Behavioral Modeling of Analog Neural Networks Inference Circuits

Yasmine Abu-Haeyeh, Sascha Schmalhofer, Lars Hedrich
Institute for Computer Science, Goethe University Frankfurt, Germany
abu-haeyeh/schmalhofer/hedrich@em.cs.uni-frankfurt.de

***Abstract*—Analog neural networks provide an energy- and area-efficient hardware solution for a wide range of edge AI applications. However, verification of such analog-based neural networks suffers from high simulation and debugging costs when simulated at the transistor level. This paper proposes an automated behavioral modeling approach for analog inference circuits. We utilize a nonlinear model order reduction technique to abstract a single neuron netlist which is the building block of large analog neural networks. Then we generate a precise Verilog-A behavioral model that maintains the neuron circuit's essential nonlinear static and dynamic behavior. Finally, we automatically reconstruct the neural network using a modular set combining thousands of these behavioral models, allowing for debugging the basic functionality and judging the network's speed. Our experimental results show an average speedup of 234 when simulating the generated network compared to the transistor-level netlist on three examples including MNIST.**

***Index Terms*—Formal Verification, Analog Neural Networks, Verilog-A Model.**

I. Introduction

Artificial Neural Networks are vital tools in a wide range of applications, such as data analysis, image recognition, and medical diagnosis. Software implementations of neural networks require an extensive computational cost and power consumption due to the increased dimensionality and complexity. Hence, hardware implementation has been employed in the inference of neural networks, either digital- or analog-based [1]. Analog-based inference hardware presents an efficient solution in terms of energy and area utilization. However, the design of these analog inference circuits is more complicated due to its vulnerability to inaccuracies such as process variations, noise, and nonlinearities of the activation functions. To model and verify the functionality of these circuits, extensive simulations on large complex netlists are required for verification [2]. Since the structure of deep neural networks exhibits thousands of repeated instances of a neuron circuit, a well-abstracted analog neuron model can speed up simulation routines and facilitate verification. However, due to the increasing size of neural networks, this is only feasible if the abstraction process is fully automated.

A. Related Work

Multiple designs have been proposed for the implementation of neural network inference circuits, where the training of the network happens off-chip while the inference is performed by the circuit. An analog perceptron was proposed in [3] with an improved source follower circuit for the ReLU activation function. However, the DAC-based multiplier consumes higher power while introducing quantization errors proportional to the input value range. In [4], a hardware implementation for a CMOS-compatible neuron circuit with three different activation functions is presented and evaluated on the MNIST dataset. Their circuit design incorporates many resistors and operational amplifiers, which affects the hardware's efficiency in terms of power and area. The analog inference circuits are designed to perform different functions such as multiplication, addition, Rectified Linear Unit (ReLU), and convolution. Where another simple circuit design was proposed in [5] for the multiply and accumulate operations along with the ReLU function. The proposed neuron with two inputs and one ReLU output outperforms other digital multipliers in terms of power efficiency while maintaining similar accuracy. However, the design has not been tested on a real example of a classification or regression problem. An overview of the recent advances in accelerator designs for neural networks can be found in [6], showing a map of comparison between the accelerators based on their power consumption and speed.

All mentioned implementations for analog hardware still require thorough validation with endless simulations to verify their functionality. Behavioral modeling can accelerate the verification process while preserving the main circuit properties, such as nonlinearities and dynamic behavior. An automated abstraction technique was proposed in [7] for transistor-level circuits, transforming the circuit netlist into a Verilog-A behavioral model with SPICE accuracy, which resulted in simulation speedup. Based on this, we propose in this paper an automated approach to generate a Verilog-A model from a neuron circuit to facilitate efficient formal verification of large analog neural networks.

