Exploring Machine Learning for Electrical Behavior Prediction: The CMOS Inverter Case Study

Gabriel Lima Jacinto, Lucas Yuki Imamura, Mateus Grellert and Cristina Meinhardt
Department of Informatics and Statistics, *Federal University of Santa Catarina*
gabriellimajacinto@gmail.com, lucasyuki@yahoo.com.br, mateus.grellert@gmail.com, cristina.meinhardt@ufsc.br

***Abstract*—With the advancement of integrated circuit manufacturing technology, a growing number of aspects must be considered during the electrical characterization of circuits in order to solve challenges such as the effect of process variability. This increases the characterization time due to the use of techniques based on exhaustive electrical simulations. Machine learning techniques are consistently being employed to assist digital design at many levels of abstraction with various successful applications. Thus, the main objective of this work is to evaluate machine learning regression algorithms as an alternative to exhaustive electrical simulation in the cell characterization project. In this step, multiple linear regression, support vector regression, decision trees, and random forest algorithms are considered. This work presents the results of a first case study: an Inverter using bulk CMOS technology. Specifically, the energy values and propagation times of this circuit will be separately predicted. A comparative analysis is done for each dependent variable between the models in order to understand which is the best regression model for the task. The algorithm with the lowest cost function proved to be Random Forests, with a R^2 above 98% for all predicted variables.**

***Index Terms*—Cell characterization, Power, Delay, Machine Learning**

I. Introduction

The evolution of the manufacturing process of transistors allows an increase in the ability to integrate features in the same project, with better performance results due to the transistors being faster, consuming less energy, and occupying less circuit area. However, this evolution also introduces new challenges, especially when designing integrated circuits in nanometer technologies. Process variability is one of the biggest challenges in nanometer technologies, affecting the expected normal behavior of cells, and altering the delay and energy consumption observed under standard conditions. Along with the effects of process variability, there is an increase in the complexity of design rules. Consequently, integrated circuit design tools need to solve increasingly difficult problems. This problem has been growing exponentially, and these difficulties directly impact the cost of developing electronic devices [1].

To better represent process variability in current fabrication technologies, the characterization of cells started to include several corner cases, in addition to the traditional ones that characterize the behavior of faster, slower, and in typical operating conditions. To explore a wide range of possible characterization corners in a target technology, designers traditionally adopt exhaustive electrical simulations for all corner points, observing the impact on delay metrics and power consumption. The main issue with this approach is that increasing the number of electrical simulations to cover these corners can become a time-consuming task. Machine learning (ML) techniques can be good alternatives, in the sense that they can replace the brute-force corner analysis in an attempt to reduce characterization time and costs for the development of electronic devices.

The literature shows that there is room for adopting machine learning in the development of tools in the microelectronics area. Its current use is strongly linked to predicting the quality of candidate solutions given by existing tools or trying to guide them to a better quality solution, through metric predictions [1]. With the increasing availability of large amounts of manufacturing data and process improvement data, machine learning techniques can be adopted to analyze, classify and predict the quality of the manufacturing step that etches the metals in the circuit [2] or investigate the effects of process variability in 3D NAND-type flash memory cells [3]. In the circuit positioning and routing steps, we found works that use machine learning to predict changes in the delay characterization of circuits based on path-time analysis adopting graph [4] and pre-routing time predictors [5]. In addition, in [6] machine learning techniques are explored in the electrical characterization of devices, however, specifically for magnetic devices. Thus, this indicates that there is scope for a long debate about exploring machine learning algorithms for electrical characterization of logic cells in bulk Complementary Metal Oxide Semiconductor (CMOS), Silicon On Insulator (SOI), or Fin Field-Effect (FinFET) technologies.

Observing the increasing adoption of machine learning techniques in the microelectronics area, this work seeks to employ these techniques as an alternative to the traditional methods of simulating the effects of radiation failures and process variability. Thus, this work aims to explore machine learning techniques to predict the electrical behavior of circuits. A first case study investigates regression techniques to predict the energy and the propagation time of CMOS Inverter circuits. The main contribution of this work is the evaluation of different regression models in a case-study using the Inverter gate for predicting energy and propagation time variations.

