Deep Neural Networks for Behavioral Modeling of Analog ICs

André Carneiro Amaral

Introduction to the Research in Electrical and Computer Engineering

Supervisor: Doctor Nuno Cavaco Gomes Horta

Co-supervisor: Doctor Nuno Calado Correia Lourenço

Examination Committee

Supervisor: Doctor Nuno Cavaco Gomes Horta

Member of the Committee: Fabio Moreira de Passos

January 2022

Resumo

Este trabalho de tese de mestrado está inserido na área de Automatização de Projetos Eletrónicos e tem como principal objetivo proceder à modelação de circuitos analógicos ou dos seus respetivos componentes. Para atingir o objetivo proposto recorre-se às redes neuronais artificiais, com o intuito de criar um modelo que represente o comportamento do circuito e que seja rápido de simular. Uma vez que são gerados dados de entrada e saída através da simulação do circuito, o treino desta abordagem será supervisionado. Contribuições recentes neste campo de investigação têm demonstrado que as redes neuronais são capazes de imitar o comportamento de circuitos produzindo um erro muito residual, entre o comportamento do circuito e o modelo, e acelerando as simulações. Neste trabalho é proposto o conceito de estrutura nos dados de entrada reduzindo assim o número de parâmetros para treino, levando a uma aprendizagem da rede neuronal mais eficiente e mais generalizada. Para tal são usadas redes neuronais de grafos e redes neuronais recorrentes para efetuar a modelação comportamental de circuitos. Este método é aplicado a topologias de amplificadores operacionais e uma correta avaliação dos resultados obtidos é feita bem como a discussão dos mesmos. Resultados preliminares foram obtidos usando um conjunto de dados construído através da simulação de um amplificador. Tais dados são usados para treinar um modelo “Long Shot Term Memory” (LSTM), de modo que este simule o comportamento do circuito. Uma análise dos resultados preliminares mostra que usando LSTM é possível modelar o comportamento do amplificador obtendo resultados encorajadores.

Palavras-chave

Automatização de Projetos Eletrónicos, Aprendizagem Supervisionada, Circuitos Analógicos, Modelação, Redes Neuronais Artificiais.

Abstract

This work is integrated into the Electronic Design Automation (EDA) area, and its main goal is to model analog circuits. To achieve the proposed goal, Artificial Neural Networks (ANNs) are used to model the circuit, speeding up the simulations. Since the pair input-output is generated through SPICE simulation, the type of learning is supervised. Recent contributions in this research field were adequately discussed and show how powerful the ANNs are in modeling circuits behavior, providing residual error results when compared with an off-the-shelf simulator, and also, push-button speed. This work proposes to model the circuit behavior with Graph Neural Networks (GNN) and Recurrent Neural Networks (RNN). Introducing structure in the input data and thus reducing the number of trainable parameters, and also, turning the model efficient and more general. A set of operational amplifiers will be used to validate the proposed approach. A preliminary study on the LSTM model was performed, where data was obtained from a low noise amplifier circuit and the LSTM model trained to mimic the output of the circuit. A careful analysis of the preliminary results shows an accurate behavioral modeling of the amplifier circuit using a LSTM network.

Keywords

Analog Circuits, Artificial Neural Networks, Behavior Modeling, Electronic Design Automation, Supervised Learning.

Table of Contents

List of Figures

List of Tables

Acronyms

ADC	Analog to Digital Converter
AI	Artificial Intelligence
AMS	Analog-Mixed Signals
ANN	Artificial Neural Network
BGR	Band-Gap Reference
Bprop	Backpropagation
CompNN	Compositional Neural Network
CPW	Coplanar Waveguide
DAC	Digital-to-Analog Converter
DL	Deep Learning
EDA	Eletronic Design Automation
EM	Electromagnetic
FFNN	Feedforward Neural Network
GNN	Graph Neural Network
HDL	Hardware Description Language
IC	Integrated Circuit
LSTM	Long Short-Term Memory
MIMO	Multiple-input Multiple-output Circuit
ML	Machine Learning
MLP	Multilayer Perceptron
MS	Mixed Signal
MSE	Mean Squared Error
NARX	Nonlinear Auto-Regressive Neural Network with Exogenous Input
RBF	Radial Basis Function
ReLu	Rectified Linear Unit
RF	Radio Frequency
RNN	Recurrent Neural Network
Rprop	Resilient Backpropagation
SoC	System on Chip
TDNN	Time Delay Neural Network

Chapter 1

Introduction

This first chapter introduces analog Integrated Circuits (IC) design, emphasizing the challenges when modeling a circuit. Furthermore, the Machine Learning (ML) concept is introduced as well as the possible use of Artificial Intelligence (AI) tool to create accurate behavior models of analog circuits as an essential step to the production of Electronic Design Automation (EDA) tools.

1.1 Motivation: The Mixed-Signal Design Problem

Technology development set the electronic industry under pressure due to the high demand for integrated systems. As we witnessed, even before the Covid-19 pandemic, many factories had to suspend production due to the lack of circuits components. In another perspective, IoT is joining millions of devices, sensors, and actuators together, enabling some incredible breakthroughs in healthcare, education, agriculture, among many other areas. The IC continues evolving over the years and, despite not being an "eye catch" development, it is evident that the circuits are getting more complex, power-efficient, with reduced sizes and integrated, which is synonymous with improvements in this area. Most of the systems and sensors use a combination of analog/Radio Frequency (RF) and digital circuits in the same chip, commonly called "Mixed-Signal (MS) System-On-Chip (SoC)." In a MS SoC, the analog circuitry is the bridge between the digital circuits and the physical devices.

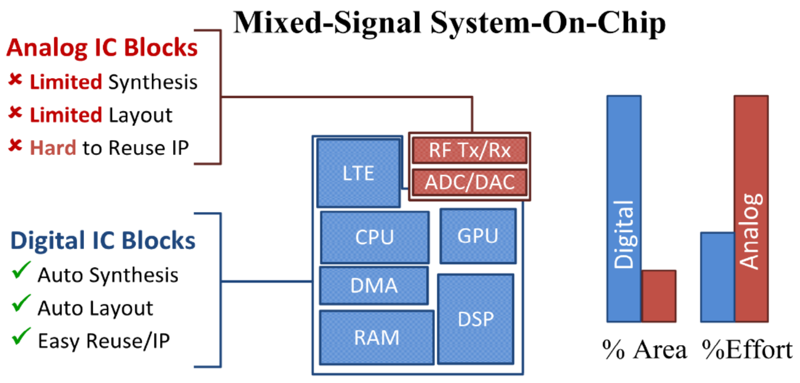


Figure 1.1 - Differences in terms of area and implementation effort between Digital and Analog from the designer perspective (from [1])

Analog circuit design is much less automated than its digital counterpart. However, the connection between the outside stimulus and the circuit is made. Below, some functionalities that are intrinsically analog and cannot be performed by digital circuits are referenced [1].

- **Sensing The System Inputs:** A signal from a transducer, microphone, or antenna needs to be sensed, amplified, and filtered to have the necessary a good signal-noise relationship and less distortion to digitalize the signal. Such analog front ends are pervasive in IoT, Healthcare/Wearable, and communication devices.

- **Conversion Between Analog and Digital Signals:** MS SoC circuits, such as Analog to Digital Converter (ADC), enable converting an analog signal to digital and allow digital signal processing. In other words, they allow the signal to be processed analyzed using digital circuity. After all the analysis and processing of the signal, a Digital-to-Analog Converter (DAC) can transform the digital signal into an analog one and have low distortion.

- **Regulate and Provide Power:** To properly use the ADC, they need to have a stable reference of the voltage and current. This can be achieved with voltage/current reference circuits and crystal oscillators.

Analyzing Figure 1.1 (reprinted from [1]), it is clear that, in an MS SoC, despite the higher area occupied by digital circuits, the effort in their design is reduced as the design flows are somewhat standardized and design automation is well established. On the other hand, designing analog circuits is still challenging and lacks automation support despite their lower area. In most cases, the circuit designer does analog circuits' design manually to achieve several contradictory specifications. This difference in the level of automation happens because analog circuits, in general, are less systematic and require exhaustive knowledge in their design.

The facts presented above underline the importance of developing tools to automate the design of the analog circuits, reducing the costs, replacing eventual stock failures, and speeding up the simulations. To sum up, it is essential to keep innovating in this area of research to reach a global automation tool that would allow the development of new technologies and deal with analog circuits more efficiently.

1.2 Overview: Analog and MS IC Design Flow

Before exploring the IC circuit design automation in more detail, it helps to have a brief overview of the IC design flow. To obtain an analog IC, a designer must follow steps to achieve the objective successfully. Some of these steps might differ depending on the company/designer since they can have a specific design flow. Gielen and Rutenbar's point of view, present in [2], shows a design flow strategy of a circuit with top-down design steps that are repeated from the system-level to the device level, with a bottom-up approach for verification and validation of the model. The approach is split into seven stages, as can be seen in Figure *1.2*:

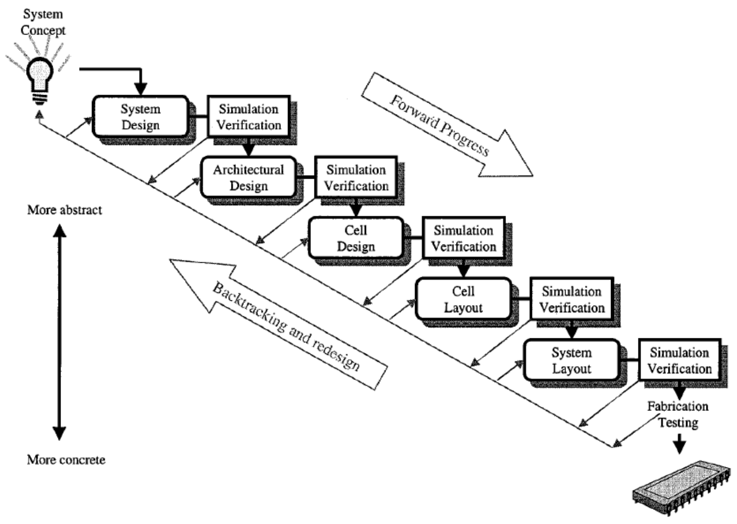


Figure 1.2: High-Level view of Analog IC Design Flow (from [2])

1) **Conceptual Design:** This is the stage to conceptualize the product, where specifications for a design are gathered and the overall product concept is developed. It might also be used for setting project management goals.

2) **System Design:** In this stage, the system's overall architecture is designed and partitioned. It is also a stage to define which parts are hardware and software and the interfaces to be used.

3) **Architectural Design:** The decomposition of the hardware part in functional blocks is the primary concern. Those functional blocks, when connected, must fulfill the behavioral specifications. This stage includes the division between analog and digital blocks. Finally, the blocks are implemented in a hardware description language (HDL) (e.g., VHDL and VHDL-AMS). The high-level architecture is then tested through simulations to check if the specifications defined in the Conceptual Design stage are fulfilled.

4) **Cell Design:** For the analog blocks, this is the stage in which the detailed implementation of the different blocks for a given specification is made, leading to a completely sized device/level circuit schematic. The obtained circuit design is validated against the specifications using the SPICE simulations.

5) **Cell Layout:** In this stage, a multilayer layout is obtained by translating the electrical schematic, of the different analog blocks, into a geometric representation. It involves area optimization, obtaining layouts that occupy a minimum amount of space in the chip, and parasitic reduction, which negatively impacts the circuit's performance.

6) **System Layout:** In this stage, the system-level layout is generated, including the system-level block place, route, and power-grid routing. It is also important to include descriptions of shielding or guarding that act as a measure to avoid crosstalk and substrate coupling. Further, the IC is verified, performing validation with the embedded software.

7) **Fabrication and Testing:** The stage where the ICs are fabricated. Simultaneously with the fabrication, the devices are tested and, if any defect is detected, they are rejected.

Those previously mentioned steps are a simple representation of a complex process to design analog IC. Backtracking and verification are required during the entire flow and are done frequently.

In [1][2], the authors provide the same stages but with a more detailed hierarchical analog design methodology. In Figure 1.3, the proposed steps are detailed.

