Development of a Hybrid Machine Learning Model (Decision Tree + Linear Model) for predicting delays in digital integrated circuits

1st Marvin Castro Castro
Escuela de Ingeniería Eléctrica
Universidad de Costa Rica
San José, Costa Rica
marvin.castrocastro@ucr.ac.cr

2nd Erick Carvajal Barboza
Escuela de Ingeniería Eléctrica
Universidad de Costa Rica
San José, Costa Rica
erick.carvajalbarboza@ucr.ac.cr

***Abstract*—Static Timing Analysis (STA) is a critical step in the design flow of integrated digital circuits, but obtaining accurate delay estimations can be challenging when limited physical information is available. This work presents a machine learning-based approach that improves pre-routing delay estimations generated by the open-source RTL-to-GDSII tool OpenLane. The proposed model achieves a 79% reduction in error compared to OpenLane's estimates and demonstrates a 71% improvement even without utilizing OpenLane-specific parameters. Overall, this method offers a lightweight and faster alternative to traditional delay propagation techniques and more complex machine learning models.**

***Index Terms*—Static Timing Analysis, Delay Prediction, Machine Learning, Decision Tree, Linear Regression, Ridge**

I. Introduction

One of the main reasons that the VLSI design flow is described as a feedback loop is because of the trial and error that is encountered during the STA stage. Iterative refinements make this process inefficient and time-consuming result, as it is necessary to avoid timing violations through processes such as gate resizing, logic restructuration, and buffer insertions.

One approach often used by STA tools to guarantee functionality is pessimism, which consists in the asumption that delays will be higher than the actual results. This approach may ensure correctness but it can also lead to other kinds of problems, such as suboptimal resource utilization, reduced space for logic, and slower clock speeds. These tasks are particularly challenging during pre-routing stages, where insufficient physical information about the circuit makes it difficult to accurately calculate the delay of the net.

Most STA algorithms rely on a timing graph where delays are calculated and propagated across all paths. While these algorithms are generally scalable and fast, they often lack sufficient physical information about the circuit, which can impair the accuracy of the results. Machine Learning models have been applied to address this issue; by utilizing information from other designs, they can aid in delay calculations or even make independent predictions."

This work presents a hybrid model to predict net delays, which it's described as hybrid due to the integration of two well-known learning algorithms: decision trees and linear regression. The decision tree is a fast and simple model, excelling in its ability to adapt to different data distributions and handling of non linear data. It is also robust to outliers which can be present on this specific application. However, decision trees can be prone to overfitting, sensitive to small data changes, biased towards dominant features, and limited in their ability to perform well in data extrapolation. On the other hand, linear regressors are among the most commonly used models in learning-based applications. They are easy to understand, fast, lightweight, and, with the proper choice of regularization, less likely to overfit.

With the merging of both models, we aim to create a robust model that addresses the weaknesses of each. The decision tree's ability to split the data into distinct regions (nodes) allows for a linear regression model to be applied within each region, ensuring that the regression is performed within a relevant area of the feature space. Similarly, the piecewise nature of the decision tree can handle non-linear relationships by partitioning the data in a way that is more manageable for the linear regression. Additionally, solutions to the decision tree's overfitting, such as reducing its depth, are less likely to degrade prediction accuracy due to the presence of the second model responsible for the final prediction.

II. Related Previous Work

The prediction of Static Timing Analysis (STA) with learning based models has been researched extensively. In [2] a machine learning-based pre-routing model is implemented through a Random Forest structure, capable of reducing pessimism and yielding accurate timing predictions when compared to a commercial tool. Similarly, [7] presents a machine learning-based approach in conjunction with offset-based timing correlation, to estimate wire delay and slew for individual timing arcs and reduce endpoint slack estimation errors. More complex learning methods such as Neural Networks are explored in [6] and [5] through the use of Multi-Layer Perceptrons (MLP), both studies demonstrated significant results in delay estimation and crosstalk analysis,

respectively. In [8] a Graphical Neural Network (GNN) is used to predict the maximum arrival time with a significant speedup and accuracy. To the best of our knowledge, no prior work has employed a hybrid model similar to the one presented here in an EDA-related application. Nevertheless, methods that combine a decision tree with multiple linear regressions applied to its leaf nodes have been explored in different contexts. For example, [4] uses this approach to forecast temperatures up to seven days in advance, while [1] applies it to various fields, including adversiting, sales, and the study of biological characteristics in animal species.

III. Methodology

A. Machine Learning Model Structure

Our model predicts post-routing delay for individual nets using pre-routing information by analyzing each net's driver and sink gates individually. For instance, in Figure 1, if the delay to sink e on Net C is to be estimated, capacitance, slew, and location information are obtained from gate C0, which serves as the driver, and gate C1, the sink for this path. The capacitance and location of both gates are considered. When estimating the delay to sink e, additional sinks, f and g, are incorporated as context sinks. These additional paths contribute to the load capacitance, while the locations of gates C2 and C3 also influence routing.

