Verilog-A Compatible Recurrent Neural Network Model for Transient Circuit Simulation

Zaichen Chen
Dept. of Electrical and Computer Eng.
Univ. of Illinois at Urbana-Champaign
Urbana, Illinois
zchen19@illinois.edu

Maxim Raginsky
Dept. of Electrical and Computer Eng.
Univ. of Illinois at Urbana-Champaign
Urbana, Illinois
maxim@illinois.edu

Elyse Rosenbaum
Dept. of Electrical and Computer Eng.
Univ. of Illinois at Urbana-Champaign
Urbana, Illinois
elyse@illinois.edu

***Abstract*—This paper presents a method for data-driven behavioral modeling of electronic circuits using recurrent neural networks (RNNs). The RNN structure is adapted based on known characteristics of the system being modeled. The discrete-time RNN is transformed to a continuous-time model and then implemented in Verilog-A for compatibility with general-purpose circuit simulators.**

***Index Terms*—Behavioral models; Circuit simulation; Recurrent neural network; Verilog-A**

I. Introduction

Behavioral models are widely used in lieu of physics-based models to represent electronic circuits, in order to protect intellectual property, reduce model development cost, and improve computational efficiency. Among them, recurrent neural networks (RNNs) were demonstrated to provide good accuracy when modeling nonlinear circuits [1]–[3]. However, in previous works, the RNN model was evaluated using fixed time-step input waveforms that are independent variables. In contrast, a general-purpose circuit simulator performs modified nodal analysis (MNA); a *variable* time-step is generally used, and each model's inputs are the *dependent* variables solved for at each time step. This work demonstrates an RNN implementation compatible with MNA, and the fidelity of the model is evaluated using a general-purpose circuit simulator.

This paper is organized as follows. Domain-specific RNN equations are introduced in Section II. In Section III, a Verilog-A implementation of the RNN model is developed for use with general-purpose circuit simulators. In Section IV, circuit simulations using both RNN models and physics-based compact models are presented to benchmark the fidelity of the modeling methodology.

II. Modeling approach

The RNN is a type of neural network used for modeling of dynamic systems. Unlike the feedforward neural network whose inputs and outputs are all static variables, the inputs and outputs of an RNN are time series. Figure 1 shows the structure of an RNN with a single hidden layer. At each time-step, the hidden state vector $\boldsymbol{x}_i$ is updated recurrently based on the input to the RNN $\boldsymbol{u}_i$ and the value of hidden state vector at the previous time-step $\boldsymbol{x}_{i-1}$.

In this work, the structure of the RNN is adapted to a key property of the circuit being modeled. Specifically, thermodynamic equilibrium requires that electronic devices have the following property: zero DC current when all terminals are biased at the same potential. This property will be referred to as zero-in zero-out (ZIZO). The following RNN equations are introduced to account for the ZIZO property:

$$\boldsymbol{x}_i = \boldsymbol{\sigma}[W_r \boldsymbol{x}_{i-1} + W_u \boldsymbol{u}_i + \boldsymbol{\sigma}^{-1}(\boldsymbol{\Omega_0})] - \boldsymbol{\Omega_0} \tag{II.1a}$$

Funding for this research was provided in part by the National Science Foundation under the CAEML I/UCRC award no. CNS 16-24811.

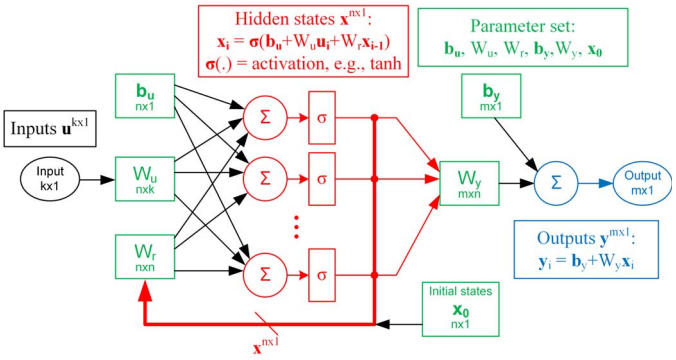

Fig. 1: Structure and model equations of a 1-hidden-layer RNN with k inputs, m outputs, and n hidden states

$$\boldsymbol{y}_i = W_y \boldsymbol{x}_i \tag{II.1b}$$

Note that simpler ZIZO formulations, e.g., setting all the bias terms to zero, will fail to model a circuit whose I-V characteristic is not an odd function.

