

Machine Learning-Enabled High-Frequency Low-Power Digital Design Implementation At Advanced Process Nodes

Dhushetty venkat sai

I. ABSTRACT

Analyze Graph Based timing Analysis (GBA), to predict Path Based timing Analysis PBA, gain flow-runtime resulting in improved area and power. With high advances in digital implementation at advanced process nodes, most commercial implementation tools use graph based analysis to gain runtime over path based analysis with the cost of high pessimism. The implemented Machine learning algorithm(s) reduces the Graph Based Analysis pessimism by predicting a value closer to vs Path Based Analysis with least runtime overhead, which results in area and power gain without compromising on bottom-line timing closure. The Results achieved by integrating this method in a commercial digital implementation tool has shown a development in area, power leakage and total power centric respectively as illustrated in the original research document. These experimental results produce predicted path based analysis from graph based analysis and are obtained by using Gradient Boosted Random Forest Regressor which is explained below. Multiple other regression models have been used to illustrate the difference in results and run time since implementation of the machine learning algorithm to a commercial tool is out of the scope of this course.

KEYWORDS:-

timing analysis, optimization, gradient-boosted forests

II. INTRODUCTION

In this project, we investigated how to analyze Path Based Analysis with respect to Graph Based Analysis with the use of Machine Learning Algorithms. This paper explains how to predict path based analysis and demonstrates about how to use model adjusted Graph based analysis timing to guide post-route optimization flow in a commercial industrial design at advanced technology nodes ranging from 7nm-16nm, which can also be referred as machine Learning augmented Graph based analysis flow.

The following are the major contributions of our work:

- A) Synthesis, Automated Place and Route (APR), Static Timing Analysis (STA), and data extraction from three designs operating at different clock frequencies


- B) Organization of data, finding correlation, feature selection, and data transformation (standardization) for optimized results and reduced runtime.
- C) Predicted Path Based Analysis using Machine Learning Algorithm(s) such as Linear Regression, Support Vector Machine, Random Forest, *Gradient Boosted Random Forest*, and Neural Network.
- D) Approached strategies that reduce over-design incurred from graph based analysis, and delay fixing and area problems by predicting loss and recovery gained by using predicted PBA in place of GBA.
- E) Illustrated modeling strategies in forms of tables and graphs which highlight the key differences between the models and the trade off between accuracy and run-time. The original research paper enabled the models in commercial electronic design automation tools and exhibited the results on Purchase price allocation in real commercial industrial designs with a range of 5nm-16nm technology nodes.

The existing work on this topic relates to the reducing of the graph based analysis that can be classified as two types:

- A) Heuristic approaches to improve Path Based Analysis Static Time Analysis runtime.
- B) Machine learning-based approaches to predict Path Based Analysis timing.

This paper is organized as follows. In Section III, we explain about various Methodologies used in Machine Learning Algorithms to obtain the Results to predict Path Based Analysis and about the platform and requirements of the system requirements to code test run. In Section IV, we explain about the implementation and simulation. In Section V, we show the results obtained from the code illustrated with the graphs.

The dataset was collected based on the following features below.

The dataset is here is :  GBA_gcn.xlsx

The code is available here :

<https://drive.google.com/drive/folders/1fQHdXikH7rVx8VoYjB4OKPzgiaAeOyO9?usp=sharing>

III. METHODS

A. Linear regression: As this regression assumes that the relationship between your input and output is linear, it does not support different paths during the prediction of the path-based analysis. This may be obvious, but it is also helpful when we have a lot of attributes. Sometimes, we need to transfer data to make the relationship linear.

B. SVM: Support Vector Machines machine learning algorithm is well-known for supervised classification methods. It is based on mapping data points by kernel trick which transforms the input to another high dimensional feature space where a separating hyper-plane can be found. Hence, it is computed by maximizing the distance of closest patterns. We have not considered any hyperparameters for this algorithm as we have implemented this method to compare the results with the random forest regressor.

C. Random forest: We proposed to use Random Forest regressor to predict the Path Based Analysis because of its unique features. One of reasons is that it provides unbiased estimation of the classification error as it is the way that forest is built, it also supports adjusting the error rate when there is an unbalanced no.of samples in the comparison features. Its goal is to find a tree which best predicts the continuous outcome of the interest such as predicting the Path Based Analysis.