This work is organized as follows: Section II discusses the basic theoretical framework of the machine learning al-

978-1-6654-8128-1/22/$31.00

gorithms that were used, together with the explanation of the mathematical calculations that are made in the predictions of the models. Then, in Section III, it is explained the methods used to extract the data to train the algorithms, as well as the techniques applied in their modeling to avoid overfitting problems, and, to improve the performance of the models based on the chosen cost function. Section IV discusses the training results through both the analysis of the prediction errors and the more intuitive graphical representation of these errors. Finally, the entire process performed and the results obtained are summarized in a conclusion that indicates which was the best algorithm and the worst algorithm candidate for electrical characterization prediction of an Inverter.

II. Background about Machine Learning Techniques

There are several machine learning algorithms applicable for different tasks, ranging from predicting stock market values to navigating a robot on Mars [7]. The set of algorithms available depends on the presence or absence of expected (true) outcomes. The former involves supervised learning, and the latter resorts to unsupervised or reinforcement learning methods. Supervised learning is subdivided into classification, when the outcome variable assumes a finite, discrete set of values, and regression, when the outcome can assume a wide range of continuous values [8]. In this work, the objective is to predict the energy and delay of a CMOS Inverter circuit using data obtained from previous simulations, characterizing a supervised problem. Since they are all continuous values, we are also dealing with regression. There are many regression algorithms, so this analysis was limited to four classic regression methods: Multiple Linear Regression (MLR), Support Vector Regression (SVR), Decision Tree (DT), and Random Forest (RF).

A. Regression Techniques

Linear regression is a widely used statistical technique when you want to predict the behavior of a target value. As in this work, the output variables are predicted based on the values of many input variables, multiple linear regression (MLR) was used instead of simple linear regression. The predicted value $\hat{y}$ is described in Eq. 1, where θ_0 is the intercept term and θ_1 to θ_n are the weights of each value of the input variables. The objective is to find the values of each θ that minimize the cost function of the model [7].

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n \tag{1}$$

Although Support Vector Machines (SVM) are typically applied to classification, this is a versatile algorithm that can also perform regressions, called Support Vector Regression (SVR) [7]. The idea is that, given a value, the SVR calculates a margin around the hyperplane of the prediction function, where we try to fit as many observations as possible between the area defined by the margins and reduce values that are outside this same area, as shown in Fig. 1. The decision function represented by Eq. 3 is calculated similarly to the regression linear, however, here we chose to put the equation in a vectorized form, which is the most common practice when representing learning algorithms. So we have that the predicted value is represented by $\hat{y}$, w^T is the transposed matrix of weights for each input variable x (the values $\theta_1 ... \theta_n$ from Eq. 1) and b is the intercept value. The SVR has many kernels (linear, radial basis function, polynomial) which are functions that can perform the dot product $\phi(a)^T\phi(b)$ just using the values of a and b, without necessarily calculating the ϕ transformation [7]. This work adopts the radial basis function (RBF) kernel represented by the Eq. 2 [7].

$$K(x, x') = \exp -\gamma||x - x'||^2 \tag{2}$$

$$\hat{y} = w^T x + b \tag{3}$$

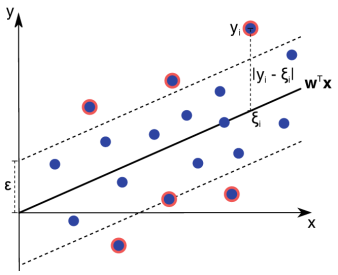

Fig. 1: Support Vector Regression

The decision tree (DT) is another algorithm most used in classification tasks. However, it is also possible to use it in regression tasks [7]. The algorithm works by creating a tree structure with internal nodes with tests that are made to predict which leaf node the resource will be assigned to, returning, in the case of this work, a continuous value. Fig. 2 shows how this algorithm performs its regression. It divides the values into sectors and tries to make the forecast based on these created sectors. These sectors are created from the function's hyperparameter configuration, for example, the maximum depth of the tree.

Random Forest (RF) is an algorithm similar to the decision trees. Fig. 3 illustrates an example of random forest prediction. The difference is that instead of computing only one decision tree, RF calculates a large number of DTs with different depths, and, then, it computes the average of all predictions to estimate the value. This makes this method one of the most powerful supervised algorithms.