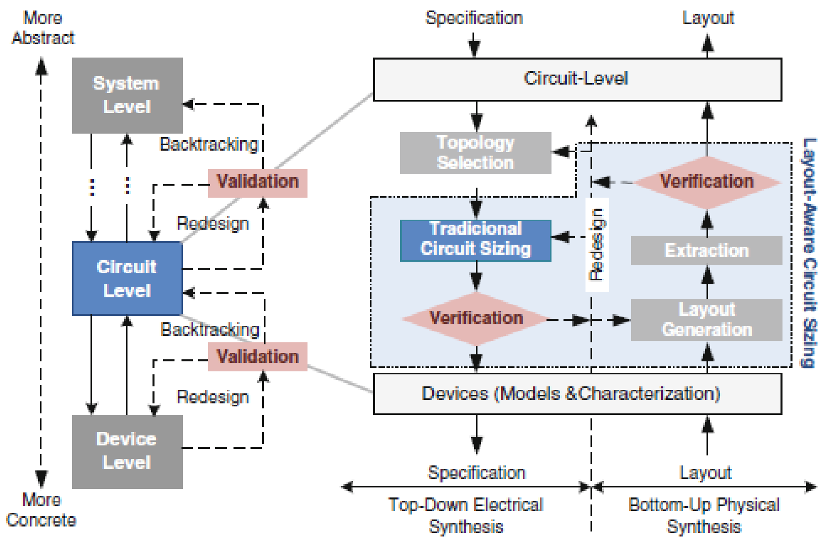


Figure 1.3 - Hierarchical level Design Detailed (*from [1][2]*)

In Figure 1.3, we can see a top-down and a bottom-up approach in the circuit level phase, which is needed to ensure circuit quality avoiding defects before moving on to further phases. Adopting a top-down design methodology as proposed in [2] turns possible to explore complex system architectures leading to better system optimization at a higher abstraction level, which is essential since after this step is added more specific implementations at the lowest level. The number of hierarchical levels is dependent on how complex the system is. It has no pre-defined rule to follow and is based on the designers' experience.

The top-down approach is divided into the following processes [1]:

1) **Topology Selection:** section where the blocks and the correspondent connections are defined.
2) **Traditional Circuit Sizing:** each block gets its specifications translated from higher-level specifications. The block specifications can be, for example, the number of transistors or the gain of a circuit. This process is repeated until all the blocks have assigned their specifications

Once the top-down approach finishes, the bottom-up starts, which is composed of the following steps:

1) **Layout Generation:** the layout of each level is obtained from the structure of the lowest hierarchical levels.
2) **Layout Extraction:** the layout generated is now incorporated in a model, proper to be verified.

It is essential to remember that a verification phase where the simulations might reveal problems exists in both approaches. So, in that case, backtracking and redesigning the model happens when necessary. Analog behavioral models, the focus of this work, are indispensable parts of the analog IC design flow. They enable the simulation of large and complex electronic circuits and allow the exploration of design alternatives in a top-down approach, helping to identify the specifications for the sub-circuits in large designs. Moreover, they can be used in design automation tools to move to larger and more complex circuits. However, to reduce design iterations, analog models must account, as best as possible, for the non-ideal behaviors of the circuits.

1.3 Modeling with Machine Learning and Verilog-A

Designing a circuit involves many tradeoffs, making it challenging to fulfill all the specifications required. Modeling a circuit in the architectural stage is an exhausting task that needs to capture the circuit behavior over all the constraints that need to be fulfilled. Typical analog IC circuit designs are very time-consuming due to the complex relationship between the design parameters and the circuit specifications and the number of parameters that the designer must consider. To overcome those issues, ML techniques can generate behavior models of a circuit that mimics their functional behaviors speeding up the simulation time. Machine Learning is a process in which a computer improves its capabilities after analyzing several past experiences. Nowadays, it is used for speech recognition, computer vision, natural language preprocessing, robot control, and many other areas. The AI experts recognize that it is easier to use input-output examples to train a system rather than program it manually, anticipating the output for all given inputs [3]. With novel uses of machine learning, particularly artificial neural networks (ANN) and deep learning (DL), it is possible to close the accuracy gap between Analog & Mixed-Signal (AMS) ICs' behavioral models and their corresponding spice-level (pre-and post-layout) models. New hybrid gray-box circuit models that combine simplified behavioral models with ML to account for implementation-dependent SPICE-level effects are to be investigated, aiming at accuracy levels close to physical-centric (white-box) manual modeling while automatically tuned to the specific topology and technology. Figure 1.4 represents the approach to consider when modeling the behavior of a circuit or device using ANN.

Verilog-A uses mathematical descriptions to describe electrical or non-electrical behavior in terms of input and output ports for components, circuits, or even systems [5]. After all, the model needs to be converted to an HDL (e.g., Verilog-A), to be used for simulation. This serves as starting point of this thesis.

1.4 Objectives

The main goal of this work is to create, using ML methodologies, a behavior model of the device/circuit allowing to speed up simulation when used in complex circuits simulations with other sub-blocks. As it is debated in chapter 2, many studies are focused on one model to one circuit. This work focuses on amplifiers and intends to create one model capable of describing the behavior of multiple amplifiers circuits, turning possible to generalize the model to upcoming amplifier circuits.

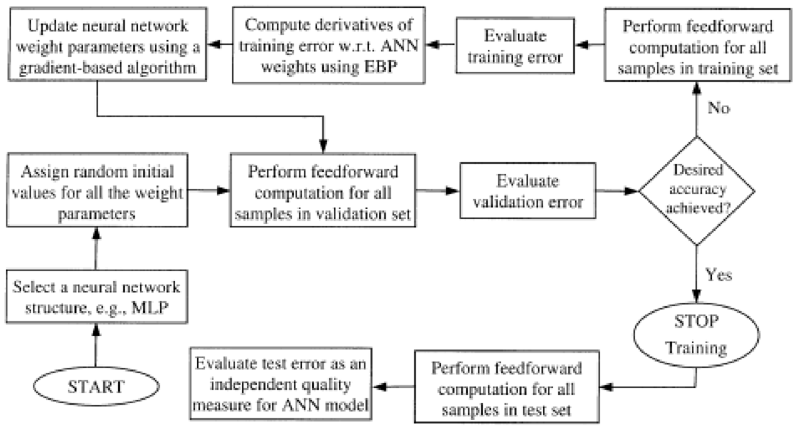

Update neural network weight parameters using a gradient-based algorithm
Compute derivatives of training error w.r.t. ANN weights using EBP
Evaluate training error
Perform feedforward computation for all samples in training set
No
Assign random initial values for all the weight parameters
Perform feedforward computation for all samples in validation set
Evaluate validation error
Desired accuracy achieved?
Yes
Select a neural network structure, e.g., MLP
STOP Training
START
Evaluate test error as an independent quality measure for ANN model
Perform feedforward computation for all samples in test set

Figure 1.4 - Flowchart ANN Modeling Approach (from [4])

The main objectives for this work are the following:

- Extract data from different amplifiers, creating a data set for each one, which contains the pairs input-output and organizes the data in a graph structure;
- Apply Graph Neural Networks (GNN) to properly extract the essential features that characterize the device topology and the circuit sizing;
- After applying the GNN, is applied an LSTM. This network provides the circuit behavior in an output waveform in the time domain;
- In the end, this model can be used in AIDA [27], a design automation solution developed at IT, to speed up the simulations of complex analog blocks composed of multiple amplifiers, such as ultra-low power analog front ends for biomedical applications [6].

1.5 Document Structure

This document structure is the following:

- Chapter 2 presents the state-of-the-art analog IC behavior modeling, presenting the numerous types and modeling techniques applied in different circuits. It is also analyzed the state-of-the-art techniques to convert artificial neural networks to Verilog-A, which is planned to be used.
- Chapter 3 presents the proposed solution for capturing the behavior of topologies of amplifiers using only one deep learning model for all the circuits and a structure to the input data. It also overviews the principal models needed for this work, such as GNN and LSTM networks. Some preliminary results are also provided. In the end, a timeline of the work for the thesis is presented.

Chapter 2

State-Of-The-Art

In this chapter, the most relevant behavior modeling techniques for analog IC, using ML methods, are analyzed and discussed. A detailed review of the latest articles in this area will frame this work in the state-of-the-art. After that, an ML overview is presented to describe and compare the existing methods used to model the behavior of circuits/devices. Finally, is presented an overview of Verilog-A for system verifications through simulations.

2.1 Modeling of Analog/RF Circuits with Machine Learning

The design of analog IC is very time-consuming due to the non-linearity relationship between the design parameters and circuit/device specifications. Generally, a handmade calculation can restrain the design space and provide a good starting point to the designer. However, the process is still cumbersome due to many iterations to achieve the expected specifications. Recently ANNs have become a solid alternative to model analog circuits behavior. In the literature, studies attempt to model circuits behavior in many different fields of analog IC.

In [7] the behavior of a switched-capacitor three-stage cross-coupled charge pump circuit is modeled. First, the data is obtained by circuit simulation, and the input (current between nodes, clock time, and voltage) and the correspondent outputs are preprocessed, leading to a balanced dataset. A Multilayer Perceptron (MLP) [8] is built to find the charge amount and implement a specific charge transfer function. It has 3 hidden layers (the first one with 50 neurons, the second one with 20 neurons, and the last with 10 neurons), with hyperbolic tangent as an activation function and Adam [9] as the optimizer. The resulting model deviates a maximum of 9.6% from the circuit behavior and a reduction of 7 seconds on the simulation time. After that, the model was implemented in VerilogAMS. According to the authors, training the model with the border values of the dataset leads to more unstable results. They suggested a creation of a safety margin in the dataset, preventing the model from being trained with that border values.

Grabmann, Landrock and Gläser in 2019 proposed to create a model to analyze the power consumption of a circuit transient [10] since manually written behavior models cannot capture the power consumption at the transistor level. As a case study, it uses a low relaxation oscillator. The inputs (frequency, time clock, and EN) are obtained from circuit simulation in SPICE with a sample period of 10ns. The data is scaled, and only the equidistant data will be kept. After that, they propose a Time-Delay Neural Network (TDNN) with 2 hidden layers, the first one with 50 neurons and the second one with 10 neurons. The network also contains 10 steps of delay. The hyperbolic tangent is the activation function for the hidden layers, and the output layer is the linear one. The optimization is done using the Gradient Descent algorithm. The paper results report that the power consumption curve differed with an error of 2.7% when using the model or the circuit in the simulation. The simulation time is reduced by 95%.

In the same line, Suissa in [11] proposes a neural network method to model the power consumption of analog components. The methodology is divided into 3 steps:

- **Measurement Step:** Sweep the input parameters of the circuit over the entire operating range and obtain, as an output, the power consumption for each case.

- **Model Creation:** Using an Feed Forward Neural Network (FFNN) with one hidden layer with the sigmoid activation function and a linear activation function in the output layer. The training is performed using the Levenberg-Marquardt method

- **Integration:** In this phase, the power consumption model is integrated with the functional behavior model of the circuit, as can be seen in Figure 2.1. This stage allows the circuit's behavior model and its power consumption behavior.

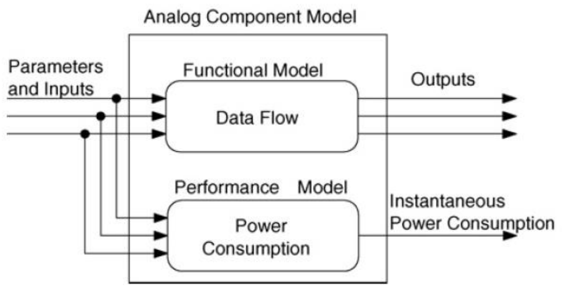


Figure 2.1 - Integration Power Consumption (from [11])

The case studies presented to validate the methodology where the neural network was built for an amplifier, an analog to digital converter, and, finally, a wireless transceiver RF. The average error is about 1.53%, and the maximum error is about 3.06%

In [12], a technique is proposed to model circuits or devices with collaborative stimulus generation. The main goal, reducing the difference between the model and circuit behavior, is achieved using a recurrent neural network (RNN). The algorithm architecture, shown in *Figure 2.2*, is the following:

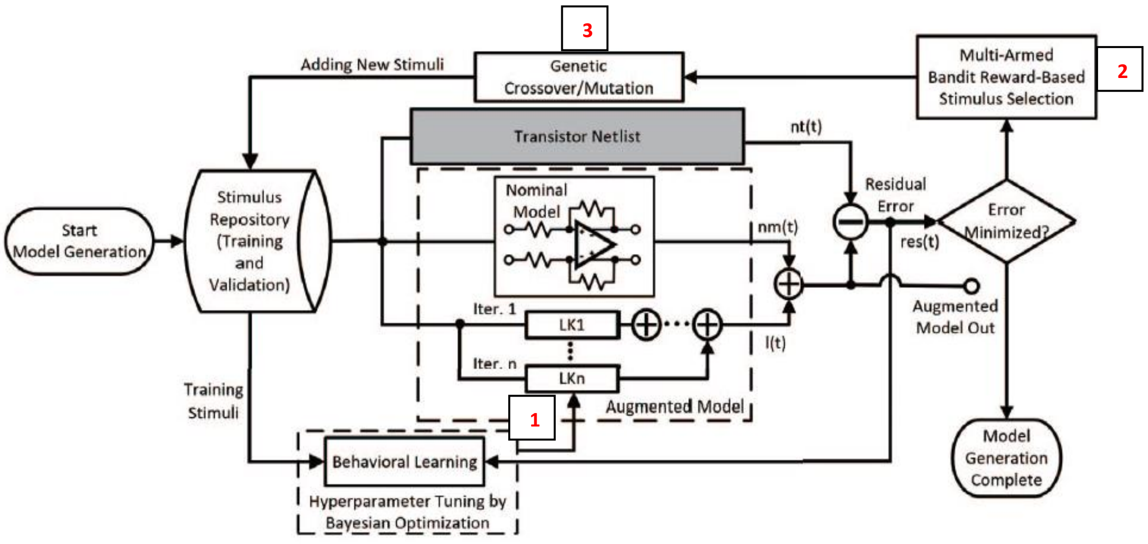


Figure 2.2 - Main Goal Model Architecture (from [12])

1) *RNN Model (1)*

- Recurrent Neural Network is used to capture the memory and non-linearity relationship between the inputs and outputs of the circuit.
- It takes the current input, past inputs, previous outputs, and residues of previous iterations to obtain the output and the residual error, between the augmented model and the transistor netlist, in the present iteration.
- Bayesian optimization is used to find suitable hyperparameters for the RNN. It has the stimulus and the residues as input, and the output is the kernels (LKn).

