# Electrical Behavior Prediction of Circuits using ML

**[1]Abstract**

This paper presents the results of four of the most common machine learning regression algorithms (multiple linear regression, support vector regression, decision trees, and random forest) in order to predict the electrical behavior of a CMOS inverter circuit. Specifically, this work will separately predict the values of the propagation delay low-to-high (tpLH), the propagation delay hight-to-low (tpHL), and the energy of the circuit. All the data was generated using Monte Carlo simulations using the hspice simulator. A comparative analysis is made for each variable between all four models in order to understand which one is the best regression model suited for the task.

**[1]Introduction**

Nowadays, integrated circuits have greater processing capabilities obtained through billions of transistors in a single chip \cite{zeppelin, tam2018skylake}. These designs require a sophisticated set of design tools.

As technologies advance, more challenges are emerging. The variability and design rules become more extensive and more complex. Consequently, design tools have to solve increasingly complex problems. This problem has been growing exponentially, and these difficulties directly reflect on the cost of developing electronic device designs \cite{custoKahng}. Machine learning techniques are presented as good alternatives as an attempt to reduce costs for the development of electronic devices.

The ability that machine learning algorithms have to solve complex problems suggests it is a good candidate to solve the high complexity problems imposed by the design of integrated circuits. Scientific research is sought in order to discover how to unite these complex problems for the synthesis of very large integrated circuits with the problem-solving competence that machine learning algorithms bring.

The bibliographic review shows that there is still a lot of space to be explored regarding the use of machine learning for microelectronic tools. Its present usage is strongly linked to quality prediction of candidate solutions given by existing tools, or attempting to guide them to a better quality solution by realizing metric predictions \cite{custoKahng, MLKahng}.

Machine learning algorithms have been explored in recent years in different levels of abstraction to characterize or predict timing, power, and even process variability effects. Regression Simulation was explored in a path-based learning methodology to produce a fast and approximate timing simulator to avoid the high cost associated with statistical timing simulation providing four components: 1) a delay test pattern set, 2) a logic simulator, 3) a set of selected paths as the basis for learning, and 4) a machine learner \cite{ML\_DelayTest}.

With the increasing availability of huge quantities of manufacturing data, and the pressures of continuous process improvement and scrap reduction, machine learning techniques, machine learning techniques can be adopted to analyze, classify, and predict the quality of metal etch \cite{ML\_etch} or investigate process variability effects on 3D NAND Flash Memory Cell\cite{ML\_ProcessVariation\_Flash}.

Machine learning is also employed to early-stage timing prediction in SoC physical design \cite{ML\_SOC\_Prediction}, to deliver timing closure of FPGA designs \cite{ML\_FPGA} or to design for reliability coverage by using critical scenario predictor for running dynamic voltage drop analysis and critical timing path predictor for accurate timing/aging analysis with the impact of voltage drop \cite{ML\_SOC}

In the Placement and Routing steps, we found works that use machine learning to predict path-based slack from graph-based timing analysis \cite{ML\_Slack} and pre-routing timing prediction \cite{ML\_PreRouting}.

In our initial literature review, we just have found one work that explores machine learning on the electrical characterization of devices, however, specifically for Magnetic Devices\cite{ML\_MagneticDevices}. Thus, this indicates that there is room to also explore machine learning algorithms for the electrical characterization of logic cells in bulk CMOS, SOI, or FinFET technologies.

This work explores machine learning techniques to predict the electric behavior of circuits.

**[1]Machine Learning Background**

There is a multitude of machine learning algorithms for different tasks, ranging from predicting the values of the stock market to decision-making in a Mars rover. Since in this work our prediction target is the energy and delay behavior of a CMOS inverter circuit, which are all continuous values, our approach will be to use a supervised learning technique known as Regression, which will give us the correlation between our different independent variables. There are many regression algorithms, hence we will delimit this analysis to the four most common models to see how they fit and compare to each other.

\subsection{Multiple Linear Regression}

Linear regression is a statistical technique widely used when one wants to predict the behavior of a target value \cite{VR}. Since in this work our dependent variables are predicted based on the values of many independent variables, we will be using multiple linear regression instead of simple linear regression. Our predicted value ($\hat{y}$) is given as follows, where $\theta\_0$ is the intercept term and $\theta\_1...\theta\_n$ are the weight of each value of the input features (independent variables).

\[\hat{y}=\theta\_0+\theta\_1x\_1+\theta\_2x\_2+...+\theta\_n x\_n\]

What is desired is to find the values of each weights $\theta\_n$ that minimize the cost function of the model, which in this work will be the root mean square error (RSME).

