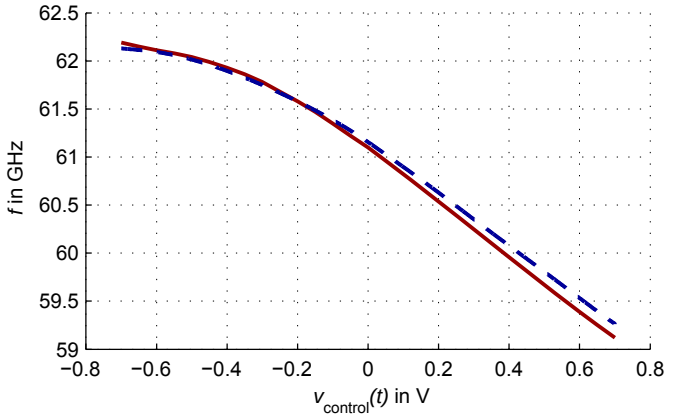
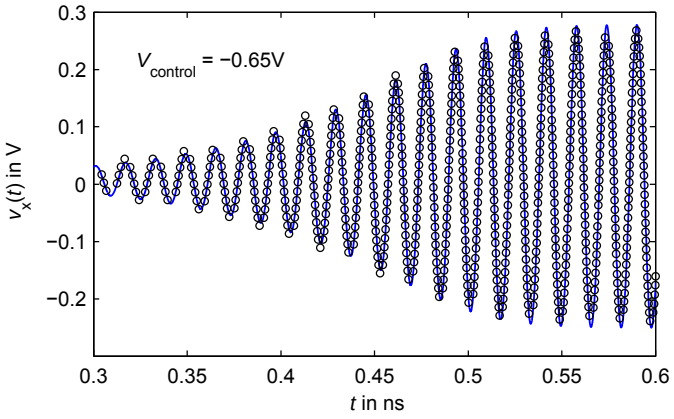


Fig. 18. Tuning-curve of the oscillator, SPECTRE (solid line) versus VHDL-AMS (dashed line)

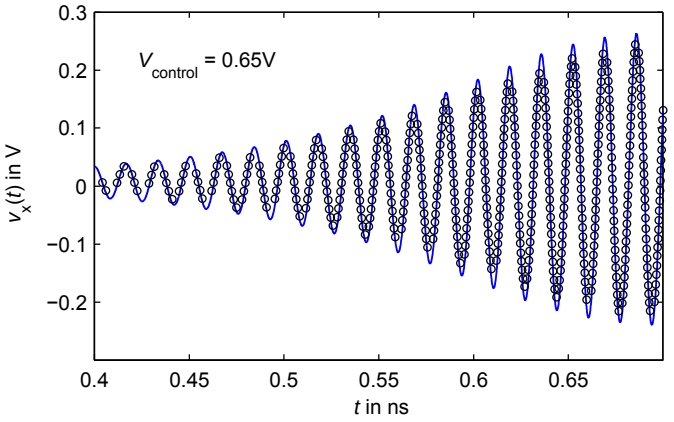
ANNs with fewer neurons yield only little worse MSEs. The VHDL-AMS model based on these ANNs is used to do the performance evaluation in the following sections.

1) Accuracy of the behavioral VHDL-AMS Model: The behavioral model's steady state accuracy is demonstrated by comparisons of both the trajectories and the tuning curves between the SPECTRE circuit model and the behavioral VHDL-AMS model.

In figure 17 the oscillation trajectories are plotted for $v_{\text{control}} = 0.0$V. They resemble each other very closely. The form of the limit cycle attained in steady state has the same shape for both of the models. This confirms that an enormous order reduction is possible even in the case of a real 60 GHz sub-micron CMOS VCO: The system order can be successfully reduced from more than two thousand to two. Differences between the two trajectories can be explained by the influence of noise and the solvers' accuracies, which is difficult to align between the two simulation environments. Furthermore, the accuracy in steady state can be confirmed by the tuning curve of figure 18, where the control voltage



(a) $v_{\mathrm{control}} = -0.65\mathrm{V}$



(b) $v_{\mathrm{control}} = +0.65\mathrm{V}$

Fig. 19. Startup waveform at output x of oscillator. VHDL-AMS model (markers) versus SPECTRE - simulation (solid line).

is swept over the whole tuning range ($v_{\mathrm{control}} = -0.7\,\mathrm{V}$ to $+0.7\,\mathrm{V}$). Excellent agreement is achieved, showing that the nonlinear curvature of the tuning curve is indeed also present in the behavioral model, even though the varactors are not explicitly modeled.