2) *Test Stimulus Generation (2)*

- Stimuli are assigned to a probability of being selected to the mutation phase. This probability is calculated as the error in that stimulus over the sum of all the stimulus errors. A higher error means that the stimulus has an underperformed learning kernel.
- In terms of the kernels, once learned, they will not be erased to ensure that previous learning behaviors are not unlearned.

3) *Genetic Crossover/ Mutation (3)*

- The highest probable stimulus goes through an algorithm based on genetic algorithms that will generate a new stimulus, that contains information and a certain level of randomness, to find a new stimulus that allows training the model with new differences (between the model and the circuit) that were undiscovered.

The algorithm stops when the residuals are minimum. That is the case where the model cannot excite further differences between the circuit and the model.

Using this approach, Lei and Chatterjee [12] show the results for modeling different circuits:

1) *Low-Dropout Regulator:* the behavior model converges in 6 iterations with an RMS = 10,4 μV.
2) *RF Receiver:* the behavior model converges in 5 iterations with an RMS = 13.8 μV.
3) *Fully Differential Amplifier:* With a nominal gain of 2, a bias circuitry of 45mm process, and 1V supply, the model has an RMS of 69 μV.

The secondary goal is to design a diagnose algorithm to detect anomalies in the transistor-level components. This is achieved using Volterra learning kernels since ML techniques are not the best for detecting anomalies due to the lack of cases when a failure exists. In *Figure 2.3*, there is an illustration of the proposed algorithm:

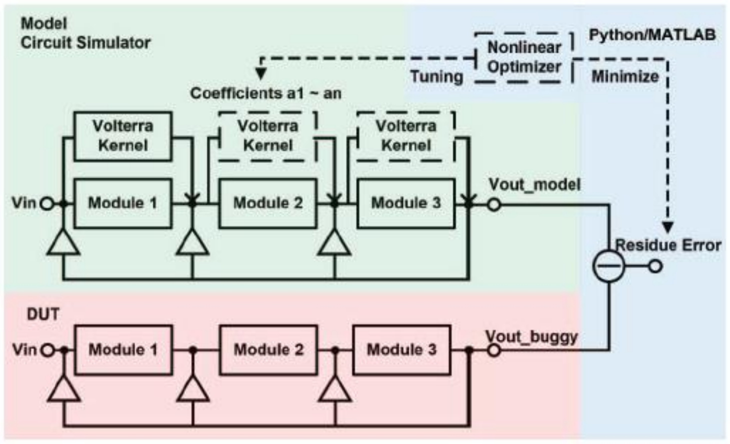


Figure 2.3 - Diagnose Algorithm Architecture (from [12])

A Volterra kernel is inserted for each module, and in each iteration, the residue is produced. It is optimized using the Levenberg-Marquardt algorithm leafing to tune the kernels better. To find the bug, if the residual error is minimum and the kernel associated with a particular module gets updated, it means the correspondent modulo has a failure or a bug.

To test this approach, the authors used the following circuits, and for each one, a conclusion about the Volterra kernel placement is obtained:

1) *RF Receiver:* The error is minimum, measured by norm L2, if the Volterra kernel is placed across the buggy modules, indicating a correct diagnosis of the bug in the module.
2) *Sigma Delta:* A smaller error occurs when the kernel is across the first integrator rather than in the second integrator indicates that the bug is in that first integrator.

To model the behavior of a Multiple-Input Multiple-Output circuit (MIMO), Hasani *et al.* in [13] propose the use of a Compositional Neural Network (CompNN) built with a nonlinear autoregressive Neural Network with Exogenous Input (NARX) [14] and a TDNN. Each feature has a small-sized NARX allowing for an observable model where the behavior of each feature can be analyzed separately. The output of each NARX is the input of the TDNN, which computes the final output of the MIMO circuit. A CMOS Bandgap Voltage Reference circuit (BGR) is presented as a case study. *Figure 2.4* represents the features and architecture of the CompNN for this particular case.

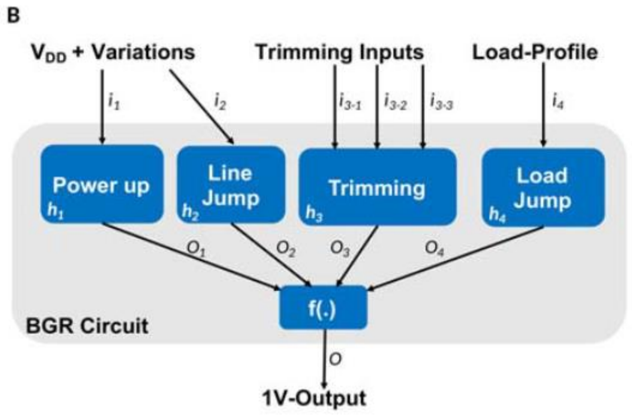


Figure 2.4 - CompNN architecture for MIMO BGR (from [13])

The h_1, h_2, h_3, h_4 are the small-sized NARX neural network for each of the input's features, power up, line jump, trimming and load jump, respectively. For trimming the NARX has 3 delay components and 10 neurons in the hidden layer, for load jump and line jump features it has 3 delay components and 7 neurons in the hidden layer. Those parameters are found using Grid Search method. The O_1, O_2, O_3, O_4 are the output of each small-sized NARX. The f(.) represents the TDNN that merges the outputs of each small-size NARX and generates the model output. It contains 200 neurons in the hidden layer. The data is acquired by input-output circuit SPICE simulation, and it is divided into the train (70%), test (15%), and validation (15%) sets. For training is given stop criteria, such as no improvement on the test set of the cost function being minimized, learning rate gets higher than 10^{10}, maximum training epochs are reached, and error is less than 10^{-7}. Finaly, the results show an error-rate at the 1-V Output of 5% and a decrease of the simulation time by a factor of 17.

In [15], a technique is proposed to capture the behavior of an oscillator in its transient mode. The main goal is to capture the oscillator output waveform. Oscillators cannot be defined in a state-space realization because in trapezoidal waveforms the plateau turns impossible to trigger the transition of the state-space model, so the authors proposed a feedforward neural network (FFNN) with a Periodic Unit, which simulates the oscillator transient.

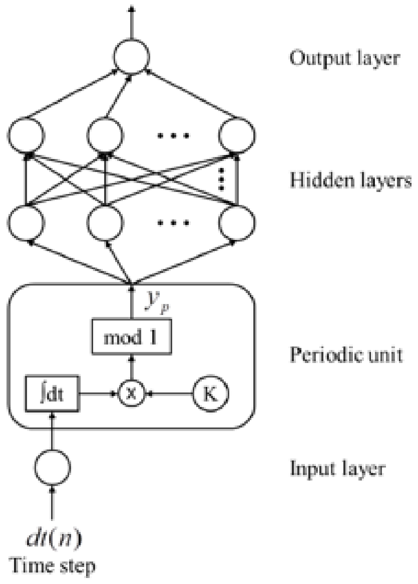


Figure 2.5 - Oscillator Model (from [15])

Figure 2.5 represents the model proposed by the authors. Analyzing it with more detail:

1) <u>Periodic Unit:</u> the inputs are a chronological sequence of time steps. The output of this unit is given by equation (1) where the phase is normalized between [0,1] solving the arbitrary phase behavior.

$$y_p(n) = (K_{osc} \sum_{1}^{n} dt(i))\ mod1 \tag{1}$$

2) <u>Hidden Layers</u>**:** the nodes' values are obtained by equation (2), meaning that the value is obtained with the sum of the multiplication of the previous neuron value, of the previous layer, with the weight difference, between the actual neuron layer and the previous one, plus a bias of the actual neuron layer.

$$y_j(n) = \tanh\,(\sum_{l=1}^{N} y_{prec}^{l}(n) w_h^{l,j} + b_h^{j}) \tag{2}$$

3) <u>Output:</u> the output value is obtained by equation (3), meaning that the output value is the sum of the multiplication of the previous layer neurons' output with the weight of the output neuron, adding the bias of the output one.

$$y_j(n) = \sum_{l=1}^{N_{prec}} y_{prec}^{l}(n) w_{out}^{l} + b_{out} \tag{3}$$

To train the FFNN, since the *mod1* function is discontinuous, is assumed that *mod1* value is 1 in the discontinuities. Equation (4) is the objective problem that is achieved using the Levenberg–Marquardt Algorithm. Note that the minimizing parameters are the weight, the bias, and the oscillatory frequency so $phi = [w, b, K_{osc}]$.

$$minimize_{phi}\ \frac{1}{2}\sum^{N}(y_{out} - y_{predicted})\hat{}2 \tag{4}$$

As a case study is presented a behavioral model of an invert ring oscillator with an oscillatory frequency of 0.1689GHz in the first case, and the correspondent FFNN has 10 neurons in a single hidden layer. In the second case, the frequency changed to 0.5734GHz, also with an FFNN with 10 neurons in the hidden layer. In both cases, the simulation time reduces by a factor of 10, comparing it to the circuit simulation.

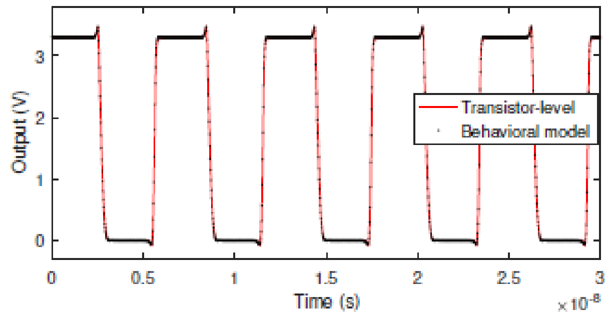


Figure 2.6 - Output Waveform for Oscillator Model, Case 1 (from [15])

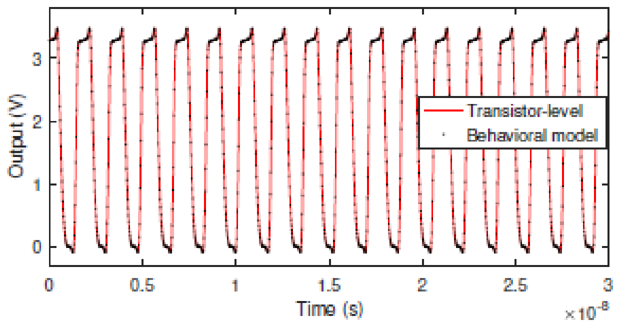


Figure 2.7 - Output Waveform for Oscillator Model, Case 2 (from [15])

Figure 2.6 and Figure 2.7 show the model's results comparing the model's output waveform with the circuit simulation.

Deepening the oscillators behavior modeling, in [16], the same authors of [15] proposed a complementary approach to the previous one. In this case, a buffer is added to the oscillator's output. The presence of a buffer allows the oscillator to generate high-quality signals. Figure 2.8 shows the adjustment made to the previous model.