B. Contributions

In this work, we present a fully automated approach to model CMOS inference circuits to replace time-consuming transistor-level simulations. Our main contributions can be summarized as follows:

- Automatic abstraction of the analog neuron, starting from the transistor-level netlist.
- Behavioral modeling that preserves the fundamental nonlinear static and dynamic behavior of the neuron circuit.
- A modular set of behavioral models that replaces all transistor-level circuitry, enabling a high speedup.
- Evaluation of our proposed approach on three different classification problems, including image recognition on the MNIST dataset.

II. Preliminaries

A. Hybrid Automata

Definition 1 (Hybrid Automata [8, 9]). *A hybrid automaton* $HA = (Loc, loc_0, X, X_0, U, inv, tran, grd, J, f)$ *contains:*

- *A finite set of locations* $Loc = loc_1, ..., loc_v$ *with an initial location* loc_0
- *A set of continuous state variables* $X \subseteq \mathbb{R}^n$*, and an initial set* $X_0 \subseteq X$
- *A continuous input space* $U \subseteq \mathbb{R}^{n_i}$
- *An invariant set* $inv : Loc \to 2^X$ *which assigns an invariant* $inv(loc) \subseteq X$ *to each location* $loc \in Loc$
- *A set of discrete transitions* $tran \subseteq Loc \times Loc$
- *A guard set* $grd : tran \to 2^X$ *that assigns a guard set* $grd(loc_i, loc_j)$ *for each transition from* loc_i *to* loc_j
- *A jump function* $J : tran \times X \to X$ *, which returns the next continuous state when a transition is taken*
- *A flow function* $f : Loc \times X \times U \to \mathbb{R}^n$*, which defines a continuous vector field for the time derivative of* X*:* $\dot{X} = f(loc, X, U)$.

B. Neural Networks

We employ a feed-forward neural network with two different types of models in this paper:

Definition 2 (ReLU Neural Network). *Given an input* $I \in \mathbb{R}^{m_0}$ *and an output* $Y \in \mathbb{R}^{m_d}$*, a ReLU Neural Network with* $d \in \mathbb{N}$ *layers can be defined consisting of dense layers as:*

$$h_0 = I,$$
$$h_k = L_k(h_{k-1}) = ReLU(W_k h_{k-1} + B_k), \quad W_k \in \mathbb{R}^{m_k \times m_{k-1}},$$
$$Y = h_d,$$

with $L_k : \mathbb{R}^{m_{k-1}} \to \mathbb{R}^{m_k}$*, weights* W_k*, and biases* $B_k \in \mathbb{R}^{m_k}$*, layer* $k \in [1, ..., d]$*. The convolution model can be defined analogously while adapting the input, weights, and bias sizes accordingly.*

III. Implementation of the Analog Neural Network

To thoroughly explain our behavioral modeling approach, we need first to present our 130 nm, 1.2 V CMOS inference circuit [10]. The circuit operates in the sub-threshold, providing a low-power solution with all the circuit's currents in the nA range.

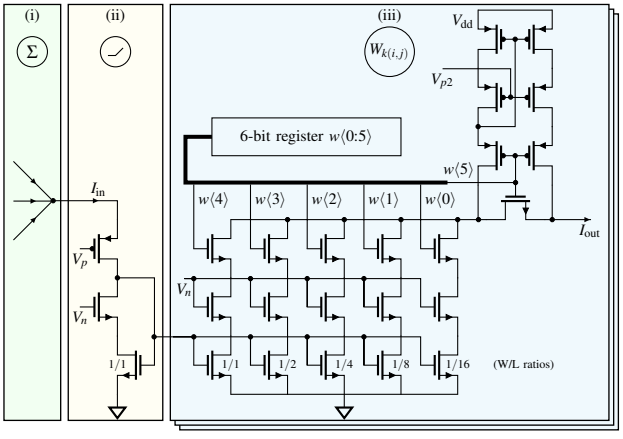

Fig. 1: Circuit schematic of the analog neuron circuit: (i) Summation block based on Kirchhoff's law, (ii) ReLU activation function, and (iii) weight cells. The programming bits are stored in a 6-bit register. The ReLU and weight blocks establish a binary weighted ($w\langle 0{:}4\rangle$) current mirror.