In Equation (II.1), $\boldsymbol{u}_i \in \mathbb{R}^k$, $\boldsymbol{x}_i \in \mathbb{R}^n$, and $\boldsymbol{y}_i \in \mathbb{R}^m$ are vectors representing the inputs, hidden states, and outputs of the RNN at time step i, respectively. The model parameters are $W_u \in \mathbb{R}^{n\times k}$, $W_r \in \mathbb{R}^{n\times n}$, $\boldsymbol{\Omega_0} \in \mathbb{R}^n$, and $W_y \in \mathbb{R}^{m\times n}$. $\boldsymbol{\sigma}(\bullet) \in \mathbb{R}^n$ is the invertible activation function of the first layer of the RNN, where the same scalar function is applied to each element of its argument vector. Common choices for the scalar function include the tanh function and the sigmoid function. The RNN input and output time series correspond to the terminal voltages and currents of the device being modeled. For an N-terminal circuit, due to KCL and KVL, there can be only $N-1$ independent voltage and current variables; therefore, an RNN with $(N-1)$-dimensional input and output will be used as its model. In practice, since most circuit simulators use modified nodal analysis (MNA), the terminal voltages are adopted as the model inputs, and currents as the outputs. The dimension of the hidden states vector $\boldsymbol{x}$ controls the complexity of the model, and is arbitrarily chosen to be 20 in this work.

The ZIZO-RNN (II.1) is trained using the stochastic gradient descent algorithm. The open-source Python libraries Keras and Theano are used in the training program.

III. Compact model implementation

A. Verilog-A implementation of RNN

The ZIZO-RNN model introduced in Section II is implemented in Verilog-A for use with a general-purpose circuit simulator. First, the RNN is converted from a collection of finite difference equations to

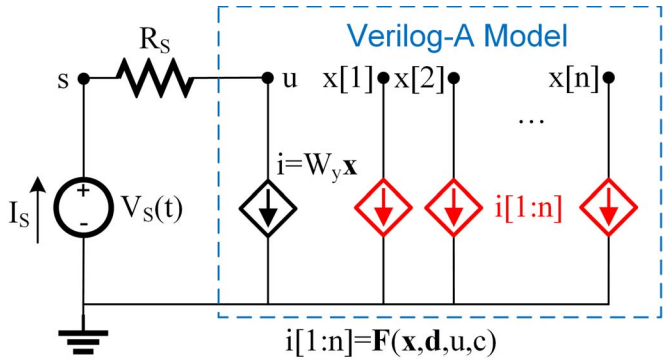

Fig. 2: Schematic representation of Verilog-A model of a two-terminal circuit driven by a voltage source with a series resistance

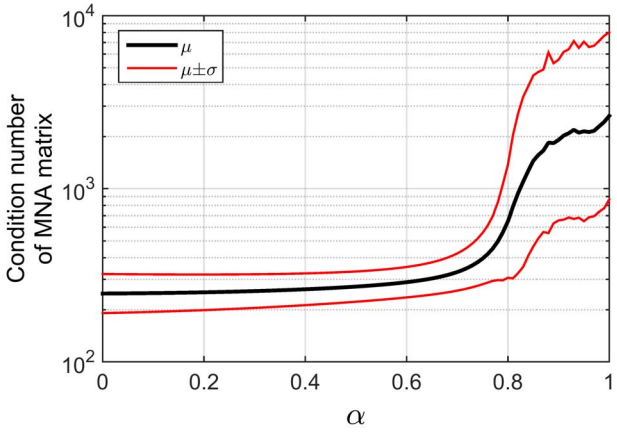

Fig. 3: MNA matrix condition number from Monte-Carlo simulation with random M_O

a collection of differential equations; this will allow the circuit to be simulated using a different time-step from that used for training—or a variable time-step. Then, the differential equations are implemented as a Verilog-A model.

