56 DESIGN AUTOMATION CONFERENCE
FROM CHIPS TO SYSTEMS – LEARN TODAY, CREATE TOMORROW

Improving design performance using Machine Learning in Synthesis

Gaurav Varshney, Akshi Tomar

Texas Instruments

Introduction

- Synthesis results are important in determining the design performance as it is responsible for
  - Technology mapping to libraries
  - Data-path optimizations and
  - Netlist generation that is consumed by the P&R flow.

- Typically Synthesis optimization engine concentrates on one critical path at a time, improving it until it is no longer critical, and then moving to another.

- Optimization engine ends up spending several iterations in improving timing of top critical path(s) if these paths cannot be optimized due to design constraint or design architecture issues.
  - E.g. macro paths with large access time contributing to slack and paths with small logical depth etc.

- This results in a large number of sub-optimal paths beneath these paths in the design, which could have been optimized if the tool had spent more effort on it. Thus, improving TNS (Total Negative Slack) significantly.

- With the ever-increasing design complexity and shrinking design cycles, there is a growing need for design techniques that optimize timing right from synthesis to closure.

Problem Statement and Motivation

- Getting desired performance through the Synthesis engine involves expert usage of the tool, good understanding of the RTL structures and understanding of technology library

- With the increase in complexity of VLSI designs, complexity of design tools (Synthesis) has also increased. Mastery of Synthesis tool to perform experiments to get desired results is becoming difficult task.

- Quality of synthesis results is direct function of
  - Technology library composition
  - Synthesis tool setup (options/parameters, features)
  - User expertise

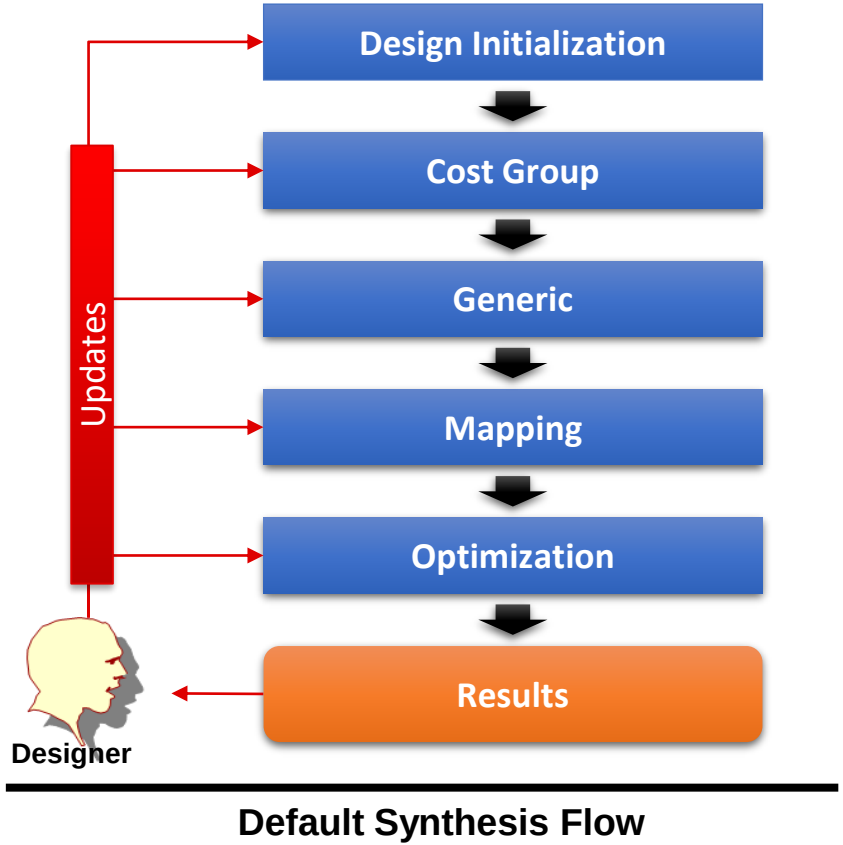
- Often designers end up spending significant amount of time and several iterations in understanding technology libraries and customizing tool setup to get the desired results (PPA)