D. Gradient Boosted Decision Trees: It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest and this has been illustrated in this paper. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function.

Effect of hyperparameters of the model(s):

1. Random Forest:

- 1.1. Max Trees in the Forest: Large number of trees allows us to create more robust model with less variance. The trade off is that it will take more training time. Having a high number of trees with feasible computational capability should give us desired results
- 1.2. Depth of the Tree: Having more depth results in increased combinations of features and represents the information efficiently.
- 1.3. Number of features to consider at each split: As a result, when utilizing a search strategy to determine the ideal hyperparameters for our forest, carefully tweaking the amount of characteristics to evaluate while splitting at each node is critical.

- 1.4. Size of Bootstrapped Dataset: An appropriate split should be used for creating bootstrapped and out of bag data.

2. Gradient Boosted Decision Trees:

- 1.1. Loss: Loss function to be optimized. The standout feature of the method which is used to reduce the loss of the ensemble by fitting a new tree to the negative gradients of the loss function, thus effectively performing gradient descent.
- 1.2. N_Estimators: The number of boosting stages to perform. Gradient boosting is fairly robust to overfitting so a large number usually results in better performance.
- 1.3. Sub-sample: The fraction of samples to be used for fitting the individual base learners. If smaller than 1.0 this results in Stochastic Gradient Boosting. subsample interacts with the parameter n_estimators. Choosing subsample < 1.0 leads to a reduction of variance and an increase in bias.
- 1.4. Criterion: This function determines the quality of the split and is a greedy method of finding the best split in a single stage.

A) Feature Extraction and Preprocessing:

The data was synthesized in Synopsys Design Compiler and Automatic Place and Route (APR) was done in Cadence Innovus. Scripts were written Tclsh and used to extract data from all the designs.

The features were extracted for each MCMM scenario using GBA analysis so that the optimization flow doesn't incur large runtime overheads. The features have been classified into the following categories:

- 1) Physical Context: The features in this category calculate the layout of critical path to endpoint in a given scenario.
- 2) Physical Constraints: We choose features of a path to assess how much the path can be optimized.
- 3) Timing Constraints: These features differentiate the paths that may lead to a lot of slew fixing leading to large divergence in PBA-GBA slack.
- 4) Logical Context: This category of features differentiate paths based on logic structures
- 5) Timing Context: These features detailed information about a critical path when the other features are similar so as to distinguish from the other paths.

The following features are selected for training the model from a dataset of 9000 samples. Among these 19 features 15 with high variance were chosen to reduce the dimension of the dataset. This resulted in more accurate results and faster run-time. The features were further filtered to 10 with the use of Linear Regression based feature selection method. We standardized the input for Linear Regression, SVM, and Neural Network before passing it to the model.

Feature	Definition
BEGIN	Pin from where data launched
END	Pin from where data captured
xmin	Min x- coordinate
ymin	Min y- coordinate
xmax	Max x- coordinate
ymax	Max x- coordinate
Xdiff	Length of datapath
Ydiff	Length of datapath
EPslewrmax	max slew of end when data is in transition from 0-1
Epslewfmax	max slew of end when data is in transition from 1-0
Epslewrmin	min slew of end when data is in transition from 1-0
Epslewfmin	min slew of end when data is in transition from 1-0
Epcapfmax	end point input pin max capacitance [1-0]
Epcapfmin	end point input pin min capacitance[1-0]
Epcaprmin	end point input pin min capacitance[0-1]
arrival max	the max time elapsed for a signal to arrive
arrival min	The min time elapsed for a signal to arrive
Skew	A difference between the clock arrival time across the chip
PBA	Path based delay
GBA	Graph based delay

Table 1: features.

A) Model Selection:

We used Linear Regression, Support Vector Machine, Random Forest, Gradient Boosted Decision Tree, and Neural Network and compared the results to identify and illustrate the most appropriate model for predicting PBA.

Gradient Boosted Decision Tree is an ensemble learning approach for classification, regression, and other tasks that constructs an additive regression model, utilizing decision trees as the weak learner.

We chose the Gradient Boosted Decision Tree for the following reasons:

1. It eliminates the requirement of data processing.
2. The results provided by the model are far superior to Random Forest since it uses gradient descent to build subsequent trees.
3. Models trained are easier to debug when compared to deep neural networks
4. Choosing hyperparameters is easier when compared to other DNN models.
- 5.