To convert the RNN equations to differential form, a first-order expansion of the continuous-time inputs and hidden states is taken at $t = t_{i-1} + \alpha h$. Here, $0 < \alpha < 1$ is a model parameter and $h > 0$ is the RNN time-step. The discrete-time RNN inputs and hidden states may then be approximated as:

$$\boldsymbol{u}_i = \boldsymbol{u} + (1-\alpha)h\dot{\boldsymbol{u}} \tag{III.1a}$$
$$\boldsymbol{x}_{i-1} = \boldsymbol{x} - \alpha h\dot{\boldsymbol{x}} \tag{III.1b}$$
$$\boldsymbol{x}_i = \boldsymbol{x} + (1-\alpha)h\dot{\boldsymbol{x}} \tag{III.1c}$$

Substituting (III.1) into the recurrent relationship (II.1a) yields

$$\begin{aligned}&\boldsymbol{x} + (1-\alpha)h\dot{\boldsymbol{x}}\\ &= \boldsymbol{\sigma}[W_r(\boldsymbol{x} - \alpha h\dot{\boldsymbol{x}}) + W_u(\boldsymbol{u} + (1-\alpha)h\dot{\boldsymbol{u}}) + \boldsymbol{\sigma}^{-1}(\boldsymbol{\Omega_0})] - \boldsymbol{\Omega_0}\end{aligned} \tag{III.2}$$

The continuous-time output is given by

$$\boldsymbol{y} = W_y\boldsymbol{x} \tag{III.3}$$

The Verilog-A implementation of (III.2) and (III.3) is illustrated in Figure 2. In the Verilog-A model, the hidden states $\boldsymbol{x}$ are represented by node voltages. At each node, a controlled current source is connected to ground. Each current source implements the function

$$\begin{aligned}i[1:n] = \boldsymbol{F}(\boldsymbol{x},\boldsymbol{d},u,c) &= \boldsymbol{x} - (1-\alpha)\boldsymbol{d}\\ &- \boldsymbol{\sigma}[W_r(\boldsymbol{x} - \alpha\boldsymbol{d}) + W_u(u - (1-\alpha)c) + \boldsymbol{\sigma}^{-1}(\boldsymbol{\Omega_0})] + \boldsymbol{\Omega_0}\end{aligned} \tag{III.4}$$

Note that $\boldsymbol{F}$ is the difference between the LHS and RHS of (III.2). By construction of the circuit in Figure 2, the current through each of the controlled sources must be zero. Therefore, when the circuit simulator attempts to solve the KCL equation for each node, it will find the solution to (III.2).

In (III.4), $\boldsymbol{d} = h\dot{\boldsymbol{x}}$ and $c = h\dot{u}$ are implemented using the `ddt` function in Verilog-A. Since a two-terminal device is presented in the example, the input is one-dimensional, and thus u and c are scalars. However, (III.4) also applies to a multi-dimensional input.

B. A numerical issue

It is observed that the value of the model parameter α affects the numerical stability of the model, i.e. the convergence of the numerical simulation. The numerical stability deteriorates when α is close to 1, as demonstrated by an analysis of the circuit shown in Figure 2. Assuming that the backward Euler method with Newton-Raphson iteration is used for the nonlinear transient simulation, the following set of linear equations will be solved during each iteration at each time-step:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & O & O \\ -1 & \frac{1}{R_s} & \frac{-1}{R_s} & 0 & O & O \\ 0 & \frac{-1}{R_s} & \frac{1}{R_s} & 0 & W_y & O \\ 0 & 0 & h/\tau & -1 & O & O \\ O & O & J_u^* & J_c^* & J_x^* & J_d^* \\ O & O & O & O & hI/\tau & -I \end{bmatrix} \begin{bmatrix} i_s \\ s \\ u \\ c \\ \boldsymbol{x} \\ \boldsymbol{d} \end{bmatrix} = \begin{bmatrix} v_s \\ 0 \\ 0 \\ hu_{\text{prev}}/\tau \\ -\boldsymbol{F}(\boldsymbol{X}^*) + J_F^*\boldsymbol{X}^* \\ h\boldsymbol{x}_{\text{prev}}/\tau \end{bmatrix} \tag{III.5}$$

