Efficient Modeling of Complex Analog Integrated Circuits Using Neural Networks

Ramin M. Hasani*, Dieter Haerle** and Radu Grosu*
*Institute of Computer Engineering
Vienna University of Technology, Vienna, Austria
Email: ramin.hasani@tuwien.ac.at
**KAI Kompetenzzentrum Automobil- und Industrieelektronik GmbH, Villach, Austria
Email: dieter.haerle@k-ai.at

***Abstract*—This paper introduces a black-box method for automatically learning an approximate but simulation-time efficient high-level abstraction of given analog integrated circuit (IC). The learned abstraction consists of a non-linear auto-regressive neural network with exogenous input (NARX), which is trained and validated from the input-output traces of the IC stimulated with particular inputs. We show the effectiveness of our approach on the power-up behavior and supply dependency of a CMOS band-gap reference (BGR) circuit. We discuss in detail the precision of the NARX abstraction, and show how this model can be used and implemented in testing of Analog ICs within the Cadence environment. By using our method one can automatically learn high-level abstractions of all the components of an Analog IC. This dramatically speeds up the transient simulation time of the Analog ICs.**

I. Introduction

Fault detection and fault coverage is a major concern for all companies producing analog or mixed analog-digital integrated circuits [1]. However, the associated tools and techniques are still in their infancy. No general solution exists so far, that can be considered experimentally verifiable and time efficient [2]. The problem can be traced back to the analog part, where the very notion of what constitutes a fault can be very fuzzy. Unfortunately, even a single transistor-level simulation in spice-like simulators, depending on the complexity of the circuit, may take from one day to several weeks. A compositional co-simulation approach would seem to be appropriate and highly desirable in such situations. In this approach, most parts of the circuit are efficiently co-simulated at a higher level of abstraction, for example in Verilog-A. Indeed, Verilog-A simulations of transistor-level abstractions, were shown to bring speed ups of up to thousand times, without degrading the simulation accuracy at unacceptable levels. However, an important question still remains unanswered: How is one going to arrive at these high-level abstractions?

In this paper we propose a machine-learning-based approach to learn a high-level abstraction, which utilizes input-output time series, to automatically synthesize a nonlinear autoregressive neural-network model with exogenous input (NARX). We assume that each part of the system to be abstracted is known, and small enough to be extensively simulated, in order to obtain the required input-output time series.

Abstracting integrated circuits (ICs) by neural networks has been recently used in the field of electromagnetic compatibility (EMC) testing, where the authors model a BGR circuit with a Neural network (NN) written in Verilog-A [3]. The model attained a reasonable time performance. We show that by using a NARX NN abstraction, we can improve the time performance of the transient simulation of a Band-Gap Voltage reference circuit by a factor of 16. Moreover, another advantage of using an NN abstraction for modeling an analog circuit is that an existing block-level test bench can be used to train the network. This implies that even without a deep understanding of the circuit, one can perform the fault simulations.

The paper is organized as follows; We introduce a CMOS band gap voltage reference circuit, an advanced analog circuit which has been used as the device under test (DUT) to be modeled, in Section II. Then in Section III, we describe in details our neural network architecture chosen for modeling the BGR circuit. Subsequently, we explain the training process and tools used in order to prepare the network, in Section IV and finally we define a method for engaging the designed neural network within Cadence AMS Designer environment.

II. Band-gap voltage reference circuit

The band-gap voltage reference (BGR) circuit is widely used in integrated circuits. Therefore, it is a suitable case study for efficient behavioral modeling. Figure 1 represents the symbolic schematic view of a CMOS BGR circuit. The circuit is supposed to deliver a constant output voltage (in our case 1V) regardless of temperature changes, power supply variations and different load profiles, when it gets activated. The power supply (V_{DD}) is employed as the activation signal of the circuit. Note that we target to model the power-up behavior and voltage dependency of the BGR circuit.

III. NARX neural network architecture

In order to automatically learn a neural network abstraction which is able to model transient behavior of an analog circuit, one should refer to nonlinear system identification tools where a predictive dynamic model is employed for solving a non linear time series problem. The nonlinear auto-regressive network with exogenous (external) input (NARX) is a feedback based