B. Evaluation metrics

To assess the suitability of each model, it is necessary to assess the error between the predicted value and the desired value. In this work, the Root Mean Square Error (RMSE) and the R-squared (R^2) were used as performance measures (also called a cost function), since the former is the most commonly used measure in regression problems and the latter is more intuitive to understand. While the RMSE is calculated using the Euclidean norm, the R^2 is the total variation in the target

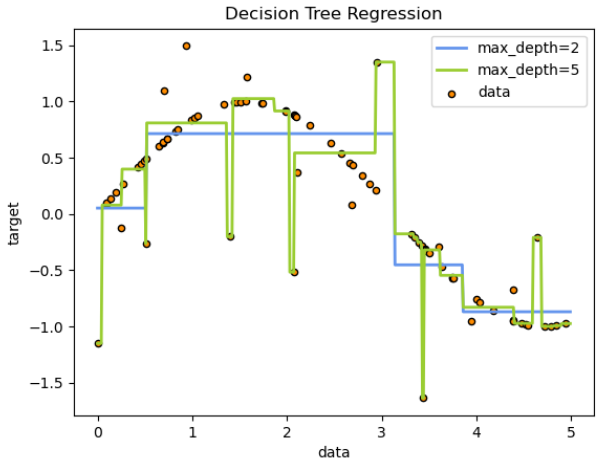

Fig. 2: Decision Tree Regression

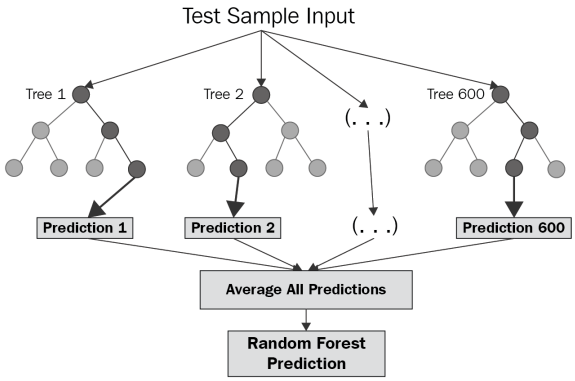

Fig. 3: Random Forest

value that is captured by the model and it ranges from 0 to 1. An R^2 score of 1 means the model was capable of explaining 100% of the modeled behavior, so low R^2 values mean the model is poorly fit and high values mean a good model fit) [9]. Furthermore, the RMSE gives higher weights for large errors, which means it is more sensitive to outliers [7], so it is important to remove these prior to training.

III. Proposed Workflow

This work proposes a workflow for characterizing an Inverter cell, as a first case study of the development of a fast predictive characterization tool based on machine learning techniques. The input variables considered were voltage, temperature, and process variability. The predicted variables are the Tp_{HL} (propagation time high-to-low, i.e., the fall delay), Tp_{LH}(propagation time low-to-high, i.e., the rising delay), and power consumption (IINT). This workflow was designed so it can be generalized to work with other circuits and technologies. Fig. 4 summarizes the training process. Thus, the process is divided into seven main steps, starting with the electrical simulations to extract the variables, followed by the treatment of these variables, performing the normalization, and also the separation of these data into training, validation, and testing sets for the evaluated algorithms. Stage four contemplates the development and adjustment of the considered algorithms. Step five performs the analysis of the algorithms. Finally, the best model is deployed in an online interface. These steps will be detailed in the next sections.

A. Feature Extraction

The electrical characterization of an Inverter circuit defines the energy and delay behavior of the circuit in different design and operation conditions. The rise and fall propagation times are used to define the delays of the circuits. These variables are influenced by several parameters of the configuration of the circuit such as transistor width, channel length, load capacitance, temperature, and operating voltage. Also, variability effects impact these variables depending on the parameters adopted.

The first step was to define clearly which input variables are relevant for the output values being modeled, which is summarized in Table I. The seven input variables simulated are the threshold voltage for NMOS and PMOS devices by the parameter $Vth0$ in the device model, the temperature, the operating voltage, the width of the transistors (W) for PMOS and NMOS devices, and the channel length (L) for PMOS and NMOS devices.