Above, $\boldsymbol{x}_{\text{prev}}$ and u_{prev} are the solutions of $\boldsymbol{x}$ and u at the previous time-step, and τ is the variable time-step used in the transient simulation; $\boldsymbol{X} = (\boldsymbol{x}, \boldsymbol{d}, u, c)$ and $\boldsymbol{X}^*$ is the value of $\boldsymbol{X}$ in the previous iteration; J_F^* is the Jacobian of $\boldsymbol{F}(\boldsymbol{X})$ when $\boldsymbol{X} = \boldsymbol{X}^*$. From (III.4), the Jacobians are found to be

$$J_u = \partial\boldsymbol{F}/\partial u = -M_O W_u \tag{III.6a}$$
$$J_c = \partial\boldsymbol{F}/\partial c = -(1-\alpha)M_O W_u \tag{III.6b}$$
$$J_x = \partial\boldsymbol{F}/\partial\boldsymbol{x} = I - M_O W_r \tag{III.6c}$$
$$J_d = \partial\boldsymbol{F}/\partial\boldsymbol{d} = (1-\alpha)I - \alpha M_O W_r \tag{III.6d}$$

In (III.6), I is the identity matrix and M_O is the diagonal matrix given by

$$M_O = \text{diag}\left\{\boldsymbol{\sigma}'[W_r(\boldsymbol{x}^* - \alpha\boldsymbol{d}^*) + W_u(u^* + (1-\alpha)c^*)]\right\}, \tag{III.7}$$

where $\boldsymbol{\sigma}'(\bullet)$ denotes the element-wise derivative of $\boldsymbol{\sigma}(\bullet)$.

The entries of matrix M_O depend on the values of intermediate variables during Newton-Raphson iterations and cannot be known a priori. However, by treating the elements of M_O as random variables and using Monte-Carlo simulation, the numerical stability of the transient simulation for a specific time-step τ can be evaluated. In fact, a large set of randomly generated M_O represents a wide range of MNA matrices being solved in actual circuit simulations. Taking the transient simulation time-step τ to be 1/10 of the RNN time-step h, the Monte-Carlo simulation is done for an RNN trained with the data described in Section IV-A. The relationship between the condition number of the MNA matrix and α is plotted in Figure 3; the results indicate that the condition number of the MNA matrix increases dramatically when α gets close to one.

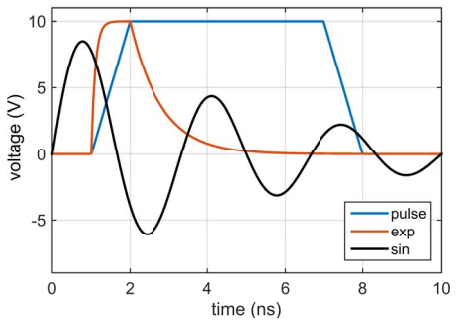

Fig. 4: Stimuli used for training data generation

IV. Model Evaluation

The proposed modeling approach is evaluated by training the RNN to replicate the behavior of a small nonlinear circuit, specifically, an active rail clamp circuit, which consists of 6 transistors and 2 passive elements and is implemented in 130 nm CMOS [4]. Training data are obtained from circuit simulation using physics-based compact models. It is intended that this modeling methodology will also be applied to training data obtained from lab measurements. To establish that this will be possible, training data are generated using netlists that emulate a measurement setup in which the voltage sources have finite output resistance, rather than forcing a known waveform at the external terminal(s) of the circuit. The RNN training data consist of the time-varying voltages and currents at the device terminals, extracted from the simulation results. During model evaluation, the RNN model is implemented in Verilog-A, and then circuit simulations are performed using previously unseen stimuli that are more complicated than those used to obtain the training samples. Since the circuit simulator treats both the device voltage and current as dependent variables to be solved for, both waveforms need to be evaluated.

