Timing Assistant for Dynamic Voltage Drop Impact on Maximum Timing Pushout

Norman Chang (1), Wentze Chuang (1,3), Ganesh Kumar Tsavatapalli (1), Hao Zhuang (2), Sankar Ramachandran (1), Rahul Rajan (1), Joao Geada (1), Ying-Shiun Li (1), Yaowei Jia (1), Mathew Kaipanatu (1), Suresh Kumar Mantena (1), Ming-Chih Shih (1), Anita Yang (1), Roger Jang (3)

(1) ANSYS Inc., (2) Google Inc., (3) National Taiwan University
Outline

• Introduction
• Timing Assistant Methodology and Flow
• Results and Discussion
• Conclusions
Scenario Selection and Timing Path Selection for DvD-aware Maximum Timing Pushout

- The flow creates maximum timing pushout scenarios/paths from multiple scenarios of both vector and vectorless algorithms based on various parameters from diverse analyses.

- This process can be assisted with ML-based Timing Assistant to learn from historical data for more accurate selection of scenarios and paths that may produce maximum timing pushout.
An integrated ML stack needed for scientific computing, and it natively understands EDA data

Definition of Setup Time Delta Slack due to DvD

- Tc2q and Tcomb will be larger when considering dynamic voltage drop (DvD) of instances along the Data Path.

The corresponding eq. for maximum operation frequency is defined below:

\[ F_{\text{max}} = \frac{1}{T_{\text{clk}}} \]

where

\[ T_{\text{clk}} = T_{\text{c2q}} + T_{\text{comb}} + T_{s2} - T_{\text{skew}} \]

- Data Arrival Time: Time of data arriving input-D of FF2
- Data Required Time: The latest time at which a signal can arrive without making the clock cycle longer than desired.
- Setup Slack: Data Required Time – Data Arrival Time
- Pre DvD Slack: The Slack value without considering DvD
- Post DvD Slack: The Slack value when considering DvD on data path
- Delta Slack: Post DvD Slack – Pre DvD Slack

Ts2 = setup time needed of FF2
Th2 = hold time needed of FF2
Tc2q = Time from clock pin to Q pin of FF
Tskew = Time for the clock path
Tcomb = Time for the combinational instances on the Data Path
System Flow of Scenario Predictor and Path Predictor

Input Timing Path Data

Scenario Predictor

Top-2 Scenarios

(Contains multiple Scenarios with each scenario containing N paths, e.g. 200K paths)

Step 1: Bin-Based classification model
Step 2: Regression model
Step 3: Weighted algorithm

(Top-2 scenarios with maximum likelihood of maximum timing pushout due to DvD)

Perform DvD Analysis on the chosen scenario(s)

Path Predictor

Ranked Paths of Top-2 Scenarios

Perform fast/accurate Timing Analysis on the top ranked paths

Bin-Based classification model to detect paths of maximum timing pushout due to DvD
ML Algorithm Applied

- KNN (K nearest neighbors)
- Decision Tree
- Random Forest
- XGBoost (eXtreme gradient boosting)
- lambdaMART (learning to rank technique)
- Neural Networks
ML Algorithm Applied – KNN (K-nearest Neighbors) Algorithm

- Test data will be classified by a majority vote of its neighbors, with the object being assigned to the class most common among its $k$ nearest neighbors

- Example, $K = 3$:
ML Algorithm Applied – Decision Tree Algorithm

Is a Person Fit?

Age < 30?

Yes?

Eat's a lot of pizzas?

Yes?

Unfit!

No?

Exercises in the morning?

Yes?

Fit

No?

Yes?

Fit

No?

Unfit!