- These efforts are not reusable from one design to another design

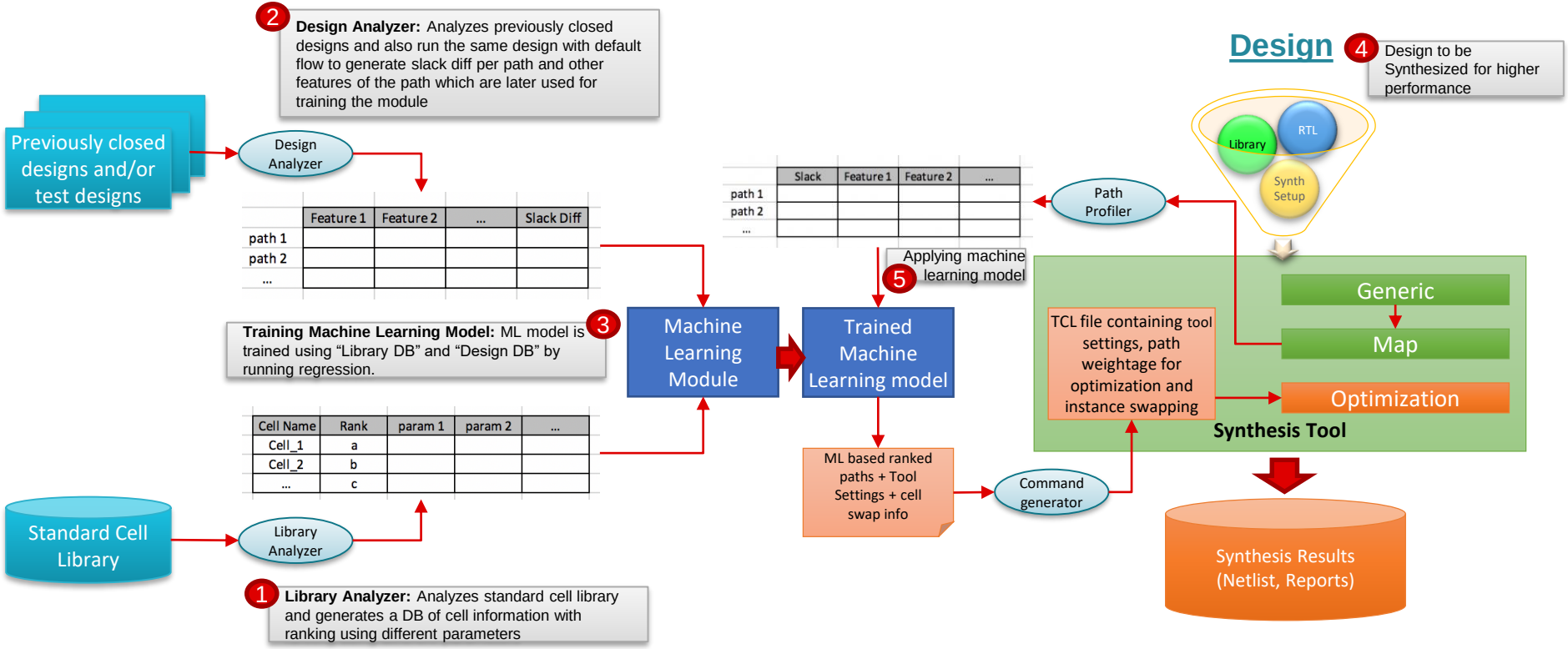
- In this paper, we propose a solution to address these challenges through a Machine Learning model for running synthesis thorough industry tools.

Solution Overview

- Machine Learning to automatically identify required changes to improve design performance
- Usage of previous synthesis results to “predict” the performance and guide the synthesis tool to optimize the predicted critical timing paths