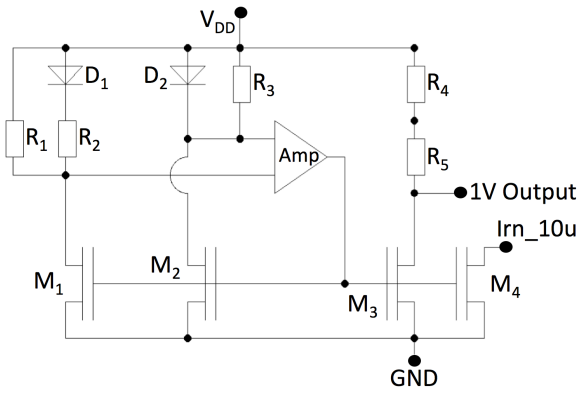

Fig. 1. Band-gap voltage reference circuit symbolic schematic

neural network realizing a recurrent dynamic network [4]. The model can be defined by the following equation:

$$y(t) = f(x(t-1), x(t-2), ..., x(t-n_x), \\ y(t-1), y(t-2), ..., y(t-n_y)) \quad (1)$$

where the next value of the output depends on the delayed version of the output signal, $y(t)$, and an external input signal, $x(t)$. One of the most important application of NARX models is the modeling of nonlinear dynamic systems. We chose NARX models over other predictive system identification methods, because of two advantages it provides. First, the use of feedback gives the network the opportunity to generate a more precise output based on the previously generated outputs. Second, the simultaneous use of delayed output data and input supplies the network with an additional degree of control.

The top-view of the structure of the designed NARX network is shown on the top left side of the Figure 2. It realizes a feed-forward neural network topology where the output, Y1, is fed back to the input stage as a part of the standard NARX architecture. An extended view of the hidden layer is additionally represented in Figure 2. Delay components define the number of variables used inside the model. In our case, we experimentally choose 3 delay components which provides us with six variables (three lagged inputs as well as three lagged outputs). Weighted lagged inputs and outputs synapse into seven neurons (with hyperbolic tangent sigmoid transfer function) and generate the output. The network is designed within Simulink MATLAB environment.

IV. Training Process and Network Performance

For training the NARX network we took some input data (to be used as the exogenous input of the network) and output data (to be used as the target values of the corresponding inputs) from a 40 ms transient simulation of the BGR circuit. Let us suppose the output of our network shown in Figure 3, Y1, is an estimation of the BGR circuit we are modeling. The output then is fed back into the input stage of the feed-forward network. As we know the exact output values (Target values) during the training process, we could cut the feedback and apply the target values directly to the Y input of the feed-forward network. In this way we gain more accuracy because

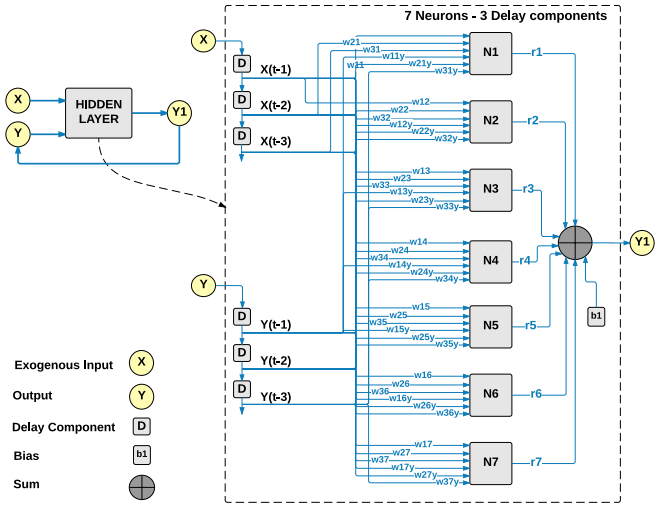

Fig. 2. NARX neural network architecture. generated output of the network can be fed back to the input depending on the application. For training purposes it is better to use the network in a feed-forward topology.

of the precise data instead of the estimated data. Moreover, since the network has feed-forward architecture now, we can use back-propagation algorithm for training [4].

A. Training Process

We randomly divided the input data and target data into three data-sets as follows:

- Training set (70%): This data is used for training.
- Validation set (15%): This data is utilized for stopping the training process after network generalization does not improve anymore.
- Testing set (15%): This data provides an extra evaluation of the network performance during and after the training phase. It has no effect on the training itself.

We train the network by using classic Leverberg-Marquardt back-propagation algorithm [5]. We simply use the function $(trainlm)$. The training process is continued until the error related to the validation data-set stops decreasing for six consequent iterations. Figure 3 represents the training performance of our NARX network. The process stopped after 15 epochs. The mean squared error (MSE) for the training, Validation and test data is calculated in Table I.

TABLE I
Achieved Mean Squared Error for Different Data-sets

Data-set	MSE
Training	3.1×10^{-4}
Validation	1.9×10^{-4}
Testing	3.9×10^{-4}

B. Network Performance

In order to evaluate the performance of the network we calculate the linear regression between target values and the

output data of the NARX network. Figure 4 indicates how good the output data follows its target with an overall correlation of 0.99928.