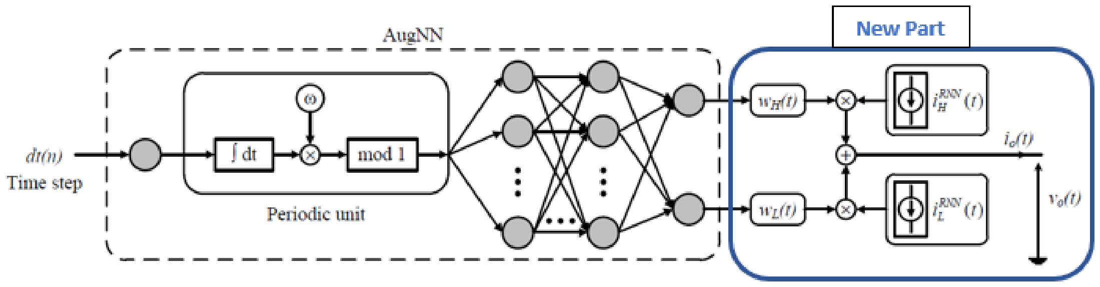


Figure 2.8 - Oscillator with Buffer Model (from [16])

The model was adjusted by adding a recurrent neural network (RNN) to capture the behavior of the buffer, which means obtaining the high and low current, which are fundamental to obtaining the output current. Equation (5) combines the weights providing from the AugNN with the currents obtained with the RNN producing the output current.

$$i_0 = w_h i_h + w_l i_l \tag{5}$$

To train the RNN is connected a multilevel piecewise-linear voltage source at the buffer output to generate the excitation identification signals fixed at high and low. The FFNN training is performed as it was described in [15], as well as its equations remain the same. This approach is applied to a Ring Oscillator with a Buffer and 1.9481GHz of oscillatory frequency. The RNN has 10 neurons in its single hidden layer and the FFNN in the AugNN has 15 neurons in the hidden layer. The results show a root mean square error (RMSE) of 0.0035 and a reduction in the simulation time of up to 96%.

Adding to what is present in [15], in [17] is added a voltage controller to, through voltage variation, control the instantaneous frequency. To proceed to that mapping is used a second FFNN inside the Periodic Unit as it can be seen in Figure 2.9.

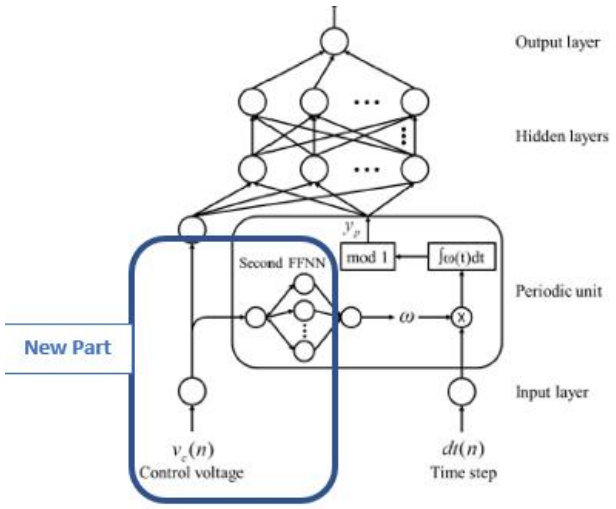


Figure 2.9 - Oscillator with a Voltage Controller Mode (from [17])

The second FFNN output is given by equation (6).

$$w(n) = \sum_{j=1}^{N} y_{h,pu}^{j} . w_{out,pu}{}^{j} + b_{out,pu}{}^{j} \tag{6}$$

Where w(n) is the oscillatory frequency, $y_{h,pu}^{j}$ is the output of the hidden layers of the second FFNN (which is inside the periodic unit), $w_{out,pu}{}^{j}$ is the weight of the output neuron of the second FFNN and $b_{out,pu}{}^{j}$ is the bias of the output neuron of the second FFNN. The $y_{h,pu}^{j}$ is given by equation (7).

$$y_{h,pu}^{j} = tanh\,(v_c(n). w_{h,pu}{}^{j} + \ b_{h,pu}{}^{j}) \tag{7}$$

Where tanh is the hyperbolic tangent, $v_c(n)$ is the voltage controller input and $w_{h,pu}{}^{j}, b_{h,pu}{}^{j}$ are the weight and bias, of the respective hidden layer neurons. So, the output of the periodic unit is given by equation (8).

$$y_p(n) = (\sum_{1}^{n} w(i).dt(i))\ mod1 \tag{8}$$

As can be seen from *Figure 2.9*, apart from the second FFNN, which allows controlling the oscillatory frequency, the rest of the model remains the same so, the original FFNN equations are equal to what is proposed in [15] and the training methodology is the same, although the minimizing parameters are $phi = [w_{original}, b_{original}, w_{second}, b_{second}]$.

To prove the usefulness of that approach is proposed a Voltage Controller Oscillator (VCO) circuit model with a voltage controller sweeping in a range of [1V, 3V]. The training data is obtained through SPICE simulations. The model has the original FFNN with 2 hidden layers with 20 neurons in the first hidden layer and 10 in the second one, and the second FFNN has a single hidden layer with 5 neurons. The simulation time reduces 93% and the model contains a root mean squared error (RMSE) of 0.61%.

In [4], ANNs are used to model the behavior of RF/microwave circuits. The authors state that using neural networks is a better alternative to conventional numerical methods, such as Support Vector Machine (SVM). They presented examples of how ANNs model signal propagation delays in Very Large-Scale Integration (VLSI) interconnect networks, coplanar waveguide discontinuities, and MESFETs to prove their point. In [18], the same authors present a study on ANN nonlinear techniques for modeling large/small signals of transistors and dynamic recurrent neural networks, which are then used to model circuits. To review the proposed techniques is used practical microwave examples. In [19] is proposed a method for modeling coplanar waveguide circuits (CPW) using ANN that is based on Electromagnetic (EM) simulations. In this approach CPW transmission lines, 90º bends, short circuit stubs, open-circuit stubs, step-in-width discontinuities, and symmetric T-junctions are modelled individually using ANN based on EM. This multilayer feedforward ANN is composed by one input layer, one hidden layer and one output layer and the training is performed using an error-backpropagation learning algorithm. As a case study the authors proposed to model a CPW folded double-stub filter and a 50-3-dB power-divider circuit. The proposed architecture (EM based ANN) can be applied in other classes of microwave and millimeter-wave circuits. One of the major drawbacks of this approach is that it takes long time in the training phase to obtain an accurate model and the efficiency can be low, so in [20] is proposed a solution to model nonlinear RF microwave devices using a Radia Basis Function (RBF) MLP with Resilient Backpropagation (Rprop) algorithm for training. To address this problem the authors proposed a “divide to conquered” technique where the complex problem is divided in sub-problems, which are modelled by individual neural networks. This approach is claimed by the authors to improve the efficiency of the EM-based ANN. The RBF network is used to approach locally the input variables and its output is the input of the MLP network that will perform a global approach, since it improves the generalization capacity of the model, acting as an output network. It is graphical explained in Figure 2.10.

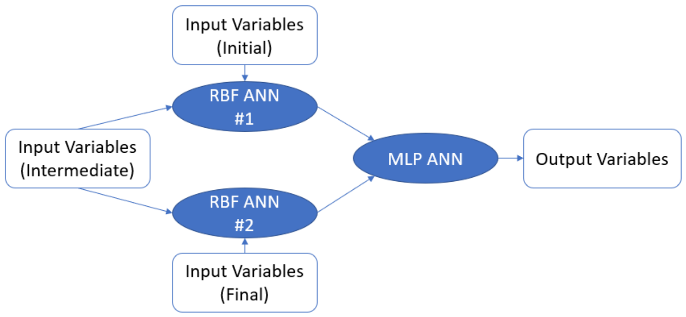


Figure 2.10 - RBF/MLP Proposed Model (adapted from [20])

To demonstrate the improvement made in EM-based ANN the authors, first, proposed a uniplanar compact-photonic bandgap (UC-PGB) rectangular waveguide. In the hidden layers of the three ANN are used 10 neurons, there are 10000 training epochs, and the obtained MSE is around $2 * 10^{-5}$. Second, is proposed a patch antenna with PGB substrate. In the hidden layers of the three ANN are used 15 neurons, there are 10000 training epochs, and the obtained MSE is around $2 * 10^{-4}$. The authors also conclude that using a single MLP ANN or a single RBF ANN has less generalization capacity, independently of the number of hidden neurons, rather than combining them. The results can be seen in Figure 2.11 and Figure 2.12.

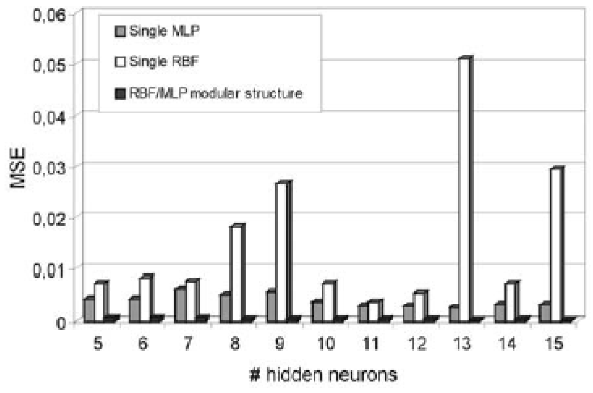


Figure 2.11 - RBF vs MLP vs RBF + MLP for UC-PBG waveguide (from [20])

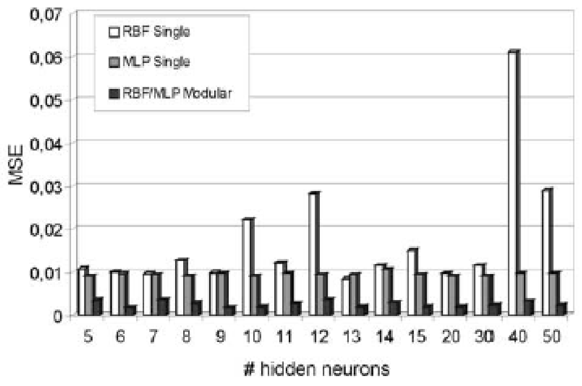


Figure 2.12 - RBF vs MLP vs RBF + MLP for patch antenna PBG substrate (from [20])

Another way to address the behavior modeling problem is using the Support Vector Machine (SVM). In [21] is used a SVM for regression problems to model analog circuits, the kernel used is the radial basis function (RBF), error function is ε since it allows a better sparsity of the solutions and the ε-function error contains a fixed and symmetrical margin, however if the data comes from real world problems a fixed and asymmetric margin or even a non-fixed and asymmetric margin are preferable. To preprocess the data is necessary to use cross-validation to prevent overfitting and an exponential Grid Search to find the parameters C (penalty degree for regression errors), γ and ε (RBF parameters). The use of exponential Grid Search will allow more straightforward parallelization. The dataset should represent well the data, including the circuit dynamic, static properties, excitation waveform, and frequency spectrum.

As a case study is presented a SCFL Buffer Cell with the objective to approximate the voltage and current output curves, the results are a mean squared error (MSE) of $2.62 * 10^{-14}\ A^2$ for the current and $3.20 * 10^{-6}\ V^2$ for the voltage. For the Resistive Mixer circuit, a MSE of $2.89 * 10^{-7}\ V^2$ is obtained for the output voltage. Finally, is analyzed a Voltage Controller Oscillator and Figure 2.13 shows the proposed model, where the three SVM are used to estimate the parameters that allows the signal generator to output a voltage that is similar to the Voltage Controller Oscillator voltage output.

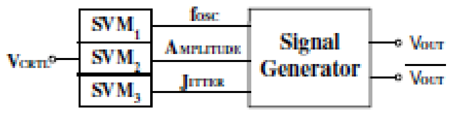


Figure 2.13- VCO SVM Model (from ([21]))

To train the model is only needed 10 to 20 input-output pairs of measurements, due to the reduction of the complexity provided by the SVM method. The model output is compared to adjacent period jitter, where it shows a competitive accuracy, however no MSE is presented. It is also worth to mention that is not referred in this article the time reduction, which is a crucial aspect when modeling circuits and it is an important metric to compare with the ANN model, for the same circuits. In [22] the SVM is used to classify the region as infeasible or not and prune the ones that are infeasible, excluding a large portion of the design space. The generated feasibility classifier can work in conjunction with the performance macro models in analog synthesis, providing the designer with a good starting point.

2.2 ANNs vs. SVM: Advantages and Drawbacks

Since the state-of-the-art review presented methods for modeling circuits with ANN and SVM and this work's focus is on using ML for modeling, it is essential to discuss the advantages and disadvantages of their use. Therefore, the following advantages were identified for the ANNs:

- High-speed "simulations" and low computation complexity;
- No complete knowledge about the physic devices is needed;
- Overcome the accuracy limitation of the numerical models;
- Can handle optimization constraints;
- Due to a high sample efficiency, they show a higher accuracy in fewer training epochs.

These advantages make the ANNs the most suitable technique for modeling the circuit's behavior since they can also deal with nonlinear problems mapping the relationships between the input and output data. As for drawbacks, the ANNs have the following ones:

- A lot of data is required to achieve higher accuracy;
- Numerous hyperparameters that have to be tuned to improve accuracy and avoid overfitting;
- Can diverge with a high learning rate.