A. Training data

Figure 4 shows the stimuli chosen for the training samples. The waveforms shown in the figure were obtained for the case that the device being simulated is an open circuit. The frequency-content and amplitude of the stimuli are varied to generate a set of training data that cover all the operating states of the device. The variable time-step training samples obtained from circuit simulation are interpolated to a fixed time-step $h = 50\,\mathrm{ps}$ for RNN training, resulting in time sequences that have 201 steps. All circuit simulations are performed using Cadence Spectre.

B. Verilog-A model simulation results

For model evaluation, two kinds of stimuli are chosen. The first stimulus is a piecewise-linear voltage source with a $1\,\Omega$ output resistance. The amplitude and rise/fall times of the voltage source are randomly generated. The second stimulus is the waveform generated by an IEC 61000-4-2 ESD tester. Since both stimuli have a longer time-span than the training sequences, if the model accurately predicts the response, it will be claimed to replicate the device dynamics well.

Figure 5 and Figure 6 show the results obtained when circuit simulation is performed using (i) the trained RNN Verilog-A model and (ii) a transistor-level netlist to which the BSIM4v4 transistor model is applied.

Any difference between the simulated results obtained using the two models is attributed to the inaccuracy of the RNN model. The root-mean-squared error is expressed as a percentage of the peak-to-peak amplitude and the values are listed in Table I. In all cases, the RNN model has an error of less than 2 percent, indicating that the RNN model accurately predicts the transient response.

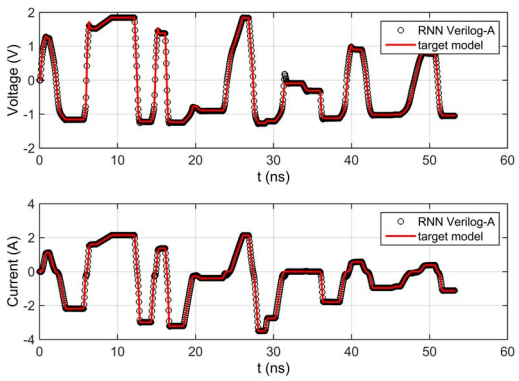

Fig. 5: Simulation result for rail clamp with PWL stimulus

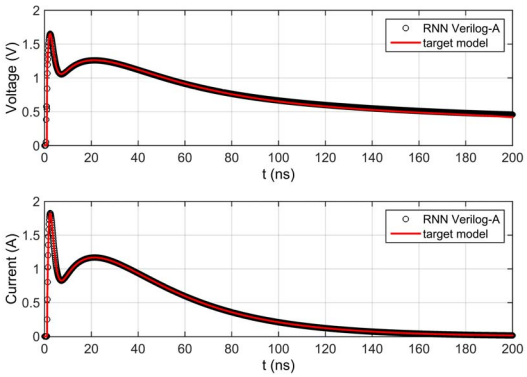

Fig. 6: Simulation result for rail clamp with IEC stimulus

TABLE I: Error in transient simulations

Stimuli	Voltage Error (RMSE/p-p)	Current Error (RMSE/p-p)
PWL	1.39%	0.77%
IEC	1.91%	0.20%

V. Conclusions

A RNN model which accounts for the ZIZO property of electronic circuits is introduced in this paper. The RNN model is implemented in Verilog-A in a form suitable for MNA simulations. The potential numerical instability of the model is mitigated by appropriate selection of the time instance within the RNN time-step at which the differential equation is formulated. It is demonstrated that the Verilog-A RNN model shows good accuracy in transient simulations, even when the stimuli are considerably different from the training data.

References

[1] D. Luongvinh and Y. Kwon, *IEEE MTT-S International Microwave Symposium Digest*, 2005.

[2] Y. Cao and Q.-J. Zhang, *IEEE Trans. Microwave Theory and Techniques*, Jun. 2009.

[3] Y. Fang, M. C. E. Yagoub, F. Wang and Q.-J. Zhang, *IEEE Trans. Microwave Theory and Techniques*, Dec. 2000.

[4] R. Mertens, N. Thomson, Y. Xiu and E. Rosenbaum, *EOS/ESD Symp.*, 2014.