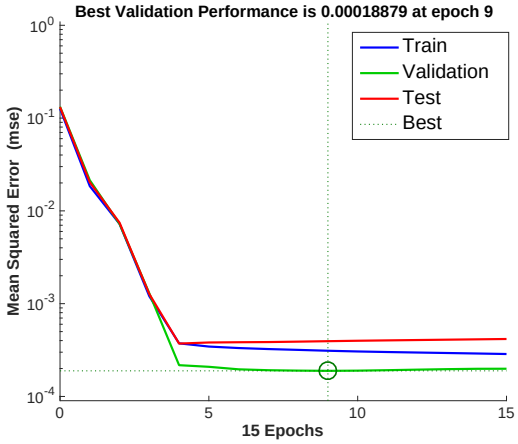

Fig. 3. Training performance.

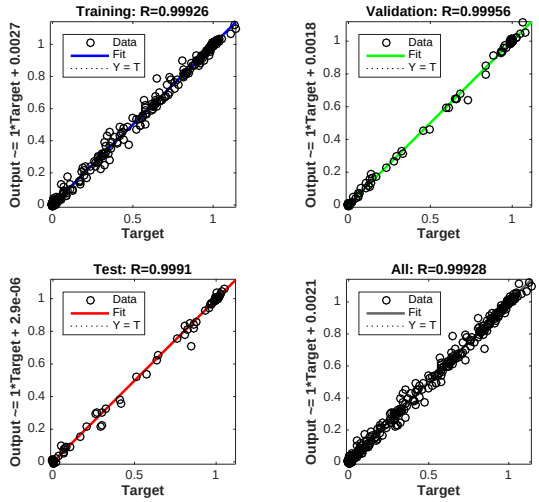

Fig. 4. Linear regression represented for training, validation, testing and all data-sets. note how data are well-fitted around the 45° line(target values).

Moreover, the error histogram of all samples is shown in the Figure 5 where we observe over 90% of the samples have an error between -0.006 and 0.008.

The error auto-correlation function describes how the computed output errors are related in time. The error auto-correlation is shown in Figure 6. For a perfect model, there should only be one nonzero bar occurring at zero lag. In our case, we notice that the correlations, except form the bar at the zero lag and the forth lag, are roughly within the 95% of the confidence limit around zero. This implies that the trained network is adequately accurate.

Figure 7 depicts the response of the BGR (target values) together with the response of its NARX NN to different shapes

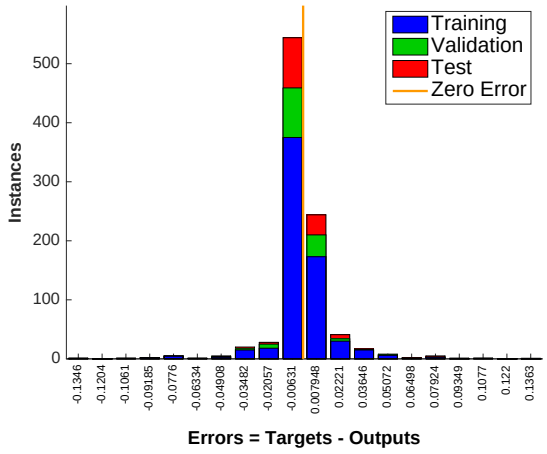

Fig. 5. Error Histogram. The denser the histogram is the more accuracy is gained after the training process.

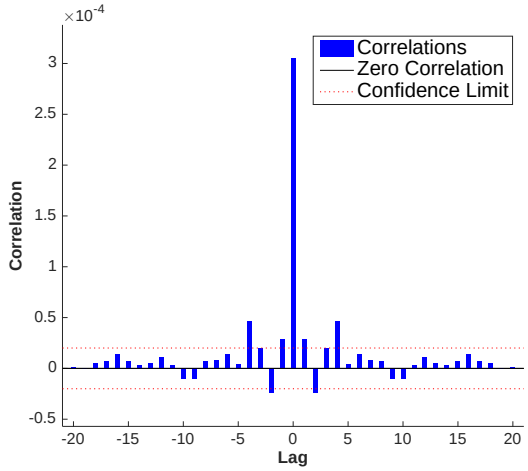

Fig. 6. Error auto-Correlation. This can validate the network performance.

of input signal (red solid line) and the errors. Note how the training, validation and testing data are distributed and how rigorous the network response follows the target values.

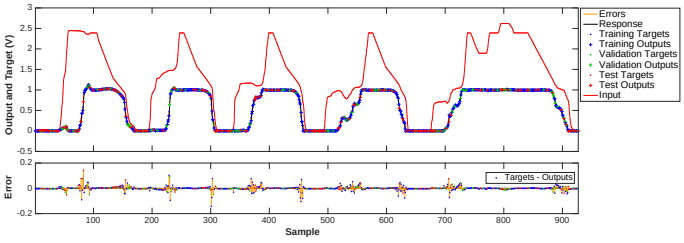

Fig. 7. The output response of the BGR circuit (blue solid line) to an input signal (V_{DD}) shown as red solid line, together with its NARX neural network model responses. The bottom figure qualitatively indicates the error between the BGR output response and the neural network response.