ML Algorithm Applied – Random Forests Algorithm

- Train classifiers based on a subset of features
- Aggregate output of trees (majority vote or averaging) as a typical ensemble learning method

XGBoost (Extreme Gradient Boosting)

- XGBoost is an effective gradient-based boosting method that iteratively improve the classifier starting with weak classifiers
- Source code available (not in Scikit-learn yet but open source); ported to SeaScape
  - Supports multiple languages and platforms
  - Distributed on various clouds
  - Wins many ML competitions
  - Highly scalable and tunable

\[ y_i^p = y_i^p + \alpha \delta \sum (y_i - y_i^p)^2 / \delta y_i^p \]
which becomes, \[ y_i^p = y_i^p - \alpha \sum (y_i - y_i^p) \]

where, \( \alpha \) is learning rate and \( \sum (y_i - y_i^p) \) is sum of residuals


In 2015, 17 out of 29 Kaggle champions used XGBoost in their solutions.
Neural Networks

- \( x \Rightarrow \) Input feature vector
- \( W^l \Rightarrow \) Weights matrix b/w \((l-1)\) to \(l\)th layer
  - \( W^l_{jk} \Rightarrow \) weight from \(k\)th neuron from \((l-1)\)th layer to \(j\)th neuron in \(l\)th layer
- \( b^l \Rightarrow \) Bias matrix of \(l\)th layer
- \( z^l \Rightarrow \) weighted input of \(l\)th layer
- \( a^l \Rightarrow \) Activation matrix of \(l\)th layer
- \( y \Rightarrow \) Output label
- \( C \Rightarrow \) Total Cost (E.g. Square Sum Error)
- \( d^l \Rightarrow \) Error at \(l\)th layer
- \( D \Rightarrow dC/dW \Rightarrow \) Partial derivative of cost w.r.t weights

- Goal to identify optimal weights (W’s) and bias (b’s) for modeling regression or classification
- After having partial derivatives at every layer, we can train model using a learning rate and stochastic gradient decent methods.

LambdaMART

• LambdaMART is the boosted tree version of LambdaRank, which is based on RankNet. RankNet, LambdaRank, and LambdaMART have proven to be very successful algorithms for solving real world ranking problems: for example an ensemble of LambdaMART rankers won Track 1 of the 2010 Yahoo! Learning To Rank Challenge.

• Solving a wide range of supervised learning problem by maximizing information retrieval (IR) functions, like NDCG, which are not smooth functions of the model scores

Possible Path-based Input Features Including Customized Combined Features

• Avg\_timing\_sensitivity\_wrt\_dvd : avg of per instance timing sensitivity w.r.t. DvD on the Data Path
• Max\_timing\_sensitivity\_wrt\_dvd : max of per instance timing sensitivity w.r.t. DvD on the Data Path
• Avg\_switching\_status : avg of per instance switching state (1 or 0 per instance switching)
• Avg\_effective\_R : avg of per instance effective R from bump to instance VDD/VSS pins
• Max\_effective\_R : max of per instance effective R from bump to instance VDD/VSS pins
• Num\_of\_datapath\_instances : number of instances on the Data Path
• Avg\_rail\_connected\_datapath\_Ipeak : avg of rail-connected peak current per instance on the Data Path
• Max\_rail\_connected\_datapath\_Ipeak : max of rail-connected peak current per instance on the Data Path
• Avg\_effective\_decap : avg of effective decap seen by each instance on the Data Path
• Max\_effective\_decap : max of effective decap seen by instances on the Data Path
• Package\_effect : package impact on Data Path instances delay. e.g. Ldi/dt
• Combined\_timing\_impact\_features : e.g., Avg\_rail\_connected\_Ipeak * Avg\_effective\_R * Avg\_timing\_sensitivity\_wrt\_dvd * Num\_of\_datapath\_instances (i.e. from feature engineering)
• Avg\_worst\_voltage\_drop : avg of per instance worst voltage drop on the Data Path (additional for Path Predictor)
• Max\_worst\_voltage\_drop : max of per instance worst voltage drop on the Data Path (additional for Path Predictor)
Scenario Predictor : Step 1 - Bin-based Classification Model

- Input data: 20 Scenarios, 70K timing paths per scenario for training and 30K for testing.
- Split Input data into N bins according to delta slack.
- ML algorithm:
  - KNN
  - Decision tree and Random Forest
  - XGBoost
  - NN
- Input Feature: 17 features
- Output: Bin of paths.