To assess the accuracy in the transient regime, figure 19 shows the startup for two different control voltages. Good agreement between circuit model and behavioral model can be observed in both cases. The fact that the amplitude differs by several mV between both models is explained by the different tolerances of the solver, despite using the same maximum time-step of 50 fs. Higher accuracy can be obtained by employing smaller time steps at the expense of longer computation time, as used for training data generation (cf. section IV-D).

The fact that startups for two different control voltages (and thus oscillation frequencies) show different transient envelopes (compare figure 19 (a) and (b)) is also correctly taken into account by the behavioral VHDL-AMS model.

2) Computation Time Comparison: In order to assess the speed-up achieved by the behavioral model with respect to the circuit model, their simulation times are compared in table II. The SPECTRE simulation is executed on a machine equipped with two 3.2 GHz Intel PENTIUM 4 processors and 3 GB RAM, using Cadence 5.10.41. The VHDL-AMS simulation is executed on an Intel Core 2 6700 2.66 GHz processor with 3 GB RAM, using SMASH 5.10. Both simulations are done single-threaded.

SPECTRE simulation, with transient noise	30 min 43.6 sec
SPECTRE simulation, without transient noise	23 min 43.7 sec
VHDL-AMS simulation	1 min 13.2 sec

TABLE II
SIMULATION TIME FOR START-UP AND SUBSEQUENT STEADY STATE OF IN TOTAL 10 NS

The simulation scenario consists of the start-up and subsequent steady state for $v_{\mathrm{control}} = 0V$ which takes in total 10 ns. The maximum time step chosen is 50 fs, the relative accuracy is 1e-5, and the used solver is *Gear*.

In the case of the SPECTRE simulation, two different setups are evaluated: The default case is a simulation which does not take into account transient noise. This is the only one supported in older versions of SPECTRE. However, because the behavioral VHDL-AMS model does simulate transient noise, this kind of simulation is more appropriate for comparison. Thus its execution time is also given.

In both cases the comparison confirms the effect expected from the order reduction: A tremendous speed-up by a factor of up to 25 is achieved for the present simulation setup, while maintaining the accuracy shown in the previous section. This speed-up is less pronounced for circuits of lesser order, because the achievable order reduction is smaller. On the other hand, an even higher speedup would be possible when optimizing the implementation of the reduced-order model.

Altogether, the strong interest of replacing the circuit level model by the proposed behavioral VHDL-AMS model after the design phase is confirmed. The computation time necessary to create the training data and train the ANNs is compensated by much faster execution of the behavioral model.

VII. CONCLUSION

This article describes a methodology to create behavioral models of microwave oscillators that can be implemented in a hardware description language like VHDL-AMS. The model is based on data obtained from a transistor level simulation using models of high order and complexity. The novel modeling methodology uses this data to parametrize a model structure that has only an order of two, but is capable of approximating a high degree on nonlinearity by artificial neural networks. The technique is applicable to state-of-the-art voltage controlled oscillators, whose behavior concerning transient, steady state and phase noise is faithfully reproduced by the presented model. Comparison of the outputs of this model and circuit level simulations show excellent agreement in all operating regimes, while the behavioral model achieves a reduction in computation time of up to 96%.

The field of application of such an accurate behavioral model are simulations where the complexity of an extracted circuit model is too high, yet an accuracy close to the one of such a model is desired. Examples could be simulations of phase locked loops or systems on chip, where the influence of the oscillator's peculiarities is decisive to assess the system performance. While even simpler models that avoid

actually solving differential equations by just plotting the (pre-calculated) solution are faster, they result in a reduction in both accuracy and flexibility.

ACKNOWLEDGMENT

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

APPENDIX
STRUCTURE OF THE VHDL-AMS CODE

The core of the VCO model is a VHDL-AMS entity with four ports, `p1` to `p4`. The first port is connected to a grounded noise current source, implemented as described in [17], in order to provide i_{n}. Port two and three represent the oscillator's outputs v_{x} and v_{y} with respect to ground. Two series noise voltage sources [17] that generate the additive noise for the flat region of phase noise are series-connected externally to these ports. The oscillator's output signal including phase noise is taken at the open pins of these noise sources. The forth port is connected to the v_{control} input.
In the architecture definition of the VCO entity, the currents through and voltages across the terminals are defined as quantities according to:

```
quantity In through p1 to ELECTRICAL_REF;
quantity voutx across p2 to ELECTRICAL_REF;
quantity vouty across p3 to ELECTRICAL_REF;
quantity vcontrol across p4 to ELECTRICAL_REF;
```