The other model presented in the state-of-the-art that is capable of mimicking the behavior of analog circuits was the SVM, which has the following advantages:

- Requires less data than the ANNs and does not converge to a local minimum of the cost function;
- Higher accuracy in classification problems;
- Has well-documented approaches to avoid overfitting, and, using the Kernel Trick, it can map nonlinearly separable data to a feature space where the data is linearly separable.

Several disadvantages of the SVMs were identified:

- Not suitable for large datasets;
- Have difficulties dealing with noisy data;
- Hyperparameters need to be tuned specifically to each circuit dealt, leading to a lack of generalization;

As mentioned in the previous sections, the use of ANN for modeling circuits behavior has been widely researched and discussed. These neural networks with the number of hidden layers, number of neurons, and weights correctly set can provide accurate results allied with high-speed “simulations”, avoiding the need for a detailed and time-consuming physical base formulation. Considering all the advantages and drawbacks of the ANNs and SVMs, the ANN methodology is therefore chosen for this work.

2.3 ANNs and Verilog-A: An Overview

Once the ANNs for the circuit models are created and trained, their description must be implemented in HDL such as Verilog-A to use the model during circuit simulation. This section presents a summary of state-of-art models involving Verilog-A and neural networks. In [23] is presented an approach to convert a neural network with backpropagation to Verilog-A using a top-down methodology. In Figure 2.14, we can see the neural network with its weights, and on the right the correspondent Verilog-A model that takes as inputs the weights and returns the bias for each neuron in each layer. Before starting coding in Verilog-A is extracted from the neural network its respective equations. In the first part, the needed variables are declared. More specifically, in the *directional specifications,* the input and output are defined, in *discipline definition* is specified, which inputs and outputs are electrical, in *parameter declaration* and *variable declaration* auxiliary variables are created. The second part is where the behavioral statements are declared, and the initial conditions, such as bias and weights, are defined. In the third part, the teaching signal is set, and the model is trained using the equations that represent the neural network. In the last part, the output is obtained, and the respective error and the weights are updated.

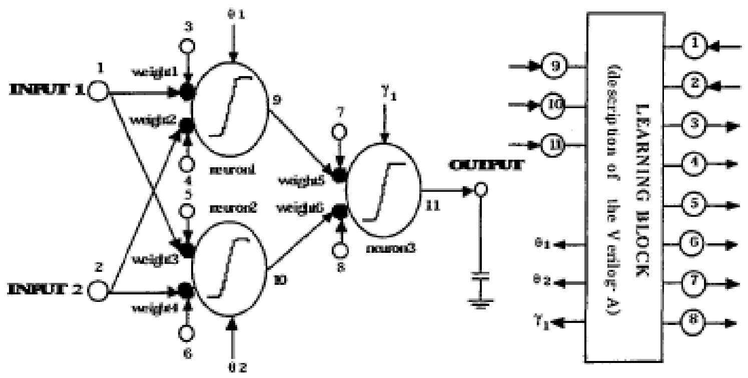


Figure 2.14- In Left the Neural Network, in the Right the correspondent Verilog A model (from [23])

In [24] is proposed the translation of an RNN behavior model to Verilog-A in order to make a modified nodal analysis of the model. For that purpose, the differential equations of the RNN need to be converted to differential equations. It also allows using a different time step in evaluating the model.

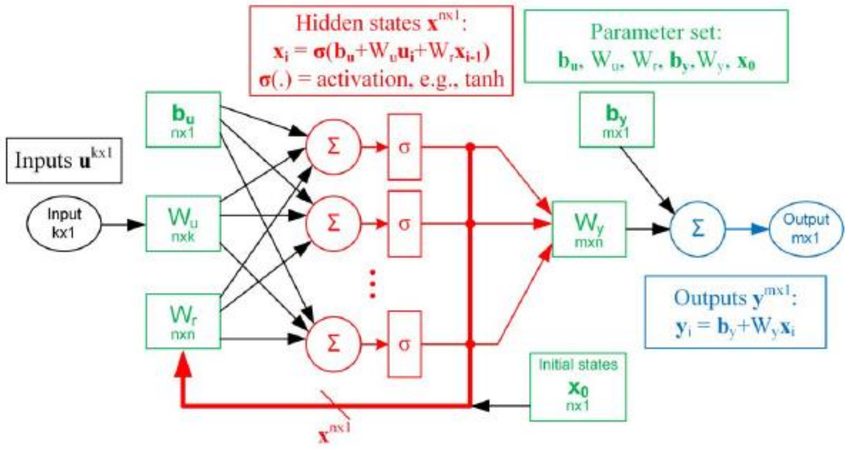


Figure 2.15 - RNN Model focuses on hidden layer and output layer (from [24]).

In Figure 2.15 is present the structure and the model equations for one hidden layer and the output one. It is important to mention that this RNN will account with a key property of circuits, the zero-in zero-out (ZIZO), meaning that the DC current, when all terminals are biased at the same potential, is zero. From Figure 2.15 - RNN Model focuses on hidden layer and output layer (from [24]).is also possible to extract equations (9) and (10). The first one relates the output of the hidden layer (x_i) with the activation function (σ), bias (b_u), the respective model parameters (W) and the input (u_i). The second one represents the model output as function of the output of hidden layer and the model parameters.

$$x_i = \sigma(b_u + W_u u_i + W_r x_{i-1}) \quad (9)$$

$$y_i = W_y x_i \quad (10)$$

To convert the difference equations to differential equations in order to implement the model in Verilog-A is important to take in consideration the approximation of the RNN inputs and hidden states and the range of the model parameters:

- $u_i = u + (1-a)h\dot{u}$
- $x_{i-1} = x - ah\dot{x}$
- $x_i = x + (1-a)h\dot{x}$
- $0 < a < 1$
- $h > 0$, where h is the RNN time-step
- Ω_0

Finally, combining equations (9) and (10) with the approximations of the RNN is possible to reach the differential equations (11), (12) and (13).

$$x + (1-a)h\dot{x} = \sigma[W_r(x - ah\dot{x}) + W_u(u + (1-a)h\dot{u}) + \sigma^{-1}(\Omega_0)] - \Omega_0 \tag{11}$$

$$y = W_y x \tag{12}$$

$$i[1{:}n] = x - (1-a)h\dot{x} - \sigma[W_r(x - ah\dot{x}) + W_u(u - (1-a)h\dot{u}) + \sigma^{-1}(\Omega_0)] + \Omega_0 \tag{13}$$

Using these equations is possible to build the Verilog-A system present in Figure 2.16.

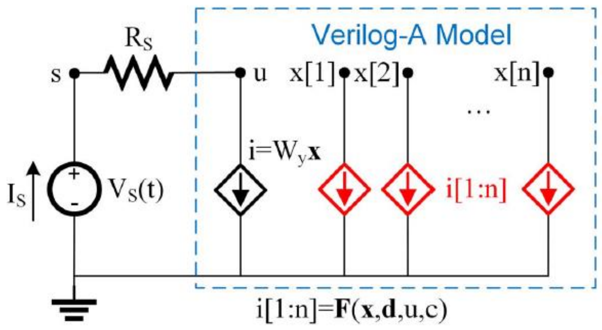


Figure 2.16 - Verilog-A Model (from [24])

It is important to mention that when the "a" parameter is close to 1 it affects the numerical stability of the model (it can affect the convergence of the model). To solve this problem is proposed the use of a time-step for the transient simulation that is 1/10 of the time-step of the RNN. As a case study, to prove the usefulness of this approach, is used an Active Rail Clamp Circuit with 6 transistors and 2 passive elements in a 130 nm CMOS. The data is acquired through physical circuit simulation, and the model is trained using the gradient descent algorithm. For evaluate the model, it is converted to Verilog-A and is used an unseen and more complex stimulus, a piecewise-linear voltage source with 1Ω of output resistance and a waveform generated by an IEC61000-4-2 ESD tester. In Figure 2.17 and Figure 2.18 is presented the respective results.

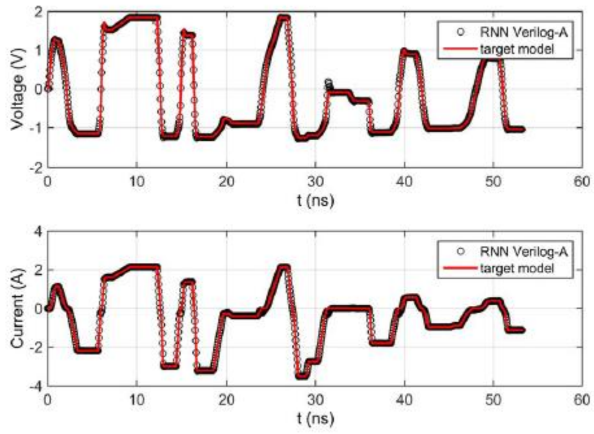


Figure 2.17 - PWL Results (from [24])

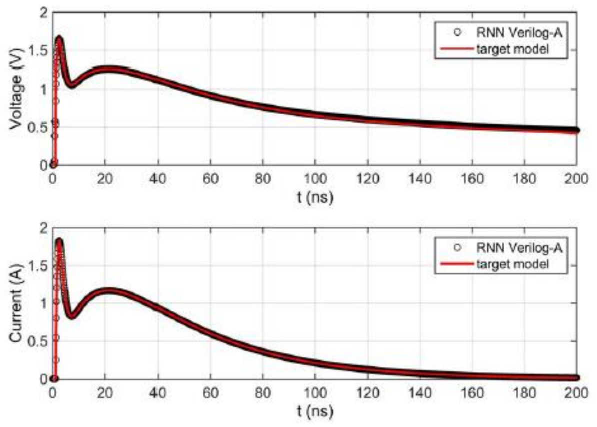


Figure 2.18 - IEC Results (from [24])

The voltage error is 1.39% for PWL and 1.91% for IEC. The current error is 0.77% for PWL and 0.20% for IEC.

2.4 Conclusions

Modeling circuits behavior is a task that ANNs can perform successfully, and it brings to the analog IC design automation many advantages: simulations are faster, less computational complexity is required, and the models can be generalized. Apart from ANNs, SVMs have also been used for behavioral modeling. Still, due to the reasons detailed in section 2.2, ANN is the method chosen to use in this work.

The major drawback of the previously mentioned works is the lack of generalization when applying the created model to other circuits topologies. That is what the proposed work will peruse, giving a sense of structure to the data through the GNN.

To perform an extensive circuit analysis is important to convert the model to a hardware language such as Verilog-A. Table 2.1 presents a summary of the state-of-art in converting artificial neural networks to Verilog-A language which is an essential task to test the model performance.

Table 2.1 Neural Networks in Verilog-A Summary

Reference	Neural Network Type	VHD Language
[23]	Backpropagation	Verilog-A
[24]	RNN	Verilog-A

Table 2.2 - Modeling of Analog/RF Devices Summary

Ref.	Device/Circuit	Method(s)	Objective	HDL	Error (circuit vs model)	Time (time reduction)	Data (simulator / dataset [in], [out])
[4]	RF- microwave components and MESFET	ANN (several)	Review of ANN based CAD for microwave designs.	-	-	-	Spectre / [v_{gs}, V_{ds}, f], [Drain Current]
[7]	Three-stage cross-coupled charge pump	ANN (MLP)	Model a charge Pump circuits	Verilog-AMS	4.85%	7s	Spice / [$v_{dd}, R_{load}, f_{clk}$], [Output Voltage]
[10]	Low Relaxation Oscillator	ANN (TDNN)	Model to analyze the Power consumption	SystemVerilog	2.70%	2min	Spectre / [f, clk, EN], [Supply Current]
[11]	Analog-n/d	ANN (Bprop)	Modeling the power consumption.	Verilog-XL	1.53%	-	- / [f, V_{out}], [Power]
[12]	RF Receiver, Sigma Delta, Low-Dropout Regulator	ANN (RNN) + Genetic Algorithm	Surrogate Model with Collaborative Stimulus	Verilog-A	RMSE: 31μV	30* Faster	- / [Stimulus], [Adapted Stimulus]
[13]	CMOS band-gap voltage reference circuit (BGR)	CompNN (NARX + TDNN)	Capture the dynamic of analog MIMO circuits	Verilog-A	5.00%	17* Faster	- / [Trimming, Load Jump, Line Jump], [1V-Out]
[15]	Inverter Ring Oscillator	ANN (FFNN)	Capture the periodicity of oscillator output waveform	Verilog-A	-	10* Faster	Spectre / [Time Step, f_{osc}], [Output Voltage]
[16]	Ring Oscillator with a Buffer	ANN (FFNN + RNN)	Capture the periodicity of the oscillator + buffer output waveform	Verilog-A	RMSE: 0.0034	10* Faster	Spectre / [Time Step, f_{osc}], [Output Voltage]
[17]	Transistor-Level VCO Circuit	ANN (2 FFNN + RNN)	Model the oscillatory frequency and see the effects on the output waveform.	Verilog-A	RMSE: 0.0061	10* Faster	Spectre / [Time Step, $f_{osc}, V_{controller}$], [Output Voltage]
[18]	RF-microwave components,HMT and MESFETs	ANN (several)	Review of model development and nonlinear modeling of microwave devices	-	-	-	-
[19]	Coplanar Waveguide Circuits	ANN (EM based)	Efficient modeling of CPW components for accurate performance estimations	-	0.16%	-	- / [f, W, G], [Line Impedance]
[20]	UC-PGB rectangular waveguide, patch antenna with PGB substrate	ANN (MLP and RBF)	Efficient modeling of RF devices for nonlinear microwave applications.	-	MSE: 2*10^{-4}	-	- / [FET Length, FET Width], [Drain Current]
[21]	SCFL Buffer Cell, Resistive Mixer, Voltage Controller Oscillator	SVM ($\varepsilon - SV\ regression$)	Robust modeling of GaAs transistors and circuits.	-	MSE: 3*10^{-7}	-	- / [V(n), I(n-k)], [Output Current]
[22]	Analog Circuits- CMOS	SVM	Efficient active learning scheme for feasible design space selection	-	-	-	Spice / [Heigth, Width]