![Diagram of Scenario Predictor]

- Input Data
  - Bin 1
  - Bin 2
  - .
  - .
  - Bin N
- ML algorithm
- Classification Model
- Predict
- Bin 3
- Path i
Scenario Predictor : Step 2 – Ranking Model

- Input Data: Output of step 1 (Top-bin data)
- ML algorithm:
  - Decision Tree regressor
  - XGBoost regressor
  - lambdaMART (learning to rank technique)
- Input Feature: 17 features
- Output: Ranked Timing Paths based on Predicted Delta Slack from Regression model (Rank 1: The most negative delta slack).

```
Bin 1
| Path_77 from Scenario 12
Bin 2
| Path_639 from Scenario 6
| ...
Bin N
| Path_19 from Scenario 18
```

```
Rank 1: Path_32 from Scenario 3
Rank 2: Path_120 from Scenario 9
| ...
Rank n: Path_198 from Scenario 12
```
Scenario Predictor: Step 3 – Weighted Algorithm

- Input Data: Output of Regression model (step 2)
- Output: Top-K scenarios (e.g., K=2)

<table>
<thead>
<tr>
<th>Rank</th>
<th>Path ID</th>
<th>Scenario ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Path_32</td>
<td>Scenario 3</td>
</tr>
<tr>
<td>2</td>
<td>Path_120</td>
<td>Scenario 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Path_198</td>
<td>Scenario 12</td>
</tr>
</tbody>
</table>

Input Data

Weighted Algorithm

Output

- Rank 2: Path_120 from Scenario 9
System Details – Weighted Algorithm

• Once the ranking order inside the top bin is achieved, weighted algorithm will be applied to determine the order of switching scenario among multiple switching scenarios.

• Each scenarios score will be calculated by the following method:
  o Scenarios<i>_score = Score of path<id>_from_scenario<i> * normalization term

  o Score of path<id>_from_scenario<i> defined as following:
    split ranked path into N bins (e.g., N = 10), the score of top-10% ranked path = 10, and bottom-10% ranked path = 1
  o Normalization term = 1 / log(rank_position + 1)
Inference Engine for Critical Path Predictor

- Bin-based classifier by classifying the timing paths to a number of bins (number of timing paths per bin could be 10K – 50K)
- From the Bin classification result, users can choose the Timing Paths in the top bin(s) with potential large delta_slack impact from DvD to run simulation with DvD waveform on Vdd/Vss pins (e.g. Path-FX) for final setup time check.
DvD aware Path Simulation Flow using Path FX

Cell Modeling Flow

- Device Models
- SPICE Netlists

→ Build FXM

→ FX Models

Path Simulation Flow

- GBA Paths & SDC
- SPEF
- Liberty (.lib)
- Netlists (.v)

→ Path FX

→ Reordered Path Reports

RedHawk-SC

Instance DvD values of selected scenarios/critical timing paths

Highlights

1. Characterize FX libraries once
2. Standard inputs for timing/simulations
3. Comprehensive scenario/critical timing path coverage with ML

New Path Report

- SPICE accurate slack report
- 100x faster than MC SPICE
- Captures process and voltage variability
Model Performance of Scenario and Path Predictor for 2M Timing Paths and 20 Scenarios as Training/Validation/Test Data - Metrics

- For Bin-based Classification Step:
  - **Accuracy** = \(\frac{\text{Number of correct predictions}}{\text{Total number of predictions}}\)
  - **Confusion Matrix**

<table>
<thead>
<tr>
<th></th>
<th>Bin 1</th>
<th>Bin 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bin 1</td>
<td>90</td>
<td>10</td>
</tr>
<tr>
<td>Bin 2</td>
<td>50</td>
<td>150</td>
</tr>
</tbody>
</table>

Bin 1 Accuracy = 90 / (90+10) = 90%
Bin 2 Accuracy = 150 / (50+150) = 75%
Avg. Accuracy = (90+150) / (90+10+50+150) = 80%

- For Regression Step:
  - **MSE** : Mean Squared Error (the smaller, the better)
  - **R2** : R-squared (between 0~1, best:1)

- For Bin-based Ranking:
  - **NDCG** (Normalized Discounted Cumulative Gain: between 0~1, best:1) for both XGBoost and lambdaMART methods
Model Performance of Scenario Predictor

- Top-1 bin accuracy and average accuracy on testing set of Bin-based classification model (5 bins) in Scenario Predictor:

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Top-1 bin Accuracy(%)</th>
<th>Average Accuracy(%)</th>
<th>Training Time (minutes)</th>
<th>Testing Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN</td>
<td>90.1</td>
<td>80.2</td>
<td>1.1</td>
<td>2.2</td>
</tr>
<tr>
<td>Random Forest</td>
<td>92.3</td>
<td>86.5</td>
<td>16.3</td>
<td>0.5</td>
</tr>
<tr>
<td>XGBoost</td>
<td>92.8</td>
<td>88.4</td>
<td>17.5</td>
<td>1.8</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>92.5</td>
<td>82.5</td>
<td>228.7 (GPU)</td>
<td>0.6</td>
</tr>
</tbody>
</table>

- MSE, R-squared and NDCG score on testing set of Ranking model:

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>MSE</th>
<th>R-squared</th>
<th>NDCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>XGBoost</td>
<td>0.037</td>
<td>0.902</td>
<td>0.79</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>0.044</td>
<td>0.883</td>
<td>0.77</td>
</tr>
<tr>
<td>LambdaMART</td>
<td>-</td>
<td>-</td>
<td>0.77</td>
</tr>
</tbody>
</table>
Model Performance of Scenario (Weighted Algorithm) and Path Predictor

- Weighted algorithm on ground truth and predicted data:

<table>
<thead>
<tr>
<th>Rank Position</th>
<th>On Ground Truth (Scenario_&lt;index&gt;)</th>
<th>On XGBoost Predicted Data</th>
<th>On LambdaMART Predicted Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scenario_3</td>
<td>Scenario_3</td>
<td>Scenario_3</td>
</tr>
<tr>
<td>2</td>
<td>Scenario_15</td>
<td>Scenario_15</td>
<td>Scenario_12</td>
</tr>
<tr>
<td>3</td>
<td>Scenario_12</td>
<td>Scenario_12</td>
<td>Scenario_15</td>
</tr>
<tr>
<td>4</td>
<td>Scenario_18</td>
<td>Scenario_18</td>
<td>Scenario_18</td>
</tr>
<tr>
<td>5</td>
<td>Scenario_2</td>
<td>Scenario_11</td>
<td>Scenario_13</td>
</tr>
</tbody>
</table>

- Top-1 bin accuracy and Average Accuracy on testing set of Bin-based classification model (5 bins)

  **in Path Predictor**: XGBoost has the best accuracy with reasonable training/testing time

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Top-1 bin Accuracy(%)</th>
<th>Average Accuracy(%)</th>
<th>Training Time (minutes)</th>
<th>Testing Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KNN</td>
<td>89.5</td>
<td>78.1</td>
<td>&lt; 1</td>
<td>3.1</td>
</tr>
<tr>
<td>Random Forest</td>
<td>92.5</td>
<td>86.1</td>
<td>17.1</td>
<td>0.6</td>
</tr>
<tr>
<td>XGBoost</td>
<td><strong>93.3</strong></td>
<td><strong>89.3</strong></td>
<td>19.8</td>
<td>2.1</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>93.2</td>
<td>83.8</td>
<td>237.2 (GPU)</td>
<td>0.7</td>
</tr>
</tbody>
</table>
Model Performance of Scenario/Path Predictor on Accuracy Using XGBoost

- **XGBoost** – 5 bins of Scenario Predictor and Path Predictor

**Step 1 of Scenario Predictor:**
Bin-based Classification Model
(17 Features)

**Path Predictor:**
Bin-based Classification Model
(19 Features)
Conclusions

- ML-based Timing Assistant can predict top-ranked switching scenario(s) from hundreds of possible scenarios for continuing dynamic voltage drop analysis and top-ranked timing paths of selected switching scenario(s) with DvD analysis for continuing accurate/fast timing simulation including DvD waveforms on the Vdd/Vss pins for final timing sign-off.

- The custom path-based features identified for training the inference engines (i.e. Scenario and Path Predictor) in Timing Assistant include possibly path-based effective P/G grid resistance from bumps to instance, combo peak current ($I_{peak}$) considering rail-connected instances, path-based timing sensitivity due to voltage drop, and combined input feature from feature engineering.

- XGBoost used in both Scenario/Path Predictors have the best accuracy (> 93%) with reasonable running time.
谢谢