The main building block in our circuit design is the analog neuron, which emulates the neural network's linear layer combined with ReLU activation. The schematic of the analog neuron circuit is shown in Fig. 1 split into three parts based on their functionality. A summation point (i), where the weighted input currents are summed with the bias current by employing the circuit's Kirchhoff laws. Followed by the ReLU activation in (ii), where the left part of the current mirror is utilized to rectify negative currents. Finally, the right part of the current mirror in (iii) scales the output current using a 6-bit programmable weight. $w < 0:5 >$ (check W/L ratios in Fig. 1 (iii)). The sign of the weight is assigned by the sixth bit $w < 5 >$ by inverting the current using another current mirror. An additional bias circuit not shown produces an analog current mapping an 8-bit programmable bias.

The training of the neural network is performed off-chip using TensorFlow and quantized to six bits, producing a total of 64 different weights in the range of [-2, 2]. The neural network input signals are scaled to the range of [-200 nA, 200 nA], where a 1 in TensorFlow resembles a current of 50 nA in the inference circuit. Implementing the neural network defined in Def. 2 using the combined ReLU and weight circuit from Fig. 1 demands applying some changes to the input and output of the network. At the input, we integrate a slightly different neuron circuit to comply with the conventional neural network functionality for negative input signals. An additional layer with a weight of 1 is needed at the output of the network to execute the final ReLU activation.

IV. Modeling of the Neuron Circuit

To effectively model the behavior of the full neural network, we propose the following approach: First, we abstract a single neuron circuit into a Verilog-A behavioral model. Then, the generated models are combined following the network architecture layer by layer to replace the original circuit, achieving a speedup in simulation and verification.

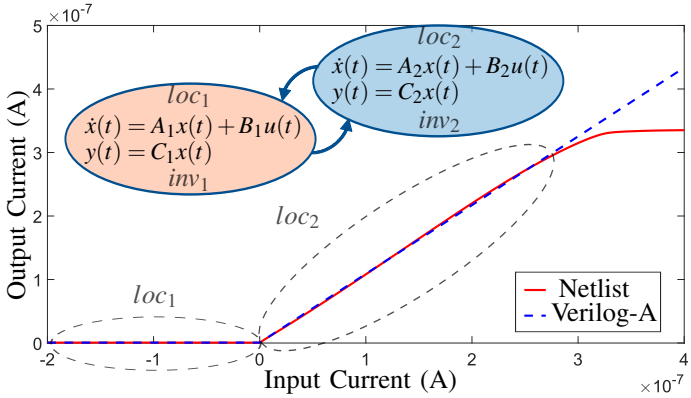

Fig. 2: DC transfer curve of the ReLU-neuron circuit from Fig. 1 with a weight of 1. The left part is ideal 0, while the right part is up to 200 nA, almost linear, but reaches saturation after 300 nA. The neuron's generated HA is shown on top, presenting the system description of each location.

The dynamic behavior of a single neuron in our inference circuit exhibits 33 poles, which require Model Order Reduction (MOR) to reduce the model's complexity. The nonlinear behavior of the ReLU function is more challenging since it requires a nonlinear MOR approach to reduce the circuit order. We propose to use a reduction to a hybrid automaton as defined

in Def. 1 consisting of locations with approximated linear dynamic behavior, where multiple locations can approximate the strongly nonlinear behavior [7].