Chapter 3

Proposed Work

The work proposes the development of a behavioral model to capture the dynamics of different topologies of circuits. As an innovation, this work will pursue, for the first time in the literature, a single LSTM model that can be reused among several circuits topologies. Such a model is a significant step forward when comparing with previous approaches since it will make the data structured, and the model itself can be generalized for different topologies, allowing to have one model capable to mimic the behavior of multiple circuits. A set of amplifiers topologies will be used as the working example in this work and the finalized model can be used to accelerate the design optimization of an analog front end circuit for biomedical applications [25]. The model's general idea follows similar concepts of [26].

3.1 Overview

The main objective of this work is, given different amplifier circuits, with different topologies, to model the behavior of any of that circuits, using only one ANN model (RNN or LSTM). To achieve that higher generalization level is proposed to explore a Graph Neural Network (GNN) to perform topology-specific feature encoding. This methodology introduces a sense of structure in the input data, drastically reducing the total number of trainable parameters, turning into a regression problem with smaller dimensions and more efficient. Figure 3.1 illustrates a simple design flow of the proposed work.

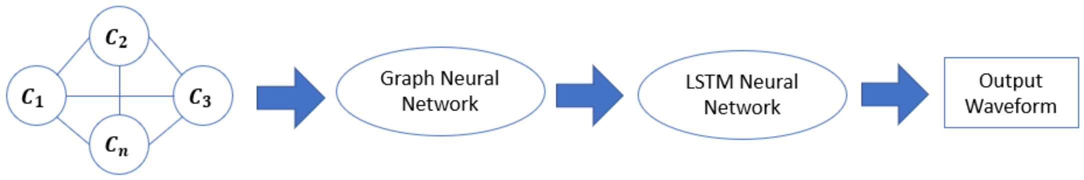


Figure 3.1 - Proposed Architecture

3.2 Data Acquisition and Organization

The data is acquired by SPICE simulation of the circuits and organized into a dataset. To obtain the dataset, the input parameters are the ones that serve as inputs of the circuit, and they are swept to obtain a satisfactory representation of the circuit behavior, and the output is the correspondent waveform for that set of inputs.

The proposed modeling methodologies (not the models) are intended to be general and not circuit class dependent. Still, in this exploratory work, a set of amplifier topologies will be used as the working examples. The choice to use these circuits was made due to:

- Amplifiers are widely studied, providing solid baselines for evaluating and debugging the generated models.
- They are readily available at ICG-Lx (the research group from Instituto de Telecomunicações where this work is being conducted) and were used to prove the established circuit sizing and optimization tool, AIDA [27], and more recently, the DeepPlacer [28] tool. Both developed and maintained at ICG-Lx.
- Moreover, the implemented modeling techniques can easily be integrated into a comprehensive flow, in other words it can be integrated in a larger system.

These topologies are implemented in several technology nodes ranging from 350-nm down to 65-nm, and multiple specification tradeoffs for each topology are also available.

To allow the model to be generic for the topology of amplifiers is important to keep the data organized as a graph. Each node represents the devices, and the edge represents the wires that connect the devices. An adjacency matrix is built accounting for the number of connections between devices as proposed in [29].

After acquiring the data, some feature engineering is performed, such as removing the *null* values, filtering and normalizing the adjacency matrix rows, and normalizing the input features.

3.3 Architecture Of The ANN Models

Once the data is well worked is time to define the ANN models. It is also essential to explore some negative aspects of each method in order to explain the proposed solutions to overcome them, and, make the proposed work more efficient and generic. The main networks that are expected to be used in this work are presented in the following subsections.

3.3.1 Graph Neural Network

Graph Neural Networks (GNN) are ANNs that map graphs to vector representations, applying operations on node features [30]. Although it is important to mention that once the GNN generates features for the LSTM is essential to avoid permutations in the adjacency matrix, so it needs to be permutation invariant. To achieve that is useful to sum all the features in all the nodes in the dataset and return a vector of the sum of each node. So Convolutional Graph Neural Network, where the model learns by inspecting the neighbor nodes in the graph, is considered. The GCN, introduced in the DL area by Kipf and Welling [31], relies on message-passing methods. Each node has information about the neighbor nodes and exchanges messages with them. In a mathematical approach, let us consider A the adjacency matrix and H^L the hidden layers. A simple layer propagation H^{L+1} is given by equation (14), where the propagation comes from multiplying the adjacency matrix by the hidden layer and the correspondent weight matrix.

$$H^{L+1} = \sigma[AH^L W^L] \tag{14}$$

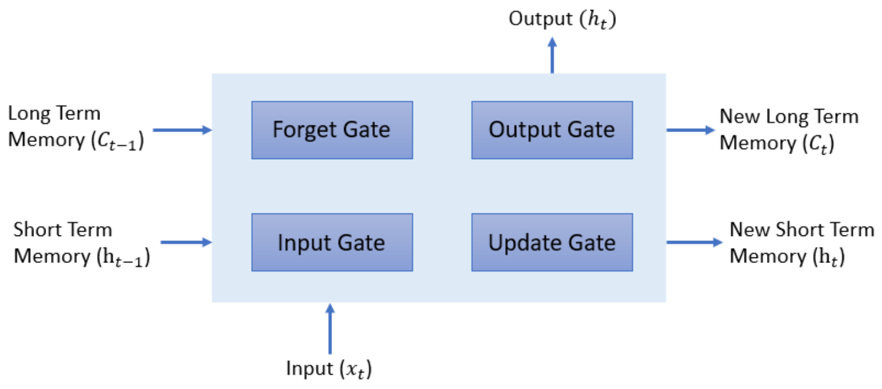
One of the significant limitations of the model in equation (14) is the scale change of the feature vectors when multiplying with the adjacency matrix A. To avoid that problem is performed a Graph Filter to remove redundant and negative edges from the graph, and then a Row Normalization is performed. This normalization is performed such as all the rows of A sum up to one, to attain that goal is used a diagonal node degree matrix $\widehat{D}$, which has in the principal diagonal the number of connections of each node. The matrix A also suffer alterations. It is added to A the identity matrix so that a message sent by a node to the others in the neighborhood can also be received by itself. Equation (15) shows the improvement of the previous layer propagation with graph row normalization [31].

$$H^{L+1} = \sigma[(\widehat{D}^{-\frac{1}{2}} A \widehat{D}^{-\frac{1}{2}}) H^{L} W^{L}] \tag{15}$$

Thus, equation (15) gives the equation of a convolutional graph layer, now it is just needed to apply as many as required on the specific problem. A critical aspect of the GCN is that the insertion of the matrix A the model can learn features of the neighbor nodes and can learn features representations even before training is performed, in other words, evaluating the model with devices unseen and not included in the training set the error remains very low, as concluded in [26], showing the generalization potential of this model.

3.3.2 “Long Short-Term Memory” Neural Network

As Figure 3.1 outlines, the output of the GNN is the input of the LSTM neural network. First, this model is chosen, to the detriment of the RNN, since the main disadvantage of the RNN model is that it only “remembers” the previous output and for this work is essential to keep track of longer outputs history. Figure 3.2 is enlightened the main blocks of an LSTM network.



Output (h_t)
Long Term Memory (C_{t-1})
Forget Gate
Output Gate
New Long Term Memory (C_t)
Short Term Memory (h_{t-1})
Input Gate
Update Gate
New Short Term Memory (h_t)
Input (x_t)

Figure 3.2 - LSTM Neural Network Model

Figure 3.3 illustrates the interior of an LSTM model mathematically formulated by equations in (16).

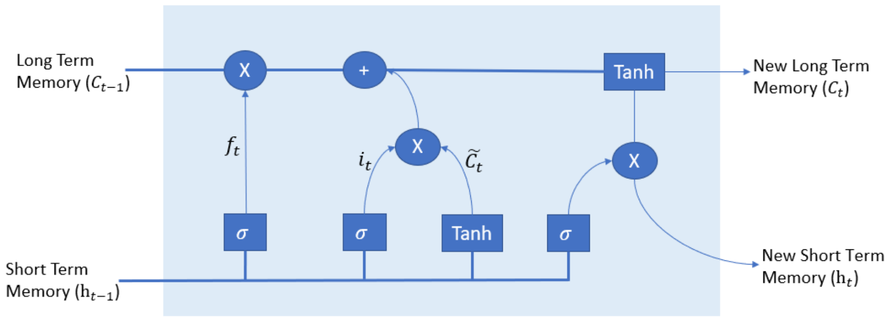

Long Term Memory (C_{t-1})
X
+
Tanh
New Long Term Memory (C_t)
f_t
i_t
X
$\tilde{C}_t$
X
σ
σ
Tanh
σ
Short Term Memory (h_{t-1})
New Short Term Memory (h_t)

Figure 3.3 - LSTM Interior

$$\begin{gathered} h_t = \sigma(w_o[h_{t-1}, x_t] + b_o) * \tanh(C_t) \\ C_t = f_t * C_{t-1} + \tilde{C}_t * i_t) \\ f_t = \sigma\left(w_f[h_{t-1}, x_t] + b_f\right) \\ i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \\ \tilde{C}_t = tanh(w_c[h_{t-1}, x_t] + b_c) \end{gathered} \tag{16}$$

It is essential to know how the LSTM behaves and its positive aspects. However, it is also critical to be aware of its drawbacks and overcome them. The RNN or the LSTM can suffer the following problems:

Vanish Gradients

Gradients update the weights and bias when training the LSTM neural network. When the activation function is a sigmoid or hyperbolic tangent in the backpropagation, its derivatives are multiplied, and the gradient can decrease exponentially if the derivatives are very small, making the neural network not update the values of the weights in the initial layers. To overcome this issue the literature suggests making use of Rectified Linear Unit (ReLu) , "Leaky" ReLu and "Exponential Linear Unit" as activation function, normalize each batch using its mean and standard deviation and different weights initializations such as Xavier Initialization where the bias is initialized with 0 and the weights initialization depends on a uniform distribution and the size of the previous layer [32].

Gradient Explosion

In backpropagation, the derivatives are multiplied, and the gradient can increase exponentially if the derivatives are high, making the neural network training nonconvergent. "Gradient Clipping" means that the gradients are cut off before reaching a predetermined limit. It effectively prevents the gradients to reach higher values and cause gradient explosion.

3.4 Architecture of the Global Model

Figure 3.4 represents the proposed GCN-LSTM model that merges both GCN and LSTM models, which will be used in the practical implementation of this work.