TABLE I: Input and output variables

Output Variable	Tp_{HL}	Tp_{LH}	IINT
Input Variables	NMOS $Vth0$ PMOS $Vth0$ temperature voltage W PMOS W NMOS L	NMOS $Vth0$ PMOS $Vth0$ temperature voltage W PMOS W NMOS L	NMOS $Vth0$ PMOS $Vth0$ temperature voltage W PMOS W NMOS L

All electrical simulations of this work adopted Hspice from Synopsys. In this experiment, the 16 nm High-Performance bulk CMOS transistor model, provided by PTM [10], was adopted. The parameter $Vth0$ was varied to simulate the impact of process variability due to geometric variability and random dopant variability in the transistor threshold voltage. The variability of the process is simulated using the Monte Carlo method, where the change in the threshold voltage follows a Gaussian distribution with 3σ and 10% standard deviation [11]. The width, length, temperature, and voltage values were predefined, following these range of values:

- **Temperature** (^oC): -25, 0, 50, 75, and 100
- **Width of PMOS (nm)**: 140, 70, 280, 350, 420
- **Width of NMOS (nm)**: 70 and 140
- **Length of PMOS and NMOS (nm)**: 32, 20, and 40
- **Voltage (V)**: 0.6, 0.7, 0.8, and 0.9

The values of energy and propagation times (Tp_{HL} and Tp_{LH}) were all obtained by electrical simulations based on all the derived combinational cases for the defined range of the variables. In the end, more than 1000 transient simulations of 20 ns are executed, obtaining a total of 168,000 observations.

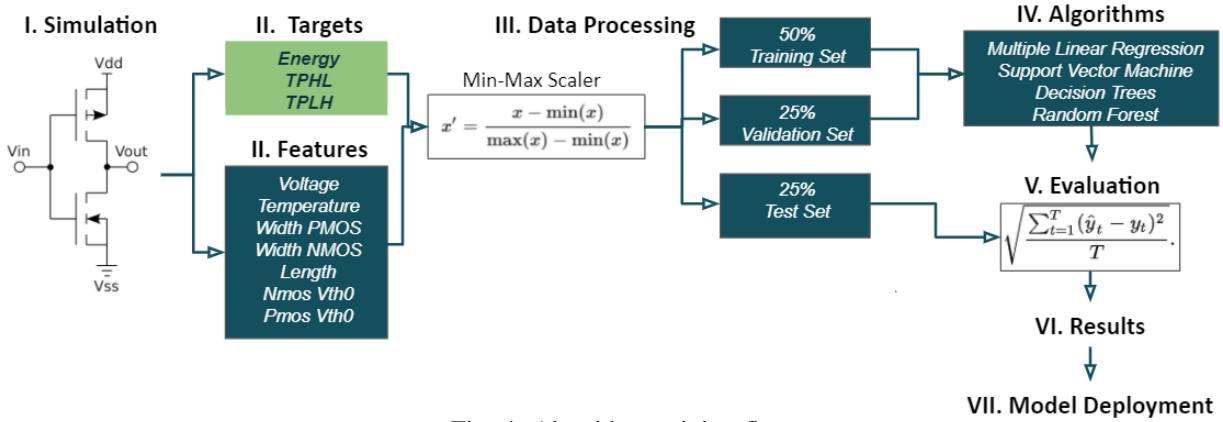

Fig. 4: Algorithm training flow

However, the data processing step removes outliers values that would confound the model results. After that, the final number of observations stood at 120,954 cases. It should be noted that these outlier values may occur due to internal errors in the electrical model used by the simulator considering edge values in the variables.

B. Normalization

The energy and delay results for the 16 nm technology are on the order of picoseconds and femtoJoules, respectively. Therefore, before training the algorithms, it is necessary to resize each variable, since, in addition to being very small, some algorithms (such as SVR) are sensitive to these scales. For this rescaling, the method known as normalization was used, which scales the data to the range [0,1]. Mathematically, Eq. 4 is adopted, where the minimum value is subtracted from the X value and divide this result by the difference between the maximum and the minimum.

$$X' = \frac{X - min(X)}{max(X) - min(X)} \tag{4}$$

C. Training

The approach to predicting the three output variables (tp_{HL}, tp_{LH}, energy) was to create four models for each of the evaluated regression algorithms (Multiple Linear Regression, Support Vector Regression, Decision Trees, and Random Forest). Thus, in the end, 12 models were trained. Three copies of the dataset of 120,954 observations were created and then the *model selection* module of *sci-kit learn* [12] was used to divide the dataset between the training group (50%), validation (25%) and test (25%).

