|  |
| --- |
| **2025 Girl Hackathon Ideathon Round: Solution Submission** |
| Project Name: AI-Based Prediction of Combinational Logic Depth for Identifying Timing Violations in RTL Circuits |
| Participant: Name: Aluru Doondi |
| Participant Email ID: alurudoondi@gmail.com |
| Participant GOC ID: 539231797551 |
| ReadMe File Links (Eg: Github) |
| **Brief Summary**  The study will create an AI-based algorithm for combinational logic depth estimation of signals in behavioral RTL for early identification of potential timing violations. Timing analysis is traditionally done after synthesis, taking too much computation time and typically leading to project delay due to inevitable architectural changes. Because synthesis has to be done before timing reports can be generated, violation fixes here will be time-consuming and futile.  To remove this constraint, the new AI framework makes use of a machine learning technique trained on pre-synthesized RTL information and associated synthesis reports. Through analysis of core attributes like fan-in, gate count, and path length, the model can predict logic depth without full synthesis. It aids designers in the detection of critical paths and potential violations at the early stages of the design flow, eliminating expensive iterations as well as overall design inefficiency.  By incorporating AI-driven predictions into the RTL design flow, designers can make informed logic reorganization choices prior to synthesis, minimizing design risk and turnaround time. The method enhances IP/SoC development productivity with reduced verification complexity and earlier timing constraint guarantee, resulting in quicker project completion and more predictable designs. |
| **Problem Statement**  I am creating an AI algorithm to forecast combinational complexity (logic depth) of signals in behavioral RTL so timing violations would be detected early. It eliminates inefficiency of traditional timing analysis after synthesis that is time-consuming and, in the majority of instances, leads to project delay because of required changes in architecture.  Machine learning allows the algorithm to forecast the depth of logic prior to performing complete synthesis based on pre-synthesized RTL information instead of complete synthesis to identify critical timing problems early. It enhances design productivity with fewer iterations and optimized IP/SoC design flow.  This is a Google Girl Hackathon 2025 - Silicon Track project with a goal of providing the most innovative AI-based solution for early-stage timing analysis of digital design. |
| **The approach used to generate the algorithm.**  It applies machine learning to estimate combinational logic depth of the signals at behavioral RTL in incomplete synthesis. It begins by constructing a dataset, in which the signal parameters are fan-in, number of gates, and path length. The latter are defined as input variables, and target is logic depth.  Random Forest Regressors are selected since they can perform high-level feature interaction and reasoning depth. Scaling is used in data preprocessing to normalize values and divide data into train set and test set. Model training is done using 200 estimators and max depth 10 and accuracy optimization to avoid overfitting.  Following training, the test set is predicted and Mean Squared Error (MSE) is computed in a bid to monitor performance. The prediction of logic depth for the whole dataset is performed using the trained model and predictions are stored for future analysis. |
| **Proof of Correctness**  The proof of correctness in our approach is demonstrated through multiple evaluation metrics applied to the machine learning model. We use **Mean Squared Error (MSE)** to measure the average squared difference between actual and predicted logic depth values, ensuring the model’s accuracy. Additionally, **precision, recall, and accuracy** are assessed where applicable to determine how well the model distinguishes timing-critical paths from non-critical ones. The **confusion matrix** provides further validation by showcasing the distribution of correct and incorrect predictions. By leveraging these metrics, we ensure the robustness of our AI-based logic depth estimation, enabling early detection of timing violations with high reliability. |
| **Complexity Analysis**  The computational complexity of predicting the logic depth depends on fan\_in, num\_gates, path\_length, and other circuit properties. The complexity of our model's Random Forest Regressor is O(T × N log N), with T being the number of trees (200 in our case) and N being the size of the dataset.  Each feature has a different impact: fan\_in impacts input dependency count, num\_gates impacts circuit area and computation cost, and path\_length impacts worst-case combinational delay. Due to the hierarchical nature of decision trees, the height of each tree impacts prediction speed, approximately O(log N) per query.  Feature scaling and feature partitioning add an O(N) pre-processing, while model learning is O(T × d × log N) where d represents the number of features (3 in our case: fan\_in, num\_gates, path\_length). The method generally attempts to optimize prediction effectiveness at the expense of model size and seeks to balance computation expense for detecting early timing faults. |
| **Alternatives Considered**  A couple of other approaches were tried for combinational logic depth estimation, but they were not as comprehensive as the machine learning-based method. One of them was Static Timing Analysis (STA), which extracts the longest path\_length and computes logic\_depth from post-synthesis reports. STA is slow and not suitable for early RTL analysis because it requires full synthesis before timing violations can be detected.  The other method was heuristic estimation rule-based, where the pre-specified formula in terms of fan\_in, num\_gates, and path\_length estimates logic\_depth. While it estimates quickly, it is not so flexible with varying RTL variations and cannot achieve good generalization over different designs. In addition, another approach using a Verilog framework for structural analysis was examined, where a customized parser traverses the RTL netlist to estimate the logic levels. This is a correct method but less scalable and takes a lot of manual work to find time-critical signals.  For example, a 4-bit array multiplier can be analyzed structurally to estimate the logic depth, but applying this on large designs is not possible. Because of these limitations, a machine learning solution was adopted since it correctly automates the task and also doesn't need synthesis.  By training a model on synthesized RTL data, we achieve an efficient, scalable methodology that provides early-stage timing insight, reducing design iterations and supporting better development times. |
| **References and Appendices**  Synthesis SVG:    Dataset used: |