We constructed the Gradient Boosted Decision trees using the following approach:

1. We choose 60% of training data for sampling. This is referred to as the bootstrapped dataset.
2. For each node, we randomly select 60% of the features from the extracted features
3. The values of features are chosen such that it maximizes the information gain at a node
4. Using the steps 2-3, grow a binary tree
5. Prune nodes of the above constructed trees which violate the constraints of minimum number of child nodes and maximum depth

B) Hyperparameter Tuning:

We used k-fold cross validation method with $k = 5$ to decide the best set of hyperparameters for the Random Forest Model.

The hyperparameter values obtained from 5-fold cross validation is are as in the table below:

Hyperparameters	Values
k-fold cross-validation	$k = 5$
Max # trees in forest	200
Min, Max depth of any tree	2,6
Subsample training data (K)	60%
Subsample features (M)	60%
Min child nodes	2
Pruning threshold (γ)	-0.005 - 0.005

Table 2: Hyperparameters

Neural network:

We constructed a Neural Network model with 12,000 hidden neurons to compare the results with the other models that have been discussed in this paper and more specifically, 'Gradient Boosted Decision Trees'. Artificial Neural Networks are powerful tools that reflect the behavior of the human brain, allowing computer programs to recognize patterns, solve regression and other common problems in the fields of AI, machine learning, and deep learning. Each individual node has its own linear regression model, composed of input data, weights, a bias (or threshold), and an output. Neural Networks are powerful prediction tools that improve the prediction accuracy over time by continuously learning from the data. The time overhead of a Neural Network makes it not so suitable for the problem in hand. In the 'on the fly' approach, a model is expected to be trained and predict the data faster than the time taken to evaluate PBA. The huge variation in trend from gen2gen makes it necessary for the model to be trained again and again which gives the Gradient Boosted Decision Tree the edge.

One hidden layer with:	12000 neurons
Learning rate:	3e-4
Epochs	7500
Optimizer:	Adam
Input Layers:	linear layer ReLu activation function.
Dimension of input layer:	6x5400
Output Layer:	linear with reply activation
Dimension of output layer:	1x5400

Adam Optimizer: This Optimizer is used instead of the classic stochastic gradient descent to update weights in a network iteratively based on training data.

Mathematical Aspect of Adam Optimizer:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \left[\frac{\partial L}{\partial w_t} \right] v_t = \beta_2 v_{t-1} + (1 - \beta_2) \left[\frac{\partial L}{\partial w_t} \right]^2$$

Parameters Used:

mt = aggregate of gradients at time t (initially, mt = 0)

mt-1 = aggregate of gradients at time t-1

Wt = weights at time t

Wt+1 = weights at time t+1

αt = learning rate at time t

∂L = derivative of Loss Function

∂Wt = derivative of weights at time t

β = Moving average parameter (const, 0.9)

Vt = sum of square of past gradients.

IV. Implementation and Simulation

What is the computing Platform to perform the computation?

Platform used: AMD Ryzen 5 - 8GB Memory - NVIDIA GeForce RTX 3050 Graphics - 512GB Solid State Drive.

A. How to install needed software environment

The software was run on Anaconda IDE.

Download Anaconda from [Anaconda.com/downloads](https://anaconda.com/downloads).

Anaconda (excerpt from official documentation) is a package manager, an environment manager, a Python distribution, and a collection of over 1,000+ open source packages. To install it just follow the instructions in the original documentation (If you are installing anaconda as a rllab dependency **make sure to download the python 2.7 installer**).

Anaconda creates its own environments inside the system. This means that some packages that are installed outside anaconda, have to be installed inside too. To do this you just have to follow the normal installation instructions of the package but inside the anaconda environment.

B. Code Test Run:

The code was run on anaconda successfully and necessary plots are attached from the Test run.

(see Fig 2 and Fig 4).

The Code Test Run demo video is here :-

C. Libraries Used:

1. Pandas for handling input data
2. Numpy for array and matrix operations
3. Matplotlib for plotting results and analysis
4. Sklearn for preprocessing, implementing Linear regression, SVM and Random Forest models, and Gradient Boosting Regressor.
5. Pytorch was used for NN.

V. Results

We generated correlation matrix, giving us more insights on features and their impact on predicting values. The list of features is listed here. Random forest is better at predicting PBA as stated in the paper than SVM (see Fig 2 and Fig 4).