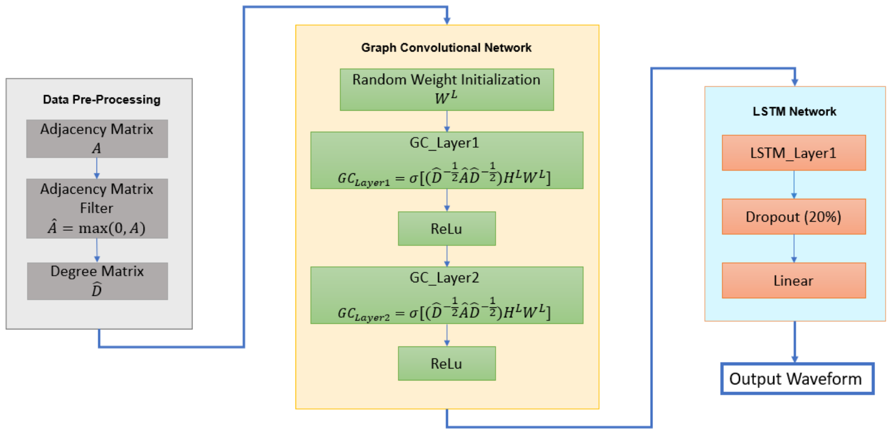
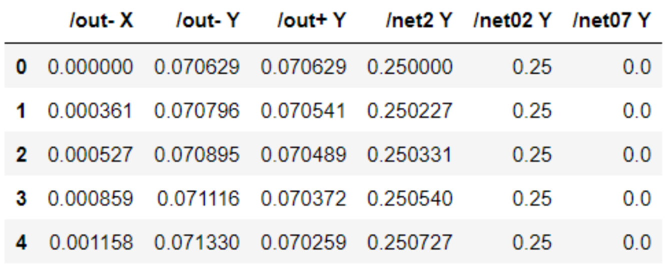

Data Pre-Processing
Adjacency Matrix A
Adjacency Matrix Filter $\hat{A} = \max(0, A)$
Degree Matrix $\hat{D}$
Graph Convolutional Network
Random Weight Initialization W^L
GC_Layer1
$GC_{Layer1} = \sigma[(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}})H^L W^L]$
ReLu
GC_Layer2
$GC_{Layer2} = \sigma[(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}})H^L W^L]$
ReLu
LSTM Network
LSTM_Layer1
Dropout (20%)
Linear
Output Waveform

Figure 3.4 - GCN-LSTM Network

In the preprocessing phase, the adjacency matrix, with the relation between the input device's ports, is cleaned, removing the redundant and negative values, and the Degree Matrix is obtained, which contains the number of connections of each device' port. In the graph convolutional network, the weights are initialized randomly, and the GC layer is built considering the row normalization of matrix A. Following are added two convolutional layers in the network with ReLu activation functions. The GCN output enters the LSTM layer, followed by a dropout layer to prevent the overfitting and then a linear layer to produce the output that is the waveform of the circuit in the time domain. It is important to mention that the current architecture it is just an initial draft, and it might suffer several adjustments until more efficient results are produced and the overfitting effect avoided. Finally, the output waveform of the GCN-LSTM network will be compared with circuit simulation, obtaining metrics such as the MSE, and conclusions will be made on its results. Posteriorly the model will be implemented in Verilog-A.

3.5 Preliminary Models and Results

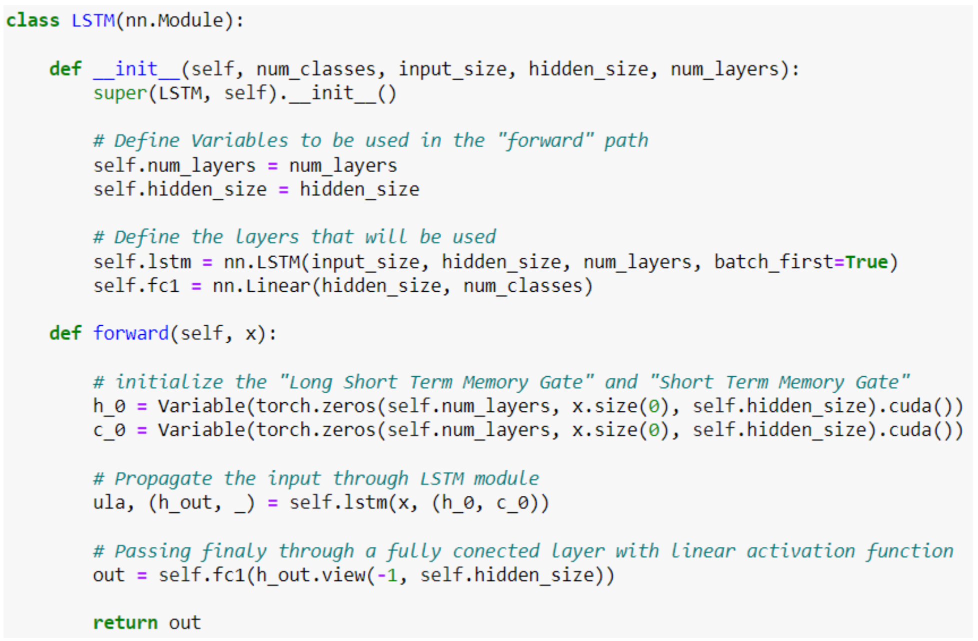
In a first approach to the practical work, the dataset present in Figure 3.5 is obtained, where "/out- X", "/net2 Y", "/net02 Y", "/net07 Y" are the input parameters of the circuit and, "/out- Y" and "/out+ Y" are the output waveforms of the circuit, the ones that will try to be mimicked by the network.



	/out- X	/out- Y	/out+ Y	/net2 Y	/net02 Y	/net07 Y
0	0.000000	0.070629	0.070629	0.250000	0.25	0.0
1	0.000361	0.070796	0.070541	0.250227	0.25	0.0
2	0.000527	0.070895	0.070489	0.250331	0.25	0.0
3	0.000859	0.071116	0.070372	0.250540	0.25	0.0
4	0.001158	0.071330	0.070259	0.250727	0.25	0.0

Figure 3.5 - Dataset

A simple model capable of modeling this circuit's behavior is an LSTM neural network. Before giving the data to the network is important to apply a sliding window on the data to create sets of inputs and the correspondent output considering the temporal flow of the data. After that, the data is divided into train, validation, and test to make a proper training and evaluation of the model. Before sending the data to the network, the last step is to scale the data by the *MinMaxScaler*. The LSTM model, in Figure 3.6, coded in python language and using PyTorch [33] ML framework, consists of 1 LSTM layer followed by a fully-connected layer with a linear activation function. The loss criterion is the MSE, and Adam is the optimizer for training, with a learning rate of 0.01 and betas of 0.9 and 0.999. The input size is the *num_features*, in this case, is 4, the *num_classes* is the number of outputs, which is 2. The *hidden_size* was set to 100 hidden neurons.



```python
class LSTM(nn.Module):

    def __init__(self, num_classes, input_size, hidden_size, num_layers):
        super(LSTM, self).__init__()

        # Define Variables to be used in the "forward" path
        self.num_layers = num_layers
        self.hidden_size = hidden_size

        # Define the layers that will be used
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc1 = nn.Linear(hidden_size, num_classes)

    def forward(self, x):

        # initialize the "Long Short Term Memory Gate" and "Short Term Memory Gate"
        h_0 = Variable(torch.zeros(self.num_layers, x.size(0), self.hidden_size).cuda())
        c_0 = Variable(torch.zeros(self.num_layers, x.size(0), self.hidden_size).cuda())

        # Propagate the input through LSTM module
        ula, (h_out, _) = self.lstm(x, (h_0, c_0))

        # Passing finaly through a fully conected layer with linear activation function
        out = self.fc1(h_out.view(-1, self.hidden_size))

        return out
```

Figure 3.6 - LSTM Model (PyTorch [33] implementation)

The model is trained using 500 epochs. Once the model is trained, it is appropriately tuned using the validation set. Once it fits the desired results, avoiding overfitting, the test set is applied to provide the results shown in Figure 3.7. From them, it is possible to see that the result, predicted by the model, can mimic the circuit behavior. Nonetheless, further tuning has to be conducted in order to output a predicted waveform that better fits the circuit simulation waveform. To have some quantitative metrics, the MSE obtained for “/out- Y” is 1.27106×10^{-5} and for “/out+ Y” is 1.27065×10^{-5}. Related to the Graph neural network as preliminary results, the following illustration shows how to, given a circuit example, extract its graph and adjacency matrix. Figure 3.8 presents the circuit schematic of the simple amplifier considered. It is possible to observe that it has p-type and n-type metal–oxide–semiconductor field-effect transistors (MOSFETs). The respective graph, in Figure 3.9, shows as nodes the circuit devices, the PMOS and NMOS, and the edges are the wire connections between nodes. Finally, having the graph, the adjacency matrix is obtained directly from the number of wire (node to node) connections. For the presented graph, the adjacency matrix is shown in Figure 3.10.

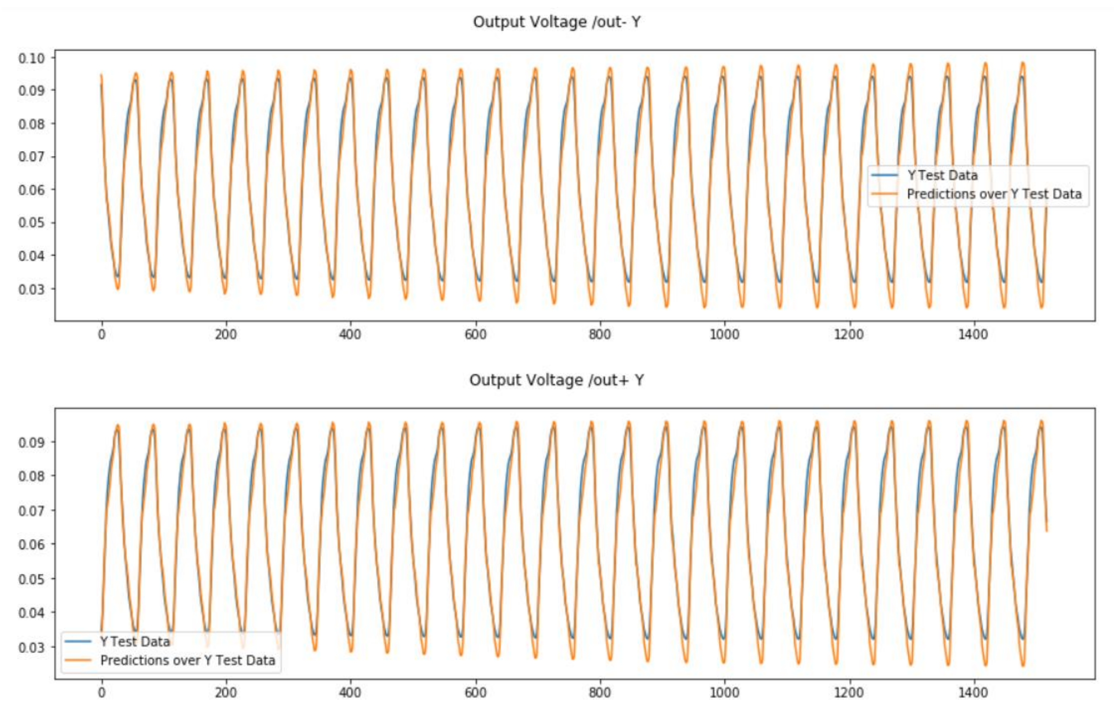


Figure 3.7 - Preliminary Results

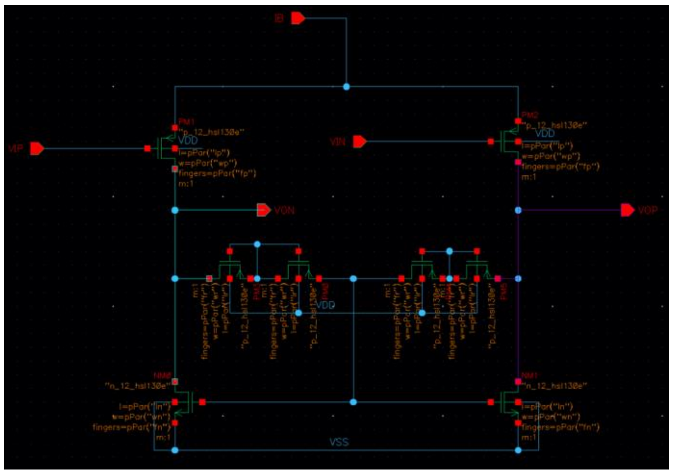


Figure 3.8 - Circuit Example

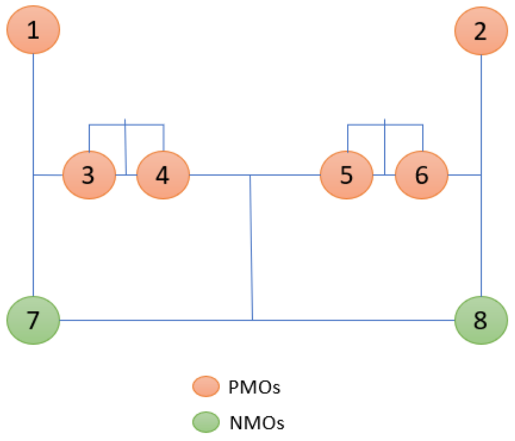


Figure 3.9 - Circuit Graph Example

	1	2	3	4	5	6	7	8
1	0	0	1	0	0	0	1	0
2	0	0	0	0	0	1	0	1
3	1	0	1	1	0	0	1	0
4	0	0	1	1	1	0	1	1
5	0	0	0	1	1	1	1	1
6	0	1	0	0	1	1	0	1
7	1	0	1	1	1	0	0	1
8	0	1	0	1	1	1	1	0