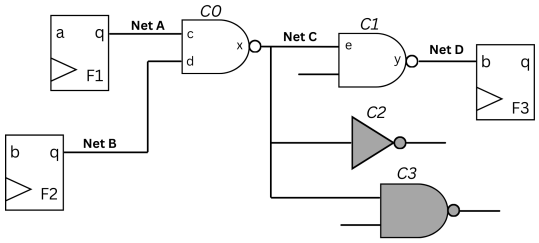

Fig. 1. An example of a circuit

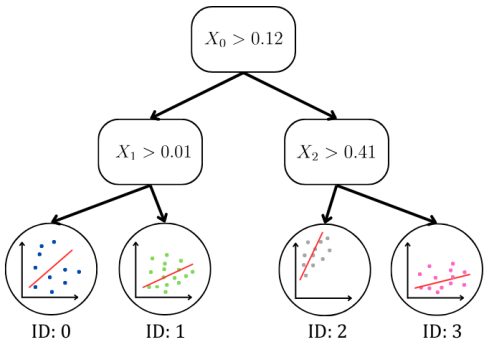

Fig. 2. Hybrid model architecture

The model architecture initiates with a decision tree to partition data based on feature splits. After the tree is constructed, each leaf node is fitted with its own linear regression, specific to the subset of data reaching that node. For new data prediction, each sample is directed by the tree to a particular leaf, where the corresponding linear regression model will generate a prediction based on the sample.

Figure 2 describes the architecture, where the top resembles a traditional decision tree, and each leaf node at the bottom is assigned a unique ID linked to its own linear regression model.

B. Model Features

- **Fanout:** Higher fanout typically leads to increased load capacitance, which directly affects signal propagation delay.
- **Slew:** The signal transition time is also directly influenced by load capacitance; a higher slew rate correlates with an increase in propagation delay
- **Delay:** The delay parameter represents an approximation made during the pre-routing phase by the EDA tool, in our case, the open-source RTL-to-GDSII tool OpenLane.
- **Driver and Sink Distance:** The distance between driver and sink reflects wire lenght which is proportional to our target variable.
- **Driver and Sink Capacitance:** The driving strength of a gate correlates with its output capacitance, while sink capacitance represents the total load applied to the driver. The difference between these capacitances influences the net delay estimation.
- **Driver and sink cell size** Similarly to the driver and sink capacitances, this feature captures the difference between driving strenght and load capacitance.
- **Context sinks location and standard deviation** As previously noted, the presence of additional paths increase routing utilization, load capacitance and slew, resulting in higher delay. To account for these effects, the median location of all context sinks along the path is used as an approximation of the average distance from the driver to the context sinks. Additionally, the standard deviation of the context sinks location serves as a measure of spread, indicating how widely are the sinks distributed.

C. Data generation

Data samples are obtained from signoff timing reports provided by OpenLane, which include each net's name, fanout, slew, and capacitance. Additional information, such as driver and sink locations, is gathered from post-placement reports. The design flow was completed in OpenLane using the Skywater 130 nm Technology PDK to generate the data. Data collection is performed by running the RTL-to-GDSII flow of various designs from benchmark circuits, including ISCAS-89 [3], OpenCores and examples provided by OpenLane. The Skywater 130 nm Technology PDK is used as the process node. Multiple runs from the same design are included, with variations related to placement, routing, clock frequency, pin layout, etc.

We use three main datasets, and their relationships are illustrated in Figure 3. The first, *train.csv*, is the largest of the three. It contains 8 different designs with up to 10 variations

for each design. *variations.csv* contains a limited subset of the same designs as *train.csv* but with different variations, while *unseen.csv* is independent of the other two, containing different designs along with their variations.

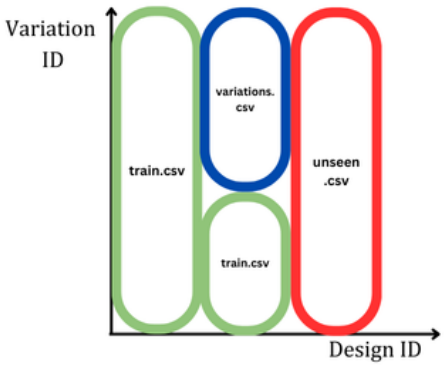

Fig. 3. Correlation between datasets

D. Model Training and Selection

For the selection of the model, numerous instances of the hybrid model were trained using various configurations of parameters on the training data. Different permutations of these parameter combinations were tested, and the configurations yielding the lowest root mean squared error were added to a database and assigned a run ID. Run ID #0 represents the lowest error for the validation of the testing data. As the ID increases so does the associated error. The following variables were considered in the hybrid model:

- **Data Scaling:** Including standardization, Min-max feature scaling or no data scaling. Standardization places different variables on the same scale by making the values on each parameter have a mean equal to zero and unit-variance. The formula is $x' = \frac{x-\bar{x}}{\sigma}$ where x is the original feature vector, $\bar{x}$ is the mean of the same feature vector, and σ is the standard deviation. The min - max feature scaling brings all values of a parameter in the range [0, 1]. The formula is $x' = \frac{x-x_{min}}{x_{max}-x_{min}}$.
- **Decision Tree Hyperparameters:** Maximum depth of the tree and feature both ranging from 5 to 15.
- **Utilization of the median location of the context sinks**
- **Utilization of the standard deviation of the location of the sinks**
- **Utilization of the Distance parameter:** Two main scenarios were tested and compared. In the first, the model's features included both the X and Y coordinates of the driver and sink positions. In the second, these coordinates were replaced with a single feature representing the Euclidean distance between the driver and sink.
- **Type of linear regressor:** Ridge or Ordinary Least Squares (OLS) regression.

E. Machine Learning Algorithms

- **Decision Trees:** Decision trees predict outcomes by recursively partitioning the feature space. Each node in the tree represents a split based on a feature, chosen to maximize the reduction in impurity, in our case, the Gini impurity. At each split, the tree chooses the feature and threshold that best separate the data to reduce variance in predictions. The cost function, often based on impurity (e.g., $\sum_{k=1}^{K} p_k(1-p_k)$ for Gini), is minimized at each split to improve predictive accuracy.
- **LinearRegression:** Corresponds to the ordinary least squares linear regression $y = \beta_0 x_0 + \beta_1 x_1 + ... + \beta_n x_n + \epsilon$. It minimizes the sum of the squared error between observed and predicted values with the cost function $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$. It offers no regularization, fitting all features equally. This makes it sensitive to outliers.
- **Ridge Regression:** Utilizes the L2 regularization to provide an enhanced cost function that penalizes the size of weights β_n. This helps mitigate possible overfitting or multicollinearity issues. The cost function is $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$

IV. Results

A. Model Selection Results

The best three results for the optimal model configurations are shown in table I . Run #47 achieved the smallest RMSE on the *test_design* dataset, utilizing a standard three structure with normalized data scaling. The second entry, Run ID #32 displays the smallest RMSE for the *test_labels* dataset, demonstrating the model's ability to perform well with a reduced tree depth. Run ID #3 presents the configuration for the smallest combined error for both datasets. All configurations used Ridge as the linear regressor. Notably, the hybrid model achieved up to five times less error compared to OpenLane, and the correlation coefficient remained above 0.9.

B. Delay prediction results

Results in this section use the hybrid model configuration ID #3, shown in Table I, as it provided the best outcomes for both test sets. Table **??** offers a detailed result list for each circuit in the testing pool. On average, the hybrid model achieves 40% less error than OpenLane for the first two circuits. For the third circuit, `s38417`, the model performs exceptionally well, showing an 84% improvement over the tool. Only `s15840` shows higher RMSE by 23%, indicating a less favorable result for the model.

Figures 4 and 5 display the prediction accuracy for the hybrid model and OpenLane, respectively. The hybrid model's results are closer to the ideal prediction line $y = x$, while OpenLane tends to yield smaller predictions compared to the actual values. Additionally, OpenLane shows a propensity for outlier predictions, as evidenced in the top-left corner of the graph.

The histogram in Figure 6 compares the error distribution between OpenLane and the proposed model. Overall, OpenLane exhibits higher errors both in terms of frequency and

TABLE I
MODEL SELECTION RESULTS

Run ID	Data Scaling	Context Features	Distance Parameter	Tree Max Depth	.pkl Size (MB)	Test	RMSE Model	MAE Model	Corr Model	RMSE Tool	MAE Tool
47	Normalized	No	Yes	10	9.83	variations.csv	0.5977	0.2527	0.93	1.9505	1.0056
						unseen.csv	0.8193	0.3308	0.91	2.3349	1.2819
32	Normalized	Yes	Yes	5	12.94	variations.csv	0.5465	0.2335	0.94	2.0877	0.2335
						unseen.csv	0.8489	0.3487	0.90	2.0011	1.3159
3	No	No	Yes	13	9.83	variations.csv	0.5942	0.2479	0.93	3.0579	0.8165
						unseen.csv	0.8212	0.3305	0.91	1.2967	0.6394