Starting from the neuron netlist displayed in Fig. 1, we compute the nonlinear equations using modified nodal analysis. Then we apply state space sampling to linearize the system around sampling points. Next, we cluster the sampled points using K-means into multiple locations based on the system eigenvalues. The first location loc_1 in the generated hybrid automata represents the limiting region when the input is negative, while loc_2 corresponds to the linear region as seen in Fig. 2. We iterate over the sampled points in each location and apply Padé-via-Lanczos (PVL) MOR on each point's conductance and capacitance matrices [11]. The generated reduced-order differential equation of each location of the HA is shown in Fig. 2, where $X \in \mathbb{R}^q$ is the vector of q state variables in the reduced system. While $A \in \mathbb{R}^{q \times q}$ is the reduced system matrix, B, and C are the input and output vectors, respectively, with a system input u and output y. To choose the order of reduction q, we compared the approximation error in the frequency response for different order values from 1 to 4, the resulting errors were $12.6\%, 2\%, 5\%, 1.7\%$ respectively. Hence, we chose $q = 2$ which achieves a very low approximation error with maximum computational efficiency compared to $q = 4$ which imposes a much higher computational cost. For each location of the HA, an invariant with a set of guards is defined to control the transition between the available locations following Def. 1. Finally, we generate a Verilog-A model file containing the neuron system description represented in an HA of two locations, which will be utilized to gain simulation speedup. Additionally, we can run a reachability analysis to verify the abstracted model behavior using the CORA tool [9] on the generated HA. We generate 64 distinct HAs, one for each weight value, and combine them in one Verilog-A model since they all share the same constraints for switching between the two different locations. However, we generate another Verilog-A model for the input neurons that exhibit linear behavior without the ReLU function of only one location. Finally, the complete neural network is constructed from the generated behavioral models using the Cadence SKILL scripting language.

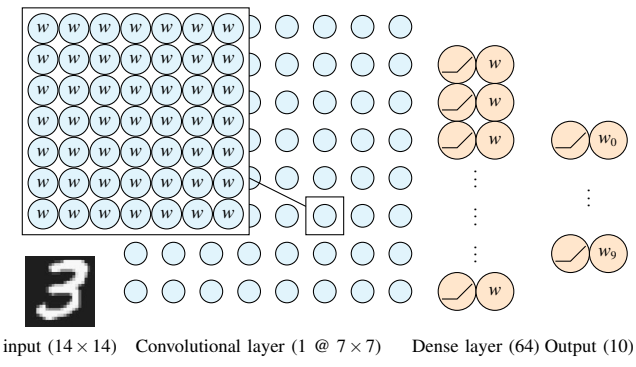

Fig. 3: The generated neural network of the MNIST example, where blue neurons resemble the Verilog-A model of the input neuron model without ReLU, and orange neurons are the ReLU neurons.

The MNIST-generated network follows the underlying transistor-level network structure, where the 2D convolution kernel is duplicated 8×8 times as shown in Fig. 3. Each kernel contains 7×7 input neurons followed by one bias producing a feature map, which is flattened in the next layer. However, the generated network has a feedback-free structure where any change on the proceeding blocks doesn't affect the previous ones, achieving a speedup compared to the netlist.

V. Experimental Results

Generating the behavioral model process is performed in the MATLAB environment along with the Vera state space sampling tool implemented in C++. We evaluate the generated Verilog-A models to compare the performance with the nominal simulation of the inference circuit directly in the Cadence Spectre simulation platform. Our proposed approach is evaluated on the circles, Iris, and the well-known benchmark MNIST dataset [12].

For the circles dataset provided by the scikit-learn Python library, we implemented a network of a fully connected layer of 20 hidden neurons and an output layer of one neuron for binary classification. The overall test accuracy of the inference circuit for a total of 100 input samples is 93%.

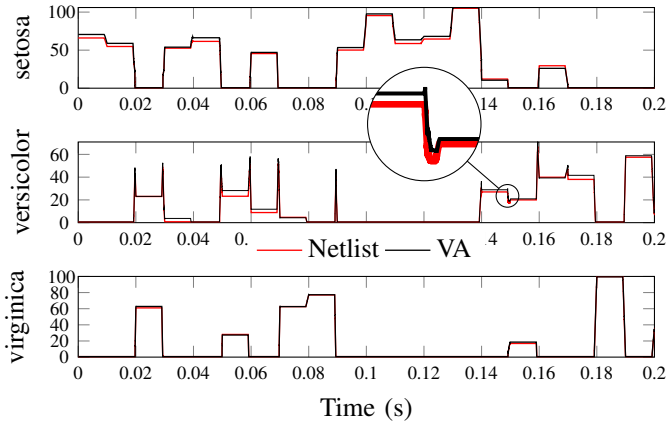

Fig. 4: Iris' inference circuit output current in (nA) of the Verilog-A generated network against the original netlist simulation for each class.

