Google

Machine Learning based Dynamic IR hotspot estimation for SoC Designs

Prateek Pendyala, Jingwei Zhang &
T Govindaswamy Rahul Sai

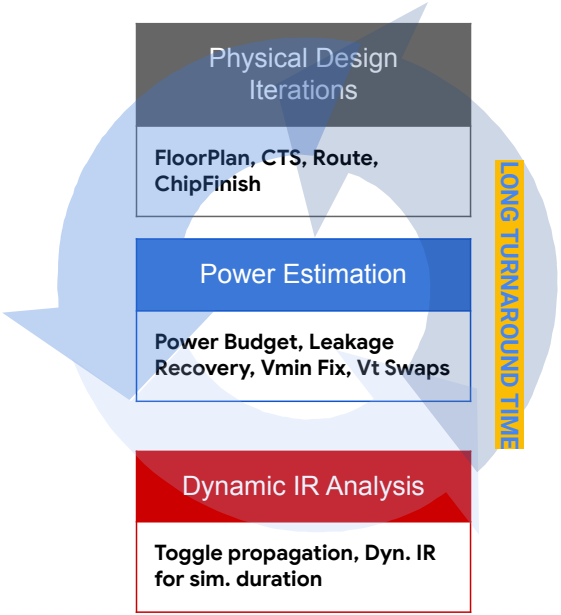
62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm

Background and Problem Statement

SPONSORED BY CEDA IEEE Council on Electronic Design Automation SIGDA ACM

Challenge: Long IR Drop Analysis Time



- Dynamic IR drop estimation is obtained by commercial tools, which are accurate but very time-consuming as these require simulating billions of nodes. Runtime complexity is N^2 for N: being number of instances in an industrial design
- VCD simulation patterns are unavailable during early design stage. Also, many vectors are needed to run for many cycles to ensure coverage. Vectorless IR which logic propagates toggle activity, is mostly employed to cover this gap also suffers from long runtimes
- Objective of this work is to solve vectorless propagated dynamic IR estimation of multi-million node power grids as well as identification of local hotspots using a quick & accurate machine learning model

Our Methodology

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm

Outline

1. Build and test machine learning models using existing algorithms on a smaller problem: incremental static IR drop prediction

2. Analyze the performance of above models in context of static IR drop prediction, and to deploy a similar framework for solving vectorless dynamic IR prediction

3. Implement previously adapted machine learning techniques presently in use in industry to analyze possible areas of optimization & for performance benchmarking

4. Test our proposed machine learning model & flow for dynamic IR estimation on incremental modifications in design

Solving a simpler problem first: Static IR drop prediction

- We implemented Tree-based XGBoost (Scikit-learn)

Design partitioned into 50ux50u & IR is estimated for tile

- Resistance feature set:

```
'cell_name','eff_res_vdd','eff_res_vss','closest_bump_res_vdd', 'closest_bump_res_vss'
```

- Power feature set:

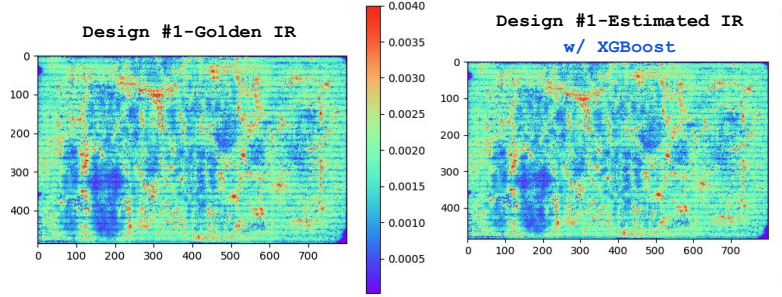
```
‘instance_total_power','sum_of_instance_power_in_tile','sum_of_instance_power_top_tile','sum_of_instance_power_bot_tile','sum_of_instance_power_left_tile','sum_of_instance_power_right_tile','sum_of_instance_power_NE_tile','sum_of_instance_power_NW_tile','sum_of_instance_power_SE_tile','sum_of_instance_power_SW_tile','x_tile','y_tile','sum_of_instance_power_row_above','sum_of_instance_power_row_below'
```