The tools used to manipulate, visualize, clean, train and test the data were Google's Colab and the python language libraries sklearn, pandas, NumPy, matplotlib, and seaborn. All algorithms were trained using the technique known as *cross validation*. It is performed so that the algorithm makes more general predictions and avoids the *overfit* or bias in the data that was fed during its training. For this, *cross-validation* uses the training set and the validation set to first evaluate how the model is behaving. Furthermore, in this process, several copies of the model being trained are created and its hyperparameters are varied to determine which is the best model with the best configuration of these hyperparameters. The following values were configured for the hyperparameters in the *cross-validation*:

- **Decision Trees (DT) - *Max Depth***: $1, 5, 10, 25, 50$
- **Random Forest (RF) - *Max Depth***: $1, 5, 10, 25, 50$
- **Random Forest (RF) - *N Estimators***: $5, 25, 50, 100, 150$

The SVR models were not included in the cross-validation because their training proved to be very extensive (more than three days), so we decided to use the default hyperparameter values: $C = 1$ and $Gamma = 1/(N_{features} \times Var_X)$ For both the Decision Tree and the Random Forest, *max depth* is the maximum depth each tree will have. The Random Forest *N Estimators* values are how many estimators, that is, how many trees will be created for the model. At the end of cross validation, the values from Table II were obtained.

TABLE II: Cross-validation results

	Hyperparameters		
Output Variable	DT *Max Depth*	RF *Max Depth*	RF *N Estim.*
Tp_{HL}	5	5	25
Tp_{LH}	10	10	150
IINT	10	10	150

We can observe that the best models are not shallow, showing that the problem has a significant degree of complexity, but at the same they are not too deep, which reduces the chance of overfitting. We also note that the best hyperparameters are different for the Tp_{HL} and Tp_{LH}. One of the reasons for this is the fact the tree-based models present high variance, so small changes in the input may lead to very different models [13].

IV. Results

After training and validating the models in 75% of the data, the performances were verified in the test set from the RMSE cost function as described in Table III for each variable, where Tp_{HL} is the fall delay, Tp_{LH} is the rise delay, and the third column presents the energy results considering the energy

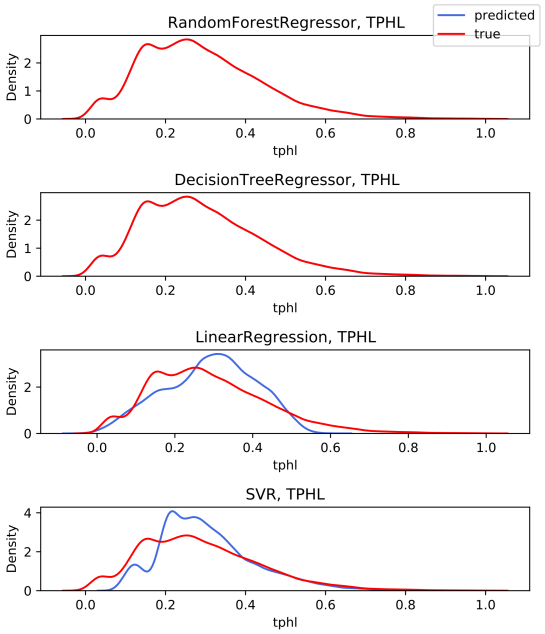

Fig. 5: Delay Prediction: the Tp_{HL} prediction and expected values

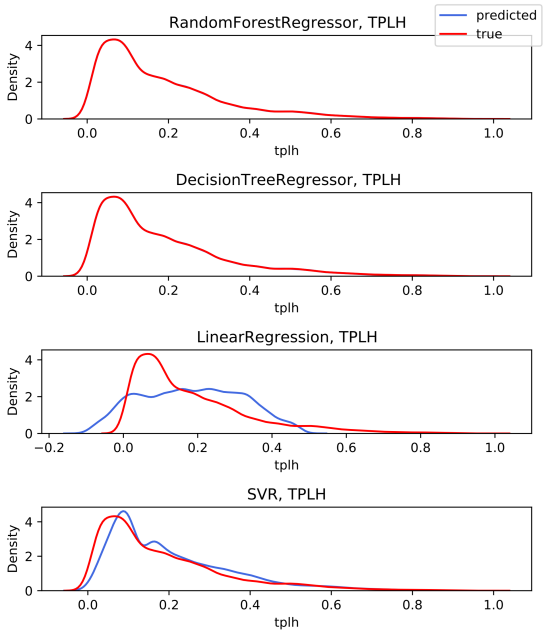

Fig. 6: Delay Prediction: the Tp_{LH} prediction and expected values

consumed to perform two switching at the Inverter input, that is, to perform a set of all possible Inverter transitions over the timing arches. The same unit of the expected time and energy is adopted to demonstrate the significance of the error on the same scale.