Proposed Solution – Block Diagram

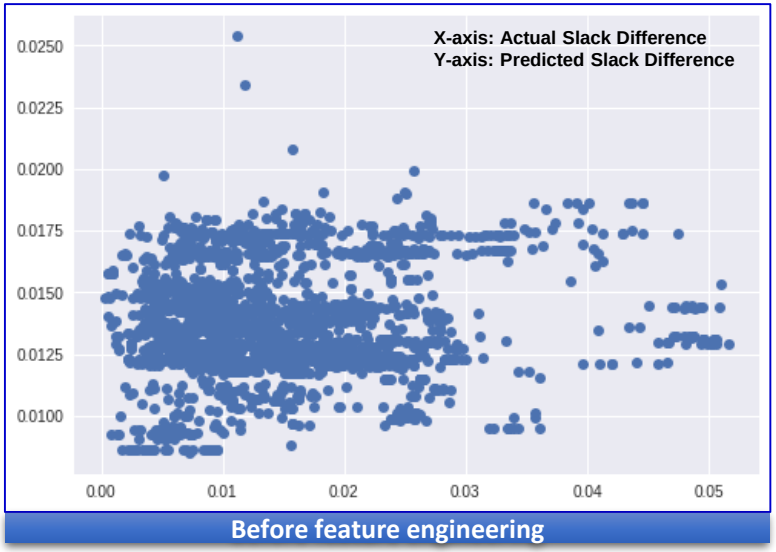


Proposed Solution – Details

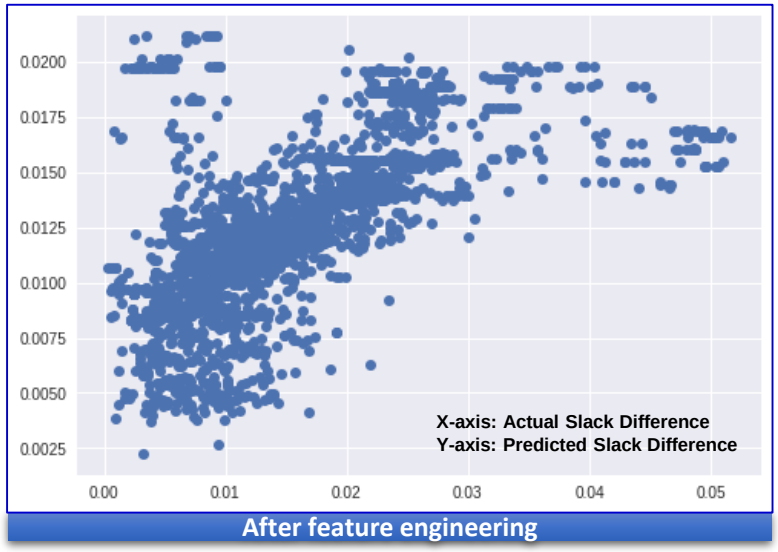
- **Machine Learning (ML) Data Preparation**
  - Library analyser: A DB is prepared for the different parameters which will be utilized by Machine Learning model
  - Design analyzer
    - Different designs which have been closed by experts through custom flows are analyzed as follows:
      - Run design using default synthesis flow till Synthesis-map stage & iterate through all the violating paths in map-DB
      - Compare the slack of the violating path with the final slack in closed DB to get the “slack difference”
      - Find characteristic of paths (path depth, library cells in the path, cell fanout, etc.) used in custom flow
      - Save information in DB (each path details being stored as one record) to be used by Machine Learning model
- **Feature engineering**
  - Once the Library DB and Design DB are prepared, slack differences are analyzed to correlate with the other parameters in the DB to identify features to be used by Machine Learning model
- **Training Machine Learning Model**
  - Model is trained using “Random Forest Regressor” with the identified features and the target being “slack difference”
- **Applying trained model on a new design**
  - For a new design after running Synthesis-map, list of violating path is created along with list of features
  - Trained Machine Learning model is applied to predict the slack improvement. This information used to rank the paths.
  - Output generated by machine learning model: ranked paths, cell swap strategy for select paths
  - This information is converted in Synthesis tool commands to run Synthesis-optimization (and potential incremental optimization). Paths which are ranked higher are assigned higher priority for the slack improvement during optimization.
- **Results**
  - With this method we see significant WNS and TNS improvement for the design

Feature Engineering

- Choosing the right set of features and preprocessing the data is the most crucial step in any machine learning task.
- Feature engineering ensures a better performance of the model: both in terms of training time, model complexity and model evaluation on unseen data.
- Detailed feature engineering to include only relevant features helped eliminate noise and reduce dimensionality as well as multicollinearity.
- The model performance for a random forest regressor is depicted below before and after feature engineering.



Before feature engineering



After feature engineering

Model Evaluation

- The primary goal is to come with up a model that generalizes well across testcases and has small bias and variance.
- Random forest regressor (Hyper-parameters tuned using Random Search CV) and linear regressor showed most promising results.
- Random Forest is better suited as it can model complex dependence on features better.