You can observe from fig 2, that the actual and predicted values match closely using Random forest as compared to running models using SVM linear regression (fig 4)

Because our models correlate with PBA and overcome GBA pessimism, this is to be expected. All the necessary results and plots are shown. 9k samples were used. We got data points from 3 different designs as listed below.

Design name	No. of paths
Distance and Sort engine	1k paths
Graph neural convolutio	4k paths
Dark RISC-V	4k paths

Table 3 : no paths for designs used.

In comparison to the GBA flow, we are able to reduce the runtime overhead of our MLaugmented flow.

Design name	clock period
Distance and Sort engine	500 ps
Graph neural convolution	1500 ps
Dark RISC-V	300 ps

Table 4 : Clock periods of different designs

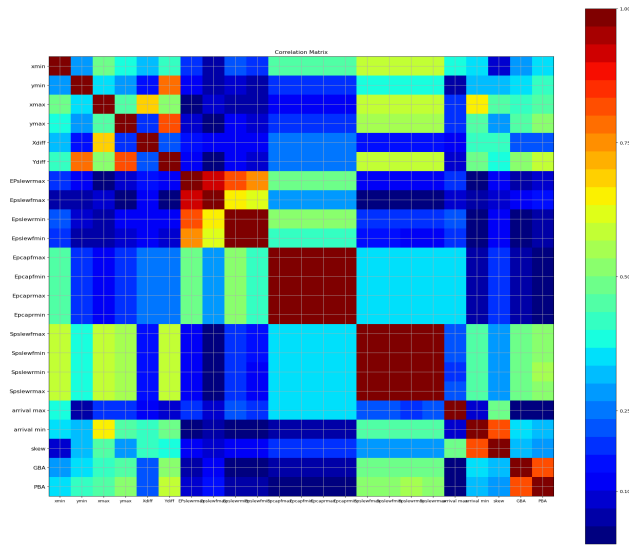


Fig 1 : Correlation Matrix

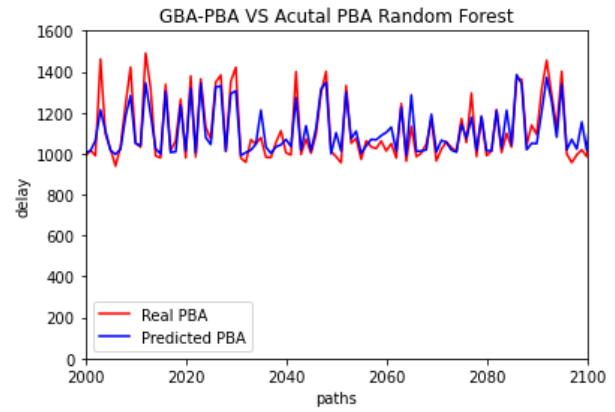


Fig 4 : GBA-PBA vs Actual PBA Random forest

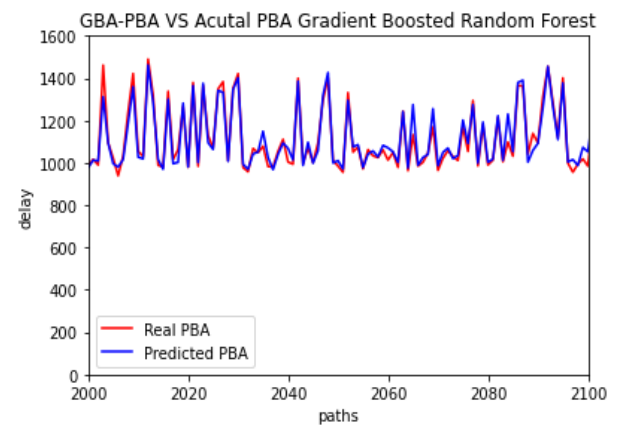


Fig 5 : Gradient Boosted results

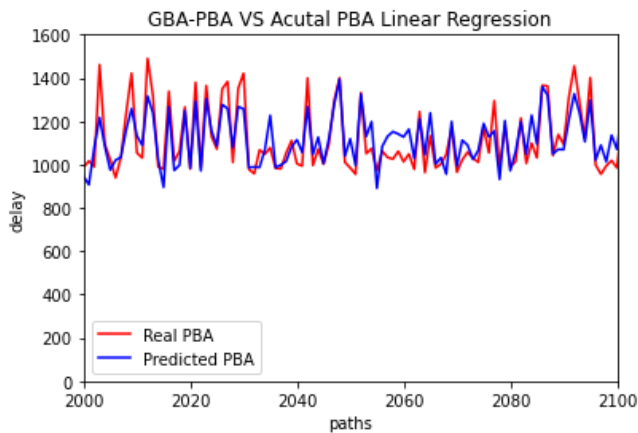


Fig 2 :GBA-PBA vs Actual PBA Linear Regression

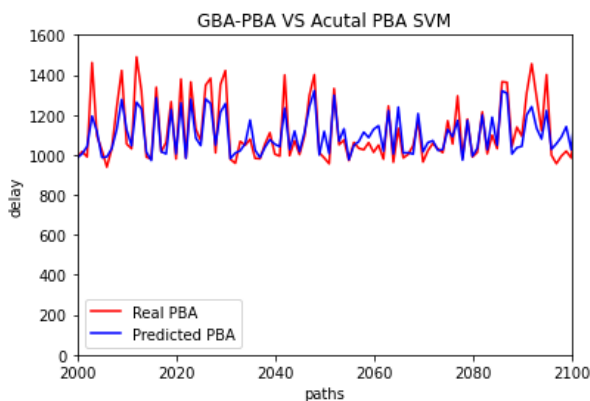


Fig 3 : GBA-PBA vs Actual PBA SVM

```
*****Linear Regression*****
Average error train for Linear Regression 59.1628606603966
Average error test for Linear Regression 61.17142369987416
<Figure size 3600x576 with 0 Axes>
*****SVM*****
Average error train 54.61021496941317
Average error test 55.596954150502974
<Figure size 3600x576 with 0 Axes>
*****Random Forest Regressor*****
Train accuracy