TABLE III: RMSE and R results for Tp_{HL} ,Tp_{LH}, and Energy for the CMOS Inverter

Model	Output variables					
	tpHL		tpLH		Energy	
	RMSE	R^2	RMSE	R^2	RMSE	R^2
MLR	0.0956	0.5950	0.0822	0.7311	0.1778	0.4732
SVR	0.0599	0.8465	0.0545	0.8820	0.0569	0.9327
DT	0.0038	0.9996	0.0035	0.9995	0.0135	0.9969
RF	0.0032	0.9997	0.0028	0.9996	0.0103	0.9983

A. *Propagation Delay*

As can be seen in Table III, for Tp_{HL}, the smallest values of RMSE are 0.003245 and 0.003760, which are the model errors for the Random Forest (RF) and Decision Trees (DT), respectively. This is a good result given the fact that the normalized values for the Tp_{HL} range from 0 to 1. This fact is illustrated in Fig. 5, in which one can barely see the difference between the predicted curve (blue lines) and the expected curve (red lines). However, observing Table III and the Fig. 5, the distributions of the predictions of each model compared to the correct distribution shows the biggest errors for regression Multiple Linear (MLR) and Support Vector Regression (SVR).

The results for Tp_{LH} are very similar to those for Tp_{HL}. The RF is still the best method to predict the desired variable and MLR the worst. In addition, the behavior similarities of DT and RF error values remain the same in the Tp_{HL} prediction. In Fig. 6 it is possible to verify these values more intuitively with the expected distribution curve and the curve predicted by each model.

B. *Energy Prediction*

For energy, the best algorithm was, again, RF, followed by DT, with SVR being the second-to-last and the MLR the last one, with the biggest error in the whole table (0.177822). In Fig. 7, it is possible to observe again the same pattern of behavior found for the delay variables.

The worst performance of the MLR algorithm can be explained by the way the correlation between the variables is given. The Pearson's linear correlation coefficient shows that the data used in this work has 0.7 as the highest correlation value, which is considered a moderate correlation, while the rest averages below 0.15, that is considered a weak correlation value. In other words, there is not much information in which the algorithm of Multiple Linear Regression can base itself to have a better performance.

C. *Prediction Time*

A final analysis was performed to assess the trade-off between accuracy and runtime per inference. Our conclusion is that the DT models outperformed the RF ones, even though RF present somewhat superior values of R^2. As an example, a single electrical simulation for the Inverter gate runs for 0.069 s in the HSPICE tools, compared to 0.0006 s of the DT model, and 0.0432 s for the RF model. Thus, even considering that three inferences are necessary to define Tp_{HL}, Tp_{LH}, and energy, a speedup of about $115\times$ is obtained with the DT model and of $1.59\times$ with the RF one.

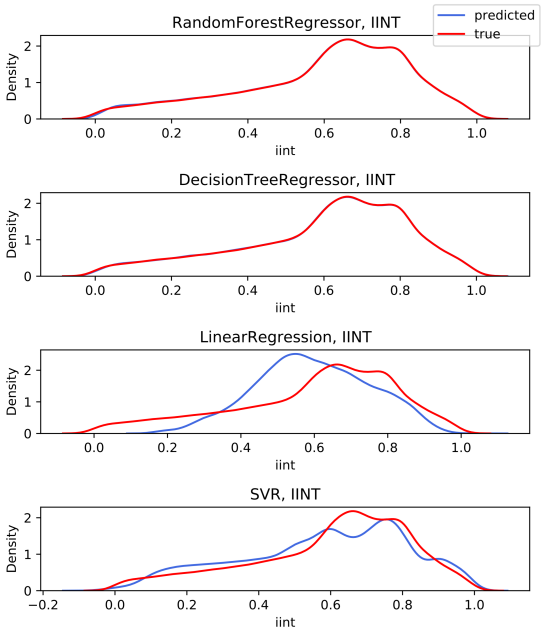

Fig. 7: Energy Prediction

V. Conclusion

In this work, four types of machine learning algorithms were analyzed for the regression task of predicting the electrical behavior of a circuit, considering process variability effects, temperature and voltage oscillations. Considering as case study the CMOS Inverter circuit, the algorithms were trained with data from electrical simulation with Monte Carlo to generate data considering process variability effects.

The results show that for all three output variables (Tp_{HL}, Tp_{LH}, and energy), the best performing algorithm was Random Forest, followed by Decision Trees, that is, the best algorithms were those that base their techniques on trees. The lower performance results of the Multiple Linear Regression algorithm can be justified by emphasizing the fact that many of the variables do not have a linear correlation. SVR shows to be a very powerful algorithm, that, perhaps with the use of other kernels like the polynomial, could be possible to get a smaller error. However, even to reach this better performance in the SVR, this algorithm has a long training time, even longer if considering the cross-validation technique.

As the simulation data set is large and the trend for future works is only increasing, the SVR presents disadvantages to scale as well for fast training. However, it does not subverts the goal of the work which is to reduce the time of circuit electrical simulation through machine algorithms learning because once the model is trained, the inference time is very short, less than a couple of seconds. Finally, it should be noted that, depending on the possible applications of this work, it might be more interesting to use the Decision Tree algorithm, since its error is not so different from Random Forest and its explainability is better and the training time is smaller. The best algorithm was made available online[1]., in a website built with the Streamlit Python library. In the web app, it is possible to simulate the values for the algorithm to make its predictions in real time.

As future work, we intend to explore other more complex and more current machine learning algorithms, such as neural network architectures. Furthermore, it is intended to simulate other circuits more complex, such as XOR and Full-adder gates, to provide a general workflow for combinational logic cells. Also, we plan to detail the evaluation of the regression algorithm including a comparative temporal analysis more accurately between simulation time, training time, and prediction time. This case study with the Inverter circuit shows the potential of adopting machine learning algorithms for cell characterization. Although the Multiple Linear Regression model did not perform well, the Random Forest and Decision Tree were able to predict test set values satisfactorily, which means that machine learning is an interesting tool and powerful to predict the behavior of electrical circuits.

Acknowledgment

This work was financed in part by National Council for Scientific and Technological Development – CNPq and the Propesq/UFSC.

References

[1] Andrew B. Kahng. Reducing time and effort in ic implementation: A roadmap of challenges and solutions. In *Proceedings of the 55th Annual Design Automation Conference*, DAC 18, New York, NY, USA, 2018. Association for Computing Machinery.

[2] A. Chatterjee, D. Croley, V. Ramamurti, and Kui-Yu Chang. Application of machine learning to manufacturing: results from metal etch. In *Nineteenth IEEE/CPMT International Electronics Manufacturing Technology Symposium*, pages 372–377, 1996.

[3] J. K. Lee, K. Ko, and H. Shin. Analysis on process variation effect of 3d nand flash memory cell through machine learning model. In *2020 4th IEEE Electron Devices Technology Manufacturing Conference (EDTM)*, pages 1–4, 2020.

[4] A. B. Kahng, U. Mallappa, and L. Saul. Using machine learning to predict path-based slack from graph-based timing analysis. In *2018 IEEE 36th International Conference on Computer Design (ICCD)*, pages 603–612, 2018.

[5] E. C. Barboza, N. Shukla, Y. Chen, and J. Hu. Machine learning-based pre-routing timing prediction with reduced pessimism. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[6] A. Kaintura, K. Foss, I. Couckuyt, T. Dhaene, O. Zografos, A. Vaysset, and B. Sorée. Machine learning for fast characterization of magnetic logic devices. In *2018 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pages 1–3, 2018.

[7] Aurelien Geron. *Hands-on machine learning with scikit-learn, keras, and TensorFlow*. O'Reilly Media, Sebastopol, CA, 2 edition, 2019.

[8] Charu C Aggarwal. *Data mining: the textbook*. Springer, 2015.

[9] Frank Kane. *Hands-on data science and python machine learning*. Packt Publishing, Birmingham, UK, 1 edition, July 2017.

[10] Wei Zhao and Yu Cao. New generation of predictive technology model for sub-45nm design exploration. In *7th International Symposium on Quality Electronic Design (ISQED'06)*, pages 6 pp.–590, 2006.

[11] Massimo Alioto, Elio Consoli, and Gaetano Palumbo. Variations in nanometer cmos flip-flops: Part i—impact of process variations on timing. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(8):2035–2043, 2015.

[12] David Cournapeau. Scikit-learn Machine Learning in Python. https://scikit-learn.org/stable/, 2007. [Online; accessed 03-May-2022].

[13] Charu C Aggarwal. *Neural networks and deep learning*, volume 10. Springer, 2018.

[1] https://electrical-characterization.herokuapp.com