We also evaluated our approach on the Iris dataset provided by the same Python library using a network of 4 inputs, a fully connected layer of 8 hidden neurons, and an output layer of 3 neurons representing the three classes. The network achieves an accuracy of 92% on the test set consisting of 75 input patterns. As noticed in the simulation results in Fig. 4, the generated Verilog-A network matches the netlist behavior with similar spikes produced from switching the input values.

The MNIST network architecture shown in Fig. 3 has a reduced input size of (14×14) to minimize the inference circuit complexity and power consumption while maintaining a high test accuracy of 92%. Since MNIST is a multiclass classification problem, the inference circuit produces 10 different outputs from y_0 to y_9, where the maximum output resembles the predicted label. As shown in the confusion matrix in Fig. 5, the generated Verilog-A network achieves similar prediction results compared to the circuit netlist with a minor error. A total of 2 patterns out of 100 were classified differently; the inference circuit misclassified a pattern as 9 while it was correctly classified as 4 using the Verilog-A network. This error was caused by the netlist producing very close output currents for both y_9 and y_4 with only a 0.2nA difference, which led to a misprediction from the netlist. The second

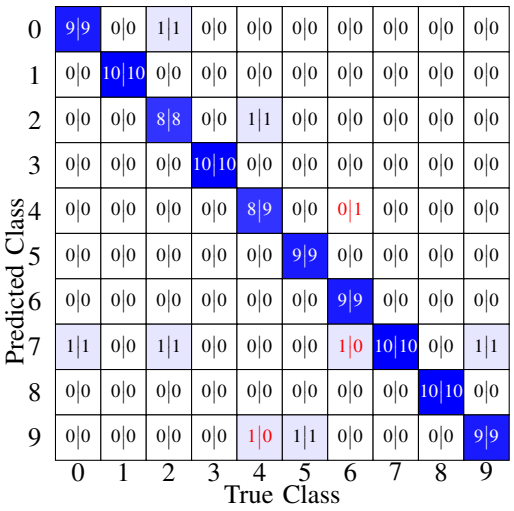

Fig. 5: Confusion matrix for MNIST: For comparison, the netlist prediction result per class is reported on the right, and the Verilog-A predictions on the left (netlist | Verilog-A), discrepancy in prediction is colored in red.

pattern (6) was misclassified from both the netlist and the Verilog-A network, which is again caused by close values of different outputs. It can be rectified by optimizing the netlist to increase the classification accuracy or designing a deeper network architecture to enhance the prediction accuracy, which comes with the cost of extra power consumption and chip area.

TABLE I: Performance comparison between the transistor-level circuit and the Verilog-A generated neural network

Dataset		*Size**	Runtime	Speedup	M_a**	δ_a	δ_d
Iris	Netlist	3,814	28.67	47	97	3.1	2.4
	VerilogA	59	**0.608**				
Circles	Netlist	6,552	473.3	434	96	4.7	4.3
	VerilogA	61	**1.090**				
MNIST	Netlist	94,420	6,084	220	98	2.0	1.9
	VerilogA	3,786	**27.65**				
Avg.				234	97	3.3	2.9

*Size** refers to the number of transistors in the netlist and number of neurons in the Verilog-A generated network, M_a** is the matching accuracy. δ_a and δ_d are the approximation and dynamic errors respectively.

