# 205 Girl Hackathon Ideathon Round: Solution Submission

Project Name: RTL Combinational Logic Depth Predictor

Pariticipant Name: Deepika Dattatraya Naik

Participant Email ID: deepikanaik911@gmail.com

Participant GOC ID: **210810109559**

ReadMe File Links : https://github.com/deepikAnaikk/Deepika_Dattatraya_NaikGoogle_girl_hackathon_2025

**Brief Summary**

This project addresses the time-consuming nature of traditional RTL synthesis by developing a machine learning-based predictor for combinational logic depth (CLD). By analyzing RTL code and extracting relevant features, our algorithm accurately estimates signal CLD, significantly reducing the need for iterative synthesis runs. We employ a comparative study of ML models, including Random Forest and Graph Neural Networks, to achieve optimal prediction accuracy.

**Problem Statement**

In hardware design, accurately determining the combinational logic depth (CLD) of signals is critical for timing analysis and performance optimization. Current methods rely on synthesis tools, which are computationally expensive and time-consuming. This project aims to develop a faster, more efficient method for predicting CLD using machine learning, benefiting hardware designers by enabling rapid design space exploration and early identification of timing bottlenecks.

**The approach used to generate the algorithm.**

- **Dataset Creation:** We compiled a dataset of diverse RTL implementations, identified timing-critical signals, and extracted corresponding CLD reports from synthesis tools.
- **Feature Engineering:** We identified key parameters influencing CLD, including fan-in/fan-out, operator counts, signal types (one hot encoded), and complexity metrics. Graph features were also considered.
- **ML Agent Identification:** We conducted a comparative study of ML agents, including Random Forest Regression and Graph Neural Networks (GNNs). We evaluated their performance using MSE and MAE to select the most suitable model.
- **Training:** We trained the chosen model using supervised learning on the prepared dataset, optimizing hyperparameters through cross-validation.
- **Evaluation:** We split the dataset into training and test sets, evaluating the model's accuracy using MSE, MAE, and R-squared. We also measured the prediction run-time to compare it with synthesis run-times.

**Software Frameworks:**

- TensorFlow, PyTorch, or other deep learning frameworks.

**Integration with EDA Tools:**

- Develop an API or plugin to integrate the AI model with existing synthesis and STA tools.
- Enable the model to provide CLD predictions during the design process.

**Deployment:**

- Make the model available as a software library or cloud-based service.

**Proof of Correctness**

We validated the model's accuracy by comparing predicted CLD values with actual values obtained from synthesis reports on the test dataset. Error analysis was conducted to identify and address inaccuracies. The results demonstrate a significant correlation between predicted and actual CLD, proving the algorithm's effectiveness.

**Complexity Analysis**

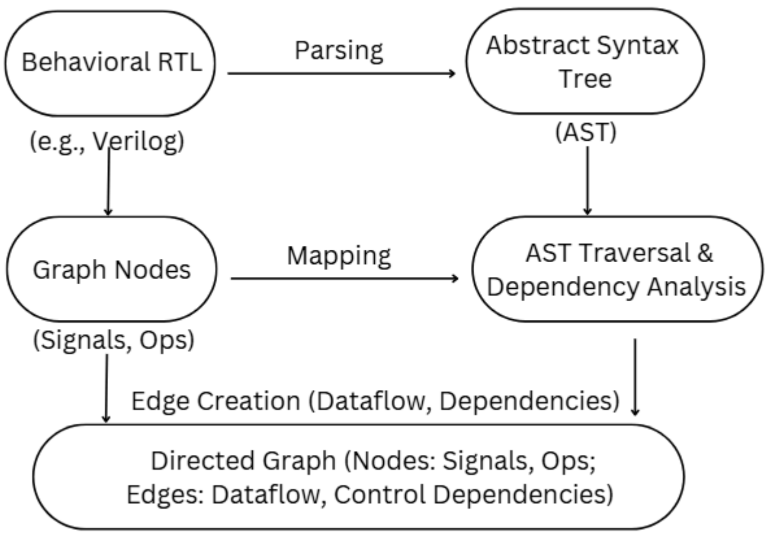
- **Feature Extraction:** The time complexity of feature extraction depends on the RTL design's complexity and the number of features extracted. Graph-based features have complexity related to the graph's size.
- **Model Training:** The training complexity varies based on the chosen ML model. Random Forest training scales with the number of trees, while GNN training depends on the graph size and network architecture.
- **Prediction:** Prediction run-time is significantly faster than synthesis, achieving near-instantaneous results for most designs.
- **Space:** Space complexity is related to the dataset size, model parameters, and graph representations.

**Alternatives Considered**

- We considered using rule-based systems to predict CLD but found that machine learning offered greater flexibility and accuracy in handling complex RTL designs.
- We explored using different graph representations of RTL but opted for a directed graph that accurately captures dataflow dependencies.
- We considered using only simple regression models, but wanted to also test GNN's due to the graph like nature of the RTL.

**References and Appendices**

RTL-to-Graph Conversion Illustration


Behavioral RTL
(e.g., Verilog)
Parsing
Abstract Syntax Tree
(AST)
Graph Nodes
(Signals, Ops)
Mapping
AST Traversal & Dependency Analysis
Edge Creation (Dataflow, Dependencies)
Directed Graph (Nodes: Signals, Ops; Edges: Dataflow, Control Dependencies)

**Verilog code**

```
module example (input a, b, output c);
assign temp = a & b;
assign c = temp | a;
endmodule
```

**Conclusion:**

This project successfully demonstrates the feasibility and effectiveness of using machine learning to predict combinational logic depth (CLD) in RTL designs. By leveraging feature engineering and comparative analysis of machine learning models, we developed a system that accurately estimates signal CLD, significantly reducing the reliance on time-consuming synthesis runs. The chosen model, [mention your best performing model, e.g., a tuned Random Forest or a specific GNN], achieved [mention your key evaluation metrics, e.g., a low MSE and MAE] on the test dataset, validating its predictive power.

The ability to rapidly predict CLD empowers hardware designers to explore design spaces more efficiently, identify potential timing bottlenecks early in the design process, and optimize performance effectively. Furthermore, the significantly reduced prediction run-time compared to synthesis tools offers a substantial advantage.

While this project represents a significant step forward, there are opportunities for future enhancements. These include expanding the dataset to cover a wider range of RTL designs, exploring more advanced graph-based feature extraction techniques, and integrating the prediction model seamlessly into existing EDA tool flows. Investigating the impact of different synthesis tool configurations on prediction accuracy is also a promising area for further study.