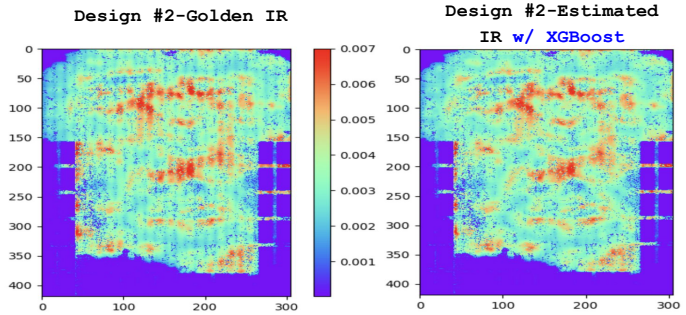
- Label:

```
‘instance_ir_drop'
```

- Results:

Circuit Design	Instance Count#	RMSE (mV)	Correlation Coeff.	Golden IR runtime	Feature Ext.+Inference runtime
Design #1	7367516	XGBoost: 0.000027	XGBoost: 0.92	58 mins @ 156 workers	XGBoost: <5 mins @ 1 worker (run on same machine as Golden IR)
Design #2	1883416	XGBoost: 0.000073	XGBoost: 0.941	38 mins @ 156 workers	XGBoost: <3 mins @ 1 worker (run on same machine as Golden IR)





Solving a simpler problem first: Static IR drop prediction (contd.)

- We implemented U-Net Image Classification Neural Network (PyTorch)

Design partitioned into gcells & IR is estimated for gcell

- 2D Resistance feature map (256x256):

`'eff_res_vdd','eff_res_vss'`

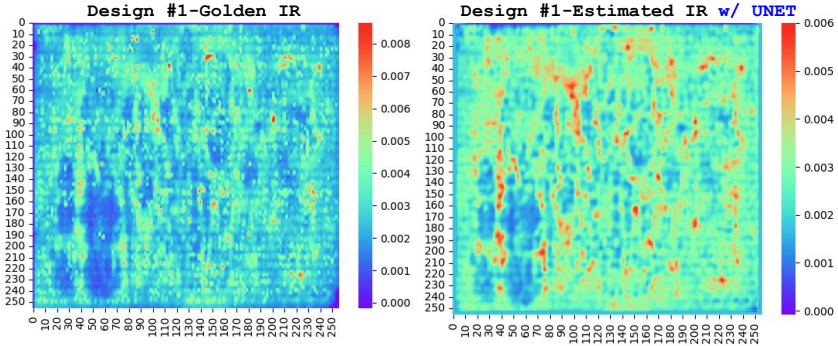
- 2D Power feature map (256x256):

`'total_instance_power'`

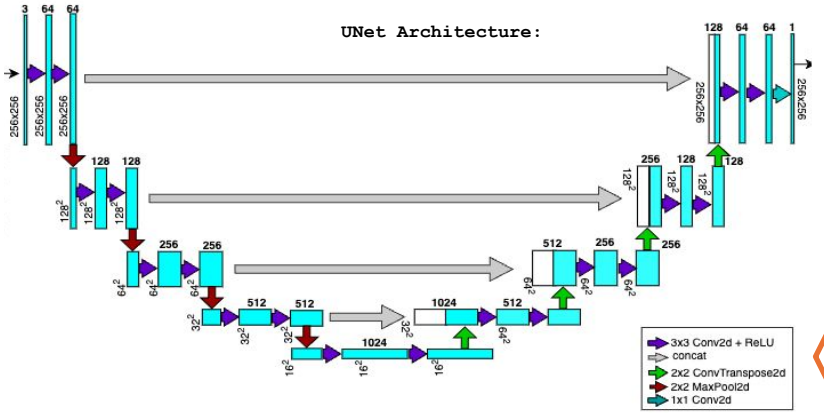
- 2D Label:

`'Instance_ir_drop', 'instance_count'`

- Results obtained:



Circuit Design	Instance Count#	RMSE (mV)	Correlation Coeff.	Golden IR runtime	Feature Ext.+Inference runtime
Design #1	7367516	UNet: 0.0006	UNet: 0.876	58 mins @ 156 workers	UNet: <3 mins @ 1 worker (run on same machine as Golden IR)
Design #2	1883416	UNet: 0.0007	UNet: 0.908	38 mins @ 156 workers	UNet: <2 mins @1 worker((run on same machine as Golden IR)



Related: Re-Implemented state-of-art Dynamic IR drop prediction

- We implemented CNN based (PyTorch) *
  - 2D Power feature map (256x256):

```
'internal_power','switching_power','leakage_power','overall_power'
```