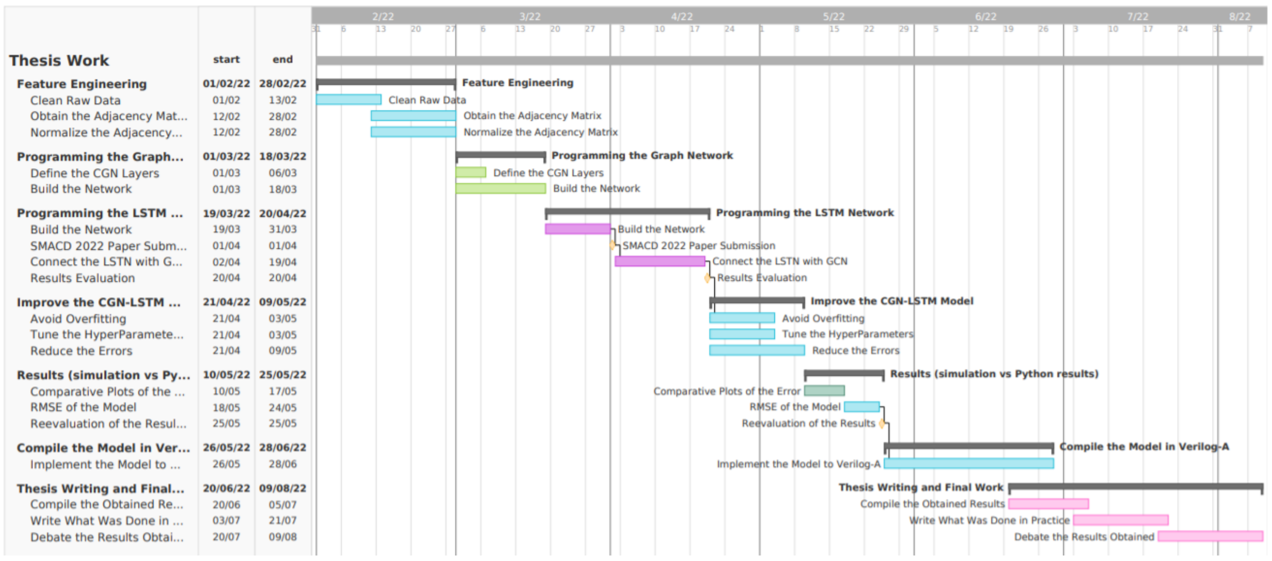
Figure 3.10 - Adjacency Matrix

Note that the procedure illustrated here will be automatically performed. These figures only intend to demonstrate the intuition behind the proposed work for the graph preprocessing and show the adjacency matrix that will be, among others, the input of the GCN.

3.6 Work Planning

The practical implementation of this work will follow the structure mentioned above, considering the vantages and drawbacks of the methods introduced and considering all the careful steps to take when using ANNs models. Finally, for the Master Thesis is important to have a detailed plan to follow in order to accomplish all the deadlines for this work. Therefore, Figure 3.11 presents a Gantt Chart with the proposed plan. The practical implementation of the proposed work will start as soon as there is formal consent by the jury.

First, careful feature analysis is performed on the raw data obtained from the circuit simulation in order to obtain cleaned and structured data for the model input. In the second stage, the GCN and the LSTM are implemented and merged to work as a block. At the end of the second stage, a results evaluation is performed to check if the work is flowing correctly. Then the model results are extracted and compared with the simulation and the MSE is obtained. Once the results satisfy the objectives, the model is implemented in Verilog-A and it can be used in AIDA for the optimization of a complete analog front end circuit. In the final stage, the thesis is written, introducing the results obtained and the methods and approaches taken to obtain them. It is also worth to mention that this work will pursue a scientific publication in SMACD 2022 Conference of the preliminary findings. The results are discussed, and the work conclusion is performed.



Thesis Work	start	end
Feature Engineering	**01/02/22**	**28/02/22**
Clean Raw Data	01/02	13/02
Obtain the Adjacency Mat...	12/02	28/02
Normalize the Adjacency...	12/02	28/02
Programming the Graph...	**01/03/22**	**18/03/22**
Define the CGN Layers	01/03	06/03
Build the Network	01/03	18/03
Programming the LSTM ...	**19/03/22**	**20/04/22**
Build the Network	19/03	31/03
SMACD 2022 Paper Subm...	01/04	01/04
Connect the LSTN with G...	02/04	19/04
Results Evaluation	20/04	20/04
Improve the CGN-LSTM ...	**21/04/22**	**09/05/22**
Avoid Overfitting	21/04	03/05
Tune the HyperParamete...	21/04	03/05
Reduce the Errors	21/04	09/05
Results (simulation vs Py...	**10/05/22**	**25/05/22**
Comparative Plots of the ...	10/05	17/05
RMSE of the Model	18/05	24/05
Reevaluation of the Resul...	25/05	25/05
Compile the Model in Ver...	**26/05/22**	**28/06/22**
Implement the Model to ...	26/05	28/06
Thesis Writing and Final...	**20/06/22**	**09/08/22**
Compile the Obtained Re...	20/06	05/07
Write What Was Done in ...	03/07	21/07
Debate the Results Obtai...	20/07	09/08

2/22
3/22
4/22
5/22
6/22
7/22
8/22
Feature Engineering
Clean Raw Data
Obtain the Adjacency Matrix
Normalize the Adjacency Matrix
Programming the Graph Network
Define the CGN Layers
Build the Network
Programming the LSTM Network
Build the Network
SMACD 2022 Paper Submission
Connect the LSTN with GCN
Results Evaluation
Improve the CGN-LSTM Model
Avoid Overfitting
Tune the HyperParameters
Reduce the Errors
Results (simulation vs Python results)
Comparative Plots of the Error
RMSE of the Model
Reevaluation of the Results
Compile the Model in Verilog-A
Implement the Model to Verilog-A
Thesis Writing and Final Work
Compile the Obtained Results
Write What Was Done in Practice
Debate the Results Obtained

Figure 3.11 - Thesis work Plan

References

[1] Martins, R., Horta, N. and Lourenço, N., “Automatic Analog IC Sizing and Optimization Constrained” (1st ed.), Springer, 2017.

[2] Gielen, G. G. E. and Rutenbar, R. A. (2000) “Computer-Aided Design of analog and Mixed-Signal Integrated Circuits,” Proceedings of the IEEE, pp. 1825-1854.

[3] Jordan, M. I. and Mitchell, T. M. (2015) , “Machine learning: Trends, perspectives, and prospects.” Science, 349(6245), 255–260.

[4] Zhang Q. and Vijay K. D. (2003). “Artificial Neural Networks for RF and Microwave Design—From Theory to Practice”. IEEE Transactions on Microwave Theory and Techniques, Vol. 51.

[5] Miller, I., FitzPatrick, D., & Aisola, R. (1997), “Analog design with Verilog-A”. Proceedings of Meeting on Verilog HDL.

[6] Bose S., Shen B., Johnstion M. (2020). “A Batteryless Motion-Adaptive Heartbeat Detection System-on-Chip Powered by Human Boady Heat”. IEEE Journal of Solid-State Circuits, Vol. 55, No.11.

[7] Grabmann, M., Landrock, C. and Gläser, G. (2021), “Machine Learning in Charge: Automated Behavioral Modeling of Charge Pump Circuits”. SMACD / PRIME 2021.

[8] Goodfellow I., Bengio Y., Courville A. (2016) Deep Learning, MIT Press.

[9] Kingma D. P., Ba J. (2014). “Adam: A Method for Stochastic Optimization”. In: CoRR, abs/1412.6980.

[10] Grabmann, M., Landrock, C. and Gläser, G. (2019), “Power to the Model: Generating Energy-Aware Mixed-Signal Models using Machine Learning”. SMACD 2019, Lausanne, Switzerl.

[11] A. Suissa, et al. (2010). “Empirical method based on neural networks for analog power modeling”. IEEE Trans. Comput. Aided Des. Integrated Circ. Syst. 29 (5) 839–844.

[12] Lei, J.Y and Chatterjee, A. (2021), “Automatic Surrogate Model Generation and Debugging of Analog/Mixed-Signal Designs Via Collaborative Stimulus Generation and Machine Learning”. 26th Asia and South Pacific Design Automation Conference.

[13] Hasani, R. M., Haerle, D., Baumgartner, C. F., Lomuscio, A. R. and Grosu, R. (2017). “Compositional neural-network modeling of complex analog circuits”. International Joint Conference on Neural Networks.

[14] Xie H., Tang H., Liao Y-H. (2009). “Time Series Prediction based on NARX neural networks: An advanced approach”. 2009 International Conference on Machine Learning and Cybernetics.

[15] Yu, H., Swaminathan, M., Ji, C. and White, D. (2017). “A method for creating behavioral models of oscillators using augmented neural networks”. Conference on Electrical Performance of Electronic Packaging and Systems.

[16] Yu, H., Swaminathan, M., Ji, C. and White, D. (2018). “Behavioral Modeling of Steady-State Oscillators with Buffers Using Neural Networks”. Conference on Electrical Performance of Electronic Packaging and Systems.

[17] Yu, H., Swaminathan, M., Ji, C. and White, D. (2018). “A Nonlinear Behavioral Modeling Approach for Voltage-controlled Oscillators Using Augmented Neural Networks”. IEEE/MTT-S International Microwave Symposium - IMS.

[18] Vijay K. D., Mustapha C. E., Yonghua F., Jianjun X., Zhang Q. (2001). “Neural Networks for Microwave Modeling: Model Development Issues and Nonlinear Modeling Techniques”. Int. J. RF Microw. Computer-Aided Eng.

[19] Watson P.M., Gupta K.C. (1997). “Design and optimization of CPW circuits using EM-ANN models for CPW components”. IEEE Trans. Microw. Theor. Tech. 45 (12) 2515–2523.

[20] Passos M.G., Silva P.d.F., . Fernandes H.C. (2006). “A RBF/MLP modular neural network for microwave device modeling”. Int. J. Comput. Sci. Netw. Secur. 6 (5A) (2006) 81–86.

[21] Ceperic V., Baric A. (2004). “Modeling of analog circuits by using support vector regression machines”. 11th IEEE International Conference on Electronics, Circuits and Systems.

[22] Ding M., Vemuri R. (2005). “An active learning scheme using support vector machines for analog circuit feasibility classification”. 18th International Conference on VLSI Design held Jointly with 4th International Conference on Embedded Systems Design, IEEE, 2005, pp. 528–534.

[23] Suzuki K., Nishio A., Kamo A., Watanabe T. and Asai H., (2000). “An application of Verilog-A to modelling of back propagation algorithm in neural networks”. 43rd IEEE Midwest Symposium on Circuits and Systems.

[24] Chen Z., Raginsky M., Rosenbaum E. (2017). “Verilog-A Compatible Recurrent Neural Network Model for Transient Circuit Simulation”. IEEE 26th Conference on Electrical Performance of Electronic Packaging and Systems.

[25] Kosari A., Breiholz J., Liu N., Calhoun B., Wentzloff D. (2018). “A 0.5V 68nW ECG Monitoring Analog Front-End for Arrhythmia Diagnosis”. Journal of Low Power Electronics and Applications.

[26] Gusmão A., Horta N., Lourenço N., Martins R. (2021). “Bringing Structure into Analog IC Placement with Relational Graph Convolutional Networks”. SMACD / PRIME 2021; International Conference on SMACD and 16th Conference on PRIME.

[27] https://www.aidasoft.com/Home

[28] Gusmão A., Horta N., Lourenço N., Martins R., Póvoa R. (2021). “DeepPlacer: A costum integrated OpAmp placement tool using deep models”.

[29] Wang H., Wang K., Yang J., Shen L., Sun N., Lee H. S., Han S. (2020). “GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning”. 57th ACM/IEEE Design Automation Conference (DAC).

[30] Niu C., Song Y., Song J., Zhao S., Grover a., Ermon S. “Permutation Invariant Graph Generation via Score-Based Generative Modeling”.

[31] Kipf T. N., Welling M. (2017) “Semi-Supervised Classification with Graph Convolutional Networks”. International Conference on Learning Representations 2017.

[32] Xavier G., Bengio Y. “Understanding the difficulty of training deep feedforward neural networks”.

[33] https://pytorch.org/