Average PBA vs GBA-PBA 40.22272839156014
Test accuracy
Average PBA vs GBA-PBA 43.68410131098848
<Figure size 3600x576 with 0 Axes>
*****GRADIENT BOOSTED RANDOM FOREST REGRESSION*****
Train accuracy
Average PBA vs GBA-PBA 7.251208810911191
Test accuracy
Average PBA vs GBA-PBA 12.361598244280612
```

Fig 6 : Average PBA vs GBA-PBA

-----Test Results of models-----	
Loss in Linear Regression	61.17142369987416
Loss in SVM	55.596954150502974
Loss in Random Forest	43.68410131098848
Loss in Gradient Boosted Decision Trees	12.361598244280612

Fig 7 : Gradient Boosted result

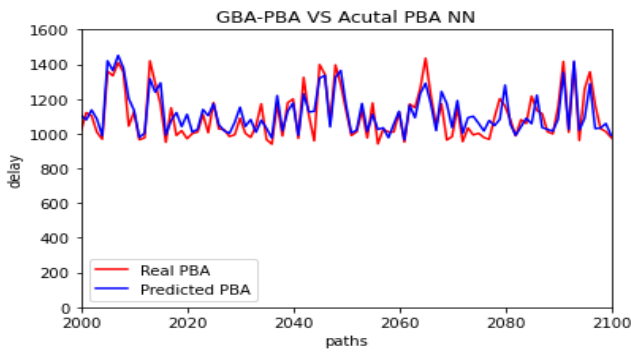


Fig.8 : Neural Network Result

```

iteration 8999
error 0.1533203125
loss 4264.974609375
iteration 9499
error 0.16943359375
loss 4178.125
iteration 9999
error 0.1806640625
error: 0.18115234375
loss 4178.125
End Iteration: 10000
*****NN*****
Train accuracy
Average PBA vs GBA-PBA [47.166454]
Test accuracy
Average PBA vs GBA-PBA [47.67819]

```

Fig.9: GBA-PBA vs PBA NN

VI. Discussions & Conclusions.

In GBA-based digital implementation processes, advanced process nodes make PPA optimization difficult. PBA-based optimization, on the other hand, reduces TAT (turnaround time), making it difficult to use in implementation flows. We developed a ML model that learns PBA timing from GBA timing and then guides an ML GBA flow using our model. We made a comparison between random forest and different ML techniques and detailed its advantages SVM, linear Regression. We gain better PPA with runtime overhead compared to a commercial GBA flow.