V. Co-simulation of AMS systems with MATLAB (Simulink)

In this section we present a method for employing our NARX network inside Cadence environment to check the time performance of a transient simulation within Cadence Analog Design Environment (ADE) where we mostly perform analog fault simulations. We utilize the Cadence Virtuoso AMS Designer co-simulation interface [6] which provides us with fast bidirectional connection between AMS designer simulator and MATLAB (Simulink).

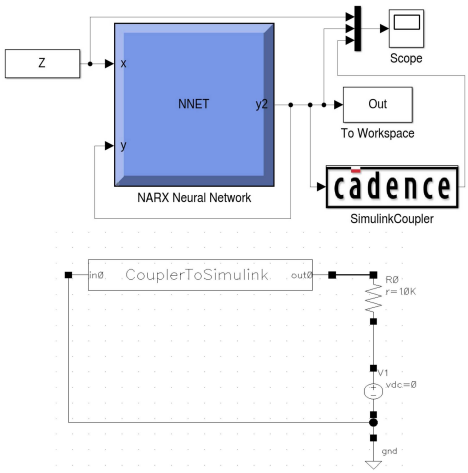

Fig. 8. Co-simulation interface between Simulink (top) and Cadence AMS Designer (bottom). The coupler provides the link between two environments.

Figure 8 shows the co-simulation environments in Simulink and in Cadence Virtuoso respectively, where the coupler module links two environments.

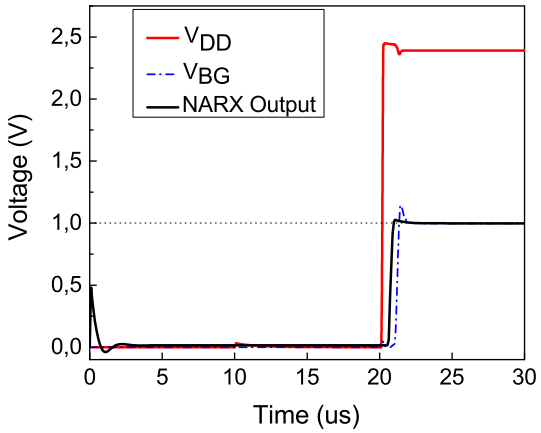

Fig. 9. BGR circuit and NARX network output responses to the same input signal (V_{DD}) simulated using co-simulation interface. Note that the response of the neural network is slightly faster than the BGR circuit yet it strictly follows the 1V signal.

We run a transient simulation using AMS simulator in ADE while applying the input to the NARX network in Simulink. Figure 9 represents the results of the transient simulation. We achieved a remarkable improvements in simulation time, where the total CPU time measured as 0.4s while for the same simulation with the BGR circuit itself the CPU time was 6.28s. This implies an enhancement with a factor of 16.

VI. Conclusion

We investigated a novel technique for automatically learning an abstract model of a complex analog IC, the CMOS BGR, which consisted of generating specific learning, validation and test time series by simulating the analog IC in Cadence, and then by employing back-propagation techniques to learn a neural NARX model, reproducing the behavior of the original IC up to a desired accuracy, during its power up. By utilizing this model, we achieved a remarkable simulation-time improvement for transient behavior. Our next step is to make use of our NARX neural network model into analog fault simulations where we test a part of the circuit in transistor level and model the rest of the circuit with NARX model.

Acknowledgment

Acknowledgment to Infineon for training, mentoring and providing the tool landscape. Authors would also like to thank Mr. Florian Korus from Infineon Munich for his kind support regarding co-simulation initial setups.

References

[1] B. Kruseman, "Testing of analog/mixed signal ics: Past, present and future," in *Test Symposium (ETS), 2015 20th IEEE European*. IEEE, 2015, pp. 1–1.

[2] M. Soma, "Analog fault models: Back to the future?" in *Test Conference (ITC), 2014 IEEE International*. IEEE, 2014, pp. 1–1.

[3] M. Magerl, C. Stockreiter, O. Eisenberger, R. Minixhofer, and A. Baric, "Building interchangeable black-box models of integrated circuits for emc simulations," in *Electromagnetic Compatibility of Integrated Circuits (EMC Compo), 2015 10th International Workshop on the*. IEEE, 2015, pp. 258–263.

[4] H. Demuth, M. Beale, and M. Hagan, "Neural network toolbox 8.4," *Users guide*, 2015.

[5] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[6] Cadence. Cadence Virtuoso AMS Designer Simulator, cosimulation of mixed-signal systems with matlab and simulink. [Online]. Available: http://www.mathworks.com/products/connections/product_detail/product_35807.html?refresh=true