Other quantities are associated to the normalized inputs, the state variables and their derivatives, the nodes of the neural networks, and other auxiliary variables:

```
quantity In_norm : real tolerance "NORMALIZED";
quantity x1      : real tolerance "NORMALIZED";
quantity x2      : real tolerance "NORMALIZED";
quantity dx2_dt  : real tolerance "NORMALIZED";
quantity Vcontrol_norm : real
                         tolerance "NORMALIZED";
                    ...
quantity nodeh1  : real_vector(1 to 15);
quantity nodeh2  : real_vector(1 to 10);
```

The `TOLERANCE` keyword specifies the tolerance used by the solver to evaluate different groups of quantities.
After defining the quantities, constants are defined that contain the parameters of the neural networks, such as weights, biases and normalization constants. To load the actual values from the text files exported by Matlab, functions created for this purpose are called in the initialization process.
The body of the entity's architecture between the `begin` and `end` keyword consists of the definition of the state space equations according to (8) and (9), the definition of the equations describing the two neural networks according to (11), and the associated normalization and denormalization.
To implement the derivatives that appear in the state equations, the `'dot`- statement is used:

```
x2 == x1'dot * norm_const1;
dx2_dt == x2'dot * norm_const2;
```

The more complex equations describing the neural network make use of the `GENERATE` statement to automate the formulation of lengthy equations:

```
BuildInputLayer: FOR i IN 1 TO 15 GENERATE
nodeh1(i) == tanh( w1(i*4+1) * x1 +
                   w1(i*4+2) * x2 +
                   w1(i*4+3) * In_norm +
                   w1(i*4+4) * Vcontrol_norm +
                   b1(i) );
                   END GENERATE;
-- build second hidden layer here:
                    ...
-- output layer:
dx2_dt == tanh( w2(1) * nodeh2(1) +
                   w2(2) * nodeh2(2) +
                    ...
                   w2(10) * nodeh2(10) +
                   b2(1) );
```

Note that in the code fragment above `w1(), w2(), b1()` and `b2( )` are vectors containing the weights and biases of the neural network. Due to the lack of matrix operators in the VHDL-AMS definition [1] these equations cannot be written more efficiently.

REFERENCES

[1] D. A. S. C. of the IEEE Computer Society, "IEEE standard VHDL analog and mixed-signal extensions: IEEE std 1076.1-1999," 1999.

[2] D. Root, J. Wood, and N. Tufillaro, "New techniques for non-linear behavioral modeling of microwave/RF ICs from simulation and nonlinear microwave measurements," in *Design Automation Conference, 2003.*, 2-6 June 2003, pp. 85–90.

[3] M. Sobhy, E. Hosny, M. Ng, and E. Bakkar, "Nonlinear system and subsystem modeling in time domain," *IEEE Transactions on Microwave Theory and Techniques*, vol. 44, no. 12, pp. 2571–2579, December 1996.

[4] D. Schreurs, "Overview of non-linear device modelling methods based on vectorial large-signal measurements," in *Gallium Arsenide Applications Symposium*, October 1999, pp. 381–386.

[5] A. Fakhfakh, N. Milet-Lewis, J. Tomas, and H. Levi, "Behavioural modelling of phase noise and jitter in voltage-controlled oscillators with VHDL-AMS," in *ICCSC 2002*, 26-28 June 2002, pp. 370–373.

[6] P. Nikitin, E. Normark, C. Wakayama, and C.-J. R. Shi, "VHDL-AMS modeling and simulation of BPSK transceiver system," in *IEEE International Conference on Circuits and Systems for Communications, Moscow, Russia*, June 2004.

[7] B. Van der Pol, "The nonlinear theory of electric oscillations," *Proceedings of the Institute of Radio Engineers*, vol. 22, no. 9, pp. 1051–1084, September 1934.

[8] J. Wood, D. Root, and N. Tufillaro, "A behavioral modeling approach to nonlinear model-order reduction for RF/microwave ICs and systems," *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 9, pp. 2274–2284, September 2004.

[9] Y. Fang, M. Yagoub, W. Fang, and Q. Zhang, "A new macromodeling approach for nonlinear microwave circuits based on recurrent neural networks," *IEEE Transactions on Microwave Theory and Techniques*, vol. 48, no. 12, pp. 2335–2344, December 2000.