From our final code run we can see that Random forest is better at predicting PBA as stated in the paper than SVM. Random forest is better suited for PBA predicting as it gives us more accurate predictions. This will eventually lead to more time saved in the EDA process.

We have added gradient boosted random forest and Neural network; its implementation with results. Gradient Boosted random forest gave us the best results among all methods.

Team experience :- This project gave us insights on improving performance in EDA flows using ML. We learned the difference between GBA and PBA optimization.. We faced challenges during initial data collection. We learnt different methods to implement our model i.e Support Vector Machine, Random Forest, Gradient Boosted Decision Tree, and Neural Network and key differences in them.

REFERENCES

- [1] Chatterjee, P., and Milanfar, P., Patch-based near-optimal image denoising. *IEEE Trans. Image Process.* 21(4):1635–1649, 2012.
- [2] Zhang, J., Xiong, R., Zhao, C., Ma, S., and Zhao, D., Exploiting image local and nonlocal consistency for mixed Gaussian-impulse noise removal. *Proc. IEEE Int. Conf. Multimedia Expo*, Jul.:592–597, 2012.
- [3] Chen, Y., and Liu, K. J. R., Image denoising games. *IEEE Trans. Circuits Syst. Video Technol.* 23(10):1704–1716, 2013.

K. Elissa, “Title of paper if known,” unpublished.

- [4] Rajwade, A., Rangarajan, A., and Banerjee, A., Image denoising using the higher order singular value decomposition. *IEEE Trans. Pattern Anal. Mach. Intell.* 35(24):849–862, 2013.
- [5] Mäkitalo, M., and Foi, A., Noise parameter mismatch in variance stabilization, with an application to Poisson–Gaussian noise estimation. *IEEE Trans. Image Process.* 23(12):5348–5359, 2014.
- [6] Zuo, W., Zhang, L., Song, C., Zhang, D., and Gao, H., Gradient histogram estimation and preservation for texture enhanced image denoising. *IEEE Trans. Image Process.* 23(6):2459–2472, 2014.
- [7] Peng, H., Rao, R., and Dianat, S. A., Multispectral image denoising with optimized vector bilateral filter. *IEEE Trans. Image Process.* 23(1):264–273, 2014.
- [8] [10] T.-W. Huang and M. D. F. Wong, “Timing Closure: Speeding Up Incremental PathBased Timing Analysis with MapReduce” *Proc. SLIP*, 2015, pp. 1-6. [11] T.-W. Huang and M. D. F. Wong, “Accelerated Path-Based Timing Analysis with MapReduce” *Proc. ISPD*, 2015, pp. 103-110. [12] A. B. Kahng, S. Kang, H. Lee, S. Nath and J. Wadhvani, “Learning-Based Approximation of Interconnect Delay and Slew in Signoff Timing Tools”, *Proc. SLIP*, 2013, pp. 1-8.
- [9] A. B. Kahng, M. Luo and S. Nath “SI for Free: Machine Learning of Interconnect Coupling Delay and Transition Effects”, *Proc. SLIP*, 2015, pp. 1-8.
- [10] A. B. Kahng, “Machine Learning Applications in Physical Design: Recent Results and Directions”, *Proc. ISPD*, 2018, pp. 68-73.
- [11] A. B. Kahng, U. Mallappa and L. Saul, “Using Machine Learning to Predict PathBased Slack from Graph-Based Timing Analysis”, *Proc. ICCD*, 2018, pp. 603-612.
- [12] A. Wrixon, A. Belov, M. Keller, R. Moloney and H. Dadheech, “Static Timing Analysis with Improved Accuracy and Efficiency”, *US Patent No. US1002225B2*, 2018.
- [13] “Signoff Summit: Tempus, Path-Based Analysis, and FinFET Timing Closure”, https://community.cadence.com/cadence_blogs/8/b/ii/posts/signoff-summit-tempuspath-based-analysis-and-finfet-timing-closure.
- [14] C. Soviani, R. N. Helaihel and K. Rahmat, “Efficient Exhaustive Path-Based Static Timing Analysis Using a Fast Estimation Technique”, *US Patent No. US8079004B2*, 2011.
- [15] R. Shyamsukha, C. Feng, S. Radhakrishnan and T. L. Craven, “Selectively Reducing Graph-Based Analysis Pessimism”, *US Patent No. 10354042B2*, 2019.