X axis: Actual slack difference
Y axis: Predicted slack difference

Results

Design-1

*Memory path

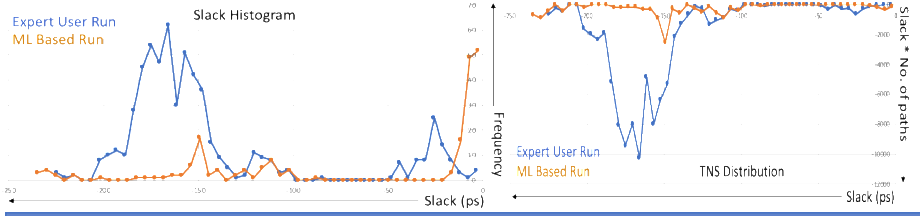
Reference: Expert User run with several rounds of tool option fine-tuning

S.No.	Steps	WNS (ps)	TNS (ns)	FEP	Cell Area
1.	Generic	-552	-138	596	1,926,074
2.	Map	-502	-405	2227	562,363
3.	Optimization	-235	-99	593	548,858
4.	Incremental Opt	-226	-85	589	538,069
5.	Placement + Opt	-744	-1045	2924	584,185
Synthesis CPU time		12,633 (sec)			

Machine Learning (ML) Based Run showing improved WNS and TNS

S.No.	Steps	WNS (ps)	TNS (ns)	FEP	Cell Area
1.	Generic	-552	-138	596	1,926,074
2.	Map	-502	-405	2227	562,363
3.	Optimization with ML	-263	-17	438	558,812
4.	Incremental Opt	-226*	-12	196	547,341
5.	Placement + Opt	-673	-981	2911	590,834
Synthesis CPU time		15,792 (sec)			

85% TNS and 67% FEP improvement in incremental Optimization
10% WNS and 6% TNS improvement after placement Optimization



Design-2

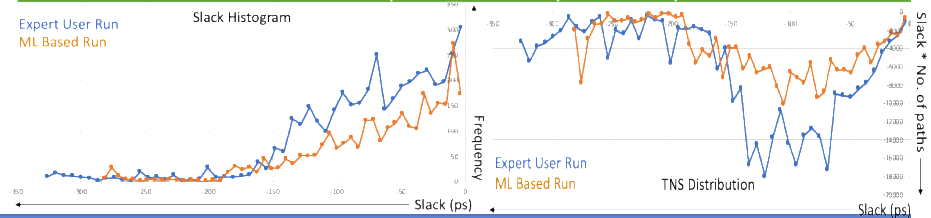
Reference

S.No.	Steps	WNS (ps)	TNS (ns)	FEP	Cell Area
1.	Generic	-1240	-3335	7052	8,094,030
2.	Map	-728	-3096	13442	2,253,562
3.	Optimization	-412	-558	6179	2,146,761
4.	2 Incremental Opt	-329	-339	5186	2,037,588
5.	Postroute	-231	-229	4565	2,519,647
Synthesis CPU time		103,346 (sec)			

ML Based

S.No.	Steps	WNS (ps)	TNS (ns)	FEP	Cell Area
1.	Generic	-1240	-3335	7052	8,094,030
2.	Map	-728	-3096	13442	2,253,562
3.	Optimization with ML	-370	-394	4147	2,151,923
4.	1 Incremental Opt	-283	-202	3027	2,058,926
5.	Postroute	-219	-163	3274	2,557,921
Synthesis CPU time		84,531 (sec)			

14% WNS and 40% TNS improvement with just 1 incremental Optimization
5% WNS and 29% TNS improvement after postroute Optimization



Testing on multiple designs show consistent performance improvement using proposed methodology. Significant cycle time improvement as the solution works out of the box with no additional input requirement. Results can further be improved by reviewing results generated by Machine Learning model with expert user and fine-tuning the generated commands.

Summary

- Paper presents a method for improving design performance using Machine Learning techniques in Synthesis
  - Trains Machine Learning model using information from reference designs (previously timing closed designs and/or input designs covering different scenarios for model training)
  - Analyzes early results in synthesis flow (Mapped DB) to profile the violating paths and applies trained Machine Learning model to guide optimization to focus on right set of paths for optimization
  - Results in improved WNS and TNS

- Methodology can be integrated in any vendor tool

- No improvement with proposed methodology would typically indicate a change required in input constraints and/or RTL update

- Future work:
  - Model performance can be improved by incorporating more features
  - Machine learning can be employed for better design space exploration in terms of the number of cost groups that need to be formed and path distribution in each cost to achieve best design time to performance tradeoff
  - Exploring training using P&R data to improve correlation between Synthesis and P&R