TABLE II
DETAILED DELAY PREDICTION RESULTS FOR THE HYBRID MODEL AND THE OPEN SOURCE TOOL

	Model			Model w/o Delay			OpenLane		
Circuit	RMSE	MAE	Correlation	RMSE	MAE	Correlation	RMSE	MAE	Correlation
picorv32	0.8794	0.3581	0.9011	1.2763	0.6475	0.7751	1.2994	0.6403	0.7999
zipdiv	0.8586	0.3517	0.8990	1.2383	0.6317	0.7771	1.2474	0.6261	0.8085
s38417	0.6625	0.2700	0.9242	0.9732	0.4797	0.8241	2.9488	0.7863	0.3140
s15850	0.6559	0.2634	0.9222	0.9469	0.4646	0.8263	3.3008	0.8740	0.2692

magnitude, as seen with occurrences such as -20. In contrast, the proposed model's prediction errors are closer to zero, and the number of instances where the error is nearly zero is 1.37 times greater than in OpenLane.

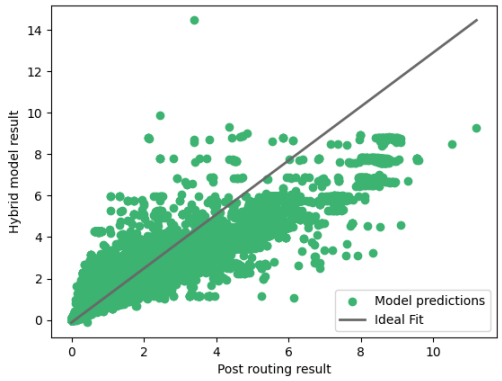

Fig. 4. Hybrid model prediction versus post routing result

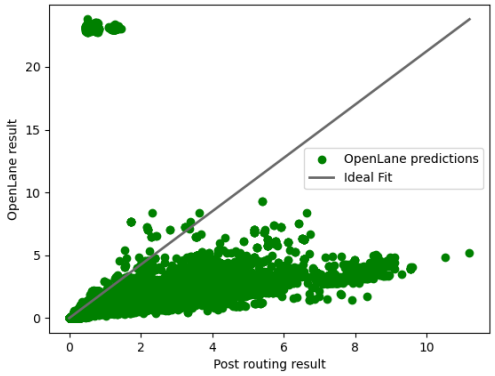

Fig. 5. OpenLane prediction versus post routing result

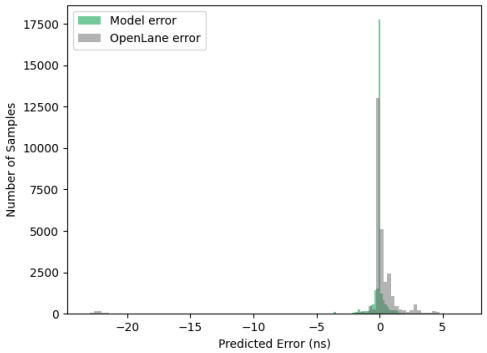

Fig. 6. Error distribution for sink delay predictions

REFERENCES

[1] Maryam Azeez, Mustafa Akpinar, and Kayhan Ayar. A hybrid prediction approach using multiple linear regression and decision trees. In *2023 9th International Conference on Information Technology Trends (ITT)*, pages 61–66, 2023.

[2] Erick Carvajal Barboza, Nishchal Shukla, Yiran Chen, and Jiang Hu. Machine learning-based pre-routing timing prediction with reduced pessimism. In *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, 2019.

[3] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *1989 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1929–1934 vol.3, 1989.

[4] Jing-Rong Chen, Yu-Heng Lin, and Yih-Guang Leu. Predictive model based on decision tree combined multiple regressions. In *2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 1855–1858, 2017.

[5] Vidya A. Chhabria, Ben Keller, Yanqing Zhang, Sandeep Vollala, Sreedhar Pratty, Haoxing Ren, and Brucek Khailany. Xt-praggma: Crosstalk pessimism reduction achieved with gpu gate-level simulations and machine learning. In *2022 ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD)*, pages 63–69, 2022.

[6] Huiyong Huang and Zhong Guan. A deep learning model for pre-routing path delay prediction in vlsi based on subpath. In *2024 4th International Conference on Electronics, Circuits and Information Engineering (ECIE)*, pages 558–561, 2024.

[7] Andrew B. Kahng, Seokhyeong Kang, Hyein Lee, Siddhartha Nath, and Jyoti Wadhwani. Learning-based approximation of interconnect delay and slew in signoff timing tools. In *2013 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, pages 1–8, 2013.

[8] Daniela Sánchez Lopera, Ishwor Subedi, and Wolfgang Ecker. Using graph neural networks for timing estimations of rtl intermediate representations. In *2023 ACM/IEEE 5th Workshop on Machine Learning for CAD (MLCAD)*, pages 1–6, 2023.