[10] J. Xu, M. Yagoub, R. Ding, and Q.-J. Zhang, "Neural-based dynamic modeling of nonlinearmicrowave circuits," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 12, pp. 2769–2780, December 2002.

[11] A. Suarez and R. Quere, *Stability Analysis of Nonlinear Microwave Circuits*. Artech House, Inc., 2003.

[12] A. Grebennikov, *RF and Microwave Transistor Oscillator Design*. John Wiley & Sons Ltd, 2007.

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

[14] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, December 1989.

[15] D. B. Leeson, "A simple model of feedback oscillator noise spectrum," *Proceedings of the IEEE*, vol. 54, pp. 329–330, February 1966.

[16] T. H. Lee, *The Design of CMOS Radio-Frequency Integrated Circuits*. Cambridge, UK: Cambridge University Press, 1998.

[17] E. Normark, L. Yang, C. Wakayama, P. Nikitin, and R. Shi, "VHDL-AMS behavioral modeling and simulation of a $\pi/4$ DQPSK transceiver system," in *BMAS 2004*, 21-22 Oct. 2004, pp. 119–124.

[18] C. Gear, "Simultaneous numerical solution of differential-algebraic equations," *IEEE Transactions on Circuits and Systems*, vol. 18, no. 1, pp. 89–95, January 1971.
[19] M. Kraemer, D. Dragomirescu, and R. Plana, "Nonlinear behavioral modeling of oscillators using artificial neural networks," in *IEEE Radio Frequency Integrated Circuits Symposium*, June 2008, pp. 689–692.

Michael Kraemer (M'07) was born in Bad Mergentheim, Germany, in 1980. He received the Dipl.-Ing. (BA) degree from BA Mosbach, Germany, in 2003 and the Dipl.-Ing. degree from the University of Stuttgart, Germany, in 2007, both in electrical engineering and is currently working toward his Ph.D. degree at LAAS-CNRS in Toulouse, France. During his studies, he stayed one year at the University of Massachusetts, Amherst, USA, and did internships at ebm-papst GmbH, DaimlerChrysler AG and Robert Bosch GmbH in Germany. His research interests include the design and modeling of microwave circuits.
Mr. Kraemer is a Fulbright fellow and the recipient of the 2007 VDE-award at the University of Stuttgart.

Daniela Dragomirescu (M'96) was born in Bucharest, Romania in 1972. She received the Electronics and Telecommunication engineering degree from Bucharest Polytechnic University, Romania in 1996 and her Ph.D. degree from the National Institute of Applied Sciences, Toulouse, France.
She is an Associate Professor at National Institute of Applied Sciences, Toulouse and a researcher at Laboratory for Analysis and Systems' Architecture (LAAS-CNRS). Her main research interests are in the Wireless Sensor Network, system architecture and modeling and circuit design.

Robert Plana was born on March 1964 In Toulouse. He obtained its PhD in 1993 at LAAS-CNRS and Paul Sabatier University on the Noise modelling and characterization of Advanced Microwave devices (HEMT, PHEMT and HBT) that includes the reliability. In 1993, as associate professor at LAAS-CNRS, he has started a new research area concerning the investigation of millimeterwave capabilities of Silicon based technologies. More precisely, he has focussed on the microwave and millimeterwave properties of SiGe devices and their capabilities for low noise circuits. In 1995, he has started a new project concerning the improvement of the passives on silicon through the use of MEMS technologies. In 1999, he has been involved with SiGe Semi-conductor in Ottawa where he was working on the low power and low noise integrated circuits for RF applications. In the same year, he has received a special award from CNRS for his works on Silicon based technologies for millimeterwave communications. In 2000, he has been professor at Paul Sabatier University and Institut Universitaire de France and he has started a research team at LAAS-CNRS in the field of Micro and Nanosystem for RF and millimeterwave communications. Its main interests are on the technology, design, modelling, test, characterization and reliability of RF MEMS for low noise and high power millimeterwave applications and the development of the MEMS IC concept for smart microsystem. He has built a network of excellence in Europe in this field "AMICOM" regrouping 25 research groups. He has authored and co-authored more than 200 international journals and conferences. In 2004, he has been appointed as Deputy Director of the Information and Communication Department at the CNRS Headquarter. From January 2005 to January 2006, he has been appointed director of the Information and Communication Department at CNRS. He is now heading a research group at LAAS-CNRS in the field of Micro and Nanosystem for wireless communications.