\subsection{Support Vector Regression}

Even though Support Vector Machines are usually applied to classification problems, it is a versatile algorithm that can perform regressions. The idea is, given a $\epsilon$ value that work as a margin that follow the hyper-plane (works as a "prediction curve") where we try to fit as many observations as possible between the area set by the margins and reduce the values that are off that same area. The decision function is computed similarly to the linear regression, however here we are putting it on a vectorized form:

\[\hat{y}=w^Tx+b\]

\begin{figure}[!htbp]

\centerline{\includegraphics[width=6cm, height=5cm]{Pictures/SVR.png}}

\caption{SVR visualization.}

\label{f: SVR visualization}

\end{figure}

The SVR has many kernels (linear, Gaussian RBF, polynomial), which are functions that calculate the dot product $\phi(a)^T\phi(b)$ without calculating the transformation $\phi$, therefore only using the vectors a, b for that computation. In this work we will use the linear kernel.

\[K(a,b)=a^Tb\]

\subsection{Decision Trees Regression}

This is another algorithm more famously used in classification problems, however it is also possible to use it in regression tasks. The algorithm works by creating a tree structure with internal nodes that are tests made in order to predict to which leaf node the feature will be assigned to, returning, in the regression case, a continuous value.

\subsection{Random Forest Regression}

The random forest, which is a ensemble learning algorithm, is a similar algorithm to the decision trees, the difference being that instead of computing only one decision tree to make a decision for the value, it actually calculates a large number of decision trees with different depths and then it computes the average of all predictions to estimate the value. That makes this method one of the most powerful supervised algorithms. We can represent the joined $n$ decision trees as below.

\[g(x)=f\_0(x)+f\_1(x)+f\_2(x)+f\_3(x)+...+f\_n(x)\]

\subsection{Performance Measurements}

In order to assess the fitness of each model, we need to evaluate the error between the predicted value and the desired value. In this work, we will be using the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) as performance measurements since they are the most commonly used in regression problems. The RMSE gives higher weights for large errors, which means that it is more sensitive to outliers. They are calculated using the Euclidean norm and the Manhattan norm respectively, as follows (in vectorized form):

\[RMSE(\textbf{X},h) = \sqrt{1/m\sum\_{i = 1}^{m}(h(x^{(i)})-y^{(i)})^2}\]

\[MAE(\textbf{X},h)=1/m\sum\_{i=1}^m|h(x)^{(i)}-y^{(i)}|\]

Where $m$ is the number of instances on the data set, $x^{(i)}$ and $y^{(i)}$ are vectors of all features values and the desired value of the $i^{th}$ feature of the data set, $\textbf{X}$ is the matrix of the feature vectors and $h$ is the model function for the prediction.

**[2]Methodology**

In order to predict the tpHL, the tpLH and the energy of the CMOS inverter circuit, we will be simulating the values of the following nine independent variables:

\begin{center}

\begin{tabular}{|p{2cm}||c|c|c|}

\hline

Dependent Variable & tpHL & tpLH & Energy\\

\hline\hline

\multirow{9}{2cm}{Independent Variables} & NMOS\_Vth0 & NMOS\_Vth0 & NMOS\_Vth0\\

& PMOS\_Vth0& PMOS\_Vth0& PMOS\_Vth0\\

& temperature& temperature& temperature\\

& voltage& voltage& voltage\\

& width pmos& width pmos& width pmos\\

& width nmos& width nmos& width nmos\\

& length np& length np& length np\\

& Energy& tpHL& tpLH\\

& tpLH& Energy& tpHL\\

\hline

\end{tabular}

\end{center}

The values for the nmos@var and the pmos@var were randomly assigned by the Monte Carlo Simulation that was run in the hspice simulation software. We preset the values for the width, length, temperature and tension. The variations were as follows:

\begin{itemize}

\item \textbf{Temperature ($^oC$)}: $-25$, $0$, $50$, $75$, and $100$

\item \textbf{Width of PMOS (nm)}: $140$, $70$, $280$, $350$, $420$

\item \textbf{Width of NMOS (nm)}: $70$ and $140$

\item \textbf{Length of PMOS and NMOS (nm)}: $32$, $20$, and $40$

\item \textbf{Voltage (V)}: $0.6$, $0.7$, $0.8$, and $0.9$

\end{itemize}

As for the tpHL, tpLH and energy, they were all calculated by hspice based on all these variables. We ran 1000 transient simulation of 20ns with a step of 0.1ns, therefore we had a total of 168,000 observations.