In Tab. I, we report the results of simulating the circles and MNIST inference circuits and their Verilog-A networks, testing a total of 100 samples and 75 samples for Iris. The runtime is measured as the elapsed total time required for transient analysis in Cadence, where the three examples' Verilog-A networks showed a significant reduction in runtime compared to the original netlist. The three examples provide an average speedup of 234, where the circles network achieved a maximum speedup of 434 while the Iris speedup was 47 due to its simple architecture of fewer neurons. The matching accuracy is reported to analyze the performance of our generated network, where the circles network showed a mismatch in classification results for a total of 4 patterns out of 100. On the other hand, the MNIST achieved the highest matching accuracy of 98%.

To precisely assess the performance of the generated behavioral models, two different types of errors are calculated and reported in Tab. I. δ_a is the approximation error calculated as the absolute difference between both models' output scaled back to TensorFlow values, then averaged over the number of test samples. The dynamic error δ_d computes the area between the two curves to judge the precise dynamical behavior of our Verilog-A model which is crucial for larger networks with thousands of neurons. The MNIST expressed the lowest approximation and dynamic errors of 2.0% and 1.9%, respectively, measured as the average of the ten outputs' errors. As noticed from the results table, our behavioral model matches the inference netlist's behavior with an average of 3.3% approximation and 2.9% dynamic errors for three different network architectures.

VI. Conclusion

This paper presents an automated behavioral modeling approach for large analog neural networks. It incorporates nonlinear model order reduction with a reduction to linearized HA to generate a Verilog-A model, which is capable of efficiently replacing the analog neuron netlist. The generated Verilog-A models are combined to reconstruct the neural network, preserving dynamics and nonlinearities. The accuracy and performance of the proposed approach were successfully evaluated with three examples ranging from 59 to 3,786 neurons or up to 94,420 transistors, achieving high speedup range from 47 to 434.

Acknowledgment

The authors gratefully acknowledge financial support by the project FAI (No. 286525601) funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG).

References

[1] M. Forssell, "Hardware implementation of artificial neural networks," *Information flow in networks*, vol. 18, no. 2014, pp. 1–4, 2014.

[2] G. Gielen, N. Xama, K. Ganesan, and S. Mitra, "Review of methodologies for pre-and post-silicon analog verification in mixed-signal socs," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1006–1009.

[3] C. Geng, Q. Sun, and S. Nakatake, "An analog cmos implementation for multi-layer perceptron with relu activation," in *2020 9th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, 2020, pp. 1–6.

[4] Z. Chen, A. Farjudian, and J. Greer, "Cmos-enabled activation functions for low-power hardware neural networks," *Authorea Preprints*, 2023.

[5] Y. Huang, Z. Yang, J. Zhu, and T. T. Ye, "Analog circuit implementation of neurons with multiply-accumulate and relu functions," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, 2020, pp. 493–498.

[6] K. Guo, W. Li, K. Zhong, Z. Zhu, S. Zeng, S. Han, and Y. Xie. "neural network accelerator comparison". [Online]. Available: https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator.html

[7] A. Tarraf and L. Hedrich, "Behavioral modeling of transistor-level circuits using automatic abstraction to hybrid automata," in *Design Automation and Testin Europe (DATE)*, Florence, 2019.

[8] O. Stursberg and B. H. Krogh, "Efficient representation and computation of reachable sets for hybrid systems," in *Hybrid Systems: Computation and Control: 6th International Workshop, HSCC 2003 Prague, Czech Republic, April 3–5, 2003 Proceedings 6*. Springer, 2003, pp. 482–497.

[9] M. Althoff, "An Introduction to CORA 2015," *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[10] F. Aul, N. Katsaouni, L. Krischker, S. Schmalhofer, M. H. Schulz, and L. Hedrich, "Schematic generation of programmable analog neural networks for signal proccessing," in *SMACD/PRIME 2021; International Conference on SMACD and 16th Conference on PRIME*. VDE, 2021, pp. 1–4.

[11] Y. Abu-Haeyeh and L. Hedrich, "Formal verification of nonlinear analog circuits using state space-based model order reduction," in *2024 20th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2024.

[12] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.