However, due to outliers removal that otherwise would give us wrong results, our final number of observations was 120,954. In order to understand how each variable relates to each other, we calculated their correlation as shown on Figure 2.

As we can see, the strongest correlation in our dependent variables are found between them and the width of the transistor and the temperature of the circuit, so they will certainly have a higher weight on the models' decision function.

\begin{figure}[!htbp]

\centerline{\includegraphics[width=10cm, height=9cm]{Pictures/download.png}}

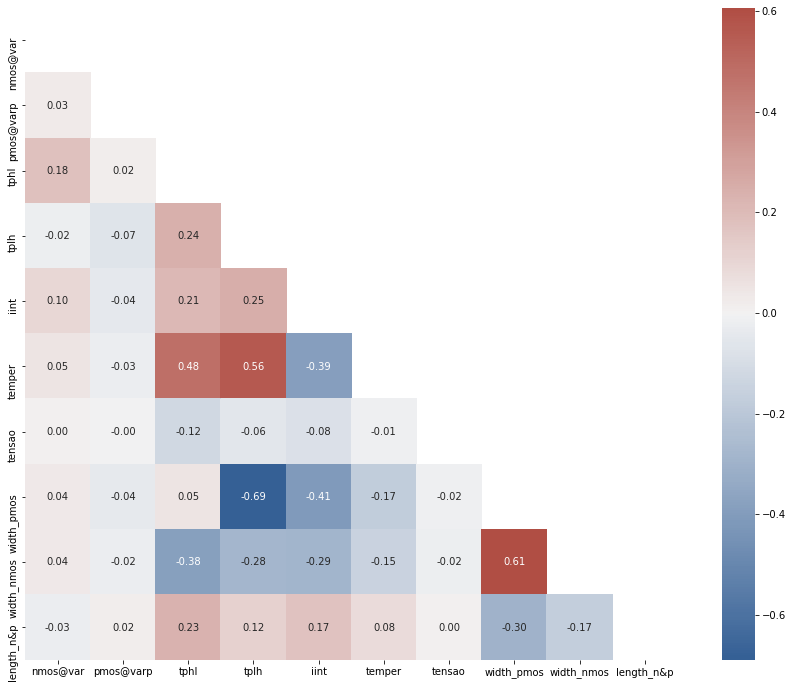
\caption{Correlations (Pearson's R)}

\label{f: Pearson's R}

\end{figure}

Our approach to predict the three dependent variables (tpHL, tpLH, energy) will be to create all four models to each of them, so in the end we end up with a total of 12 trained models. Therefore we created three copies of the 120,954 observations data set and then used sci-kit learn model selection module to split our copies of the data sets between the training set and the test set. Our chosen ratio was the standard 80/20. Scaling was done on every training set because the SVR is sensitive to value scale due to its boundaries calculation.

The tools we used to manipulate, visualize, clean, train and test the data were Google's Colab and the libraries sklearn, pandas, numpy, matplotlib and seaborn.



**[2]Polynomial Regressor**

Variáveis dependentes:

1. Energia
   1. Visualização e Erro
2. TPHL
   1. Visualização e Erro
3. TPLH
   1. Visualização e Erro

Tabela geral dos Erros.

**[3]SVR**

Variáveis dependentes:

1. Energia
   1. Visualização e Erro
2. TPHL
   1. Visualização e Erro
3. TPLH
   1. Visualização e Erro

Discussão (Usado mais para classification tasks)

**[3]Decision Trees**

Variáveis dependentes:

1. Energia
   1. Visualização e Erro
2. TPHL
   1. Visualização e Erro
3. TPLH
   1. Visualização e Erro

**[3]Random Forest**

Variáveis dependentes:

1. Energia
   1. Visualização e Erro
2. TPHL
   1. Visualização e Erro
3. TPLH
   1. Visualização e Erro

**[4]Conclusion**

Best Algorithm and why for:

1. Energy
2. TPHL
3. TPLH

Trabalhos futuros e limitações (o que não foi considerado- variabilidade de processos, outras medidas de erros - cross validation, cost function, more fine tuning, other better perfoming algorithms).

**[4]Acknowledgments**

LeruLeru

**[4]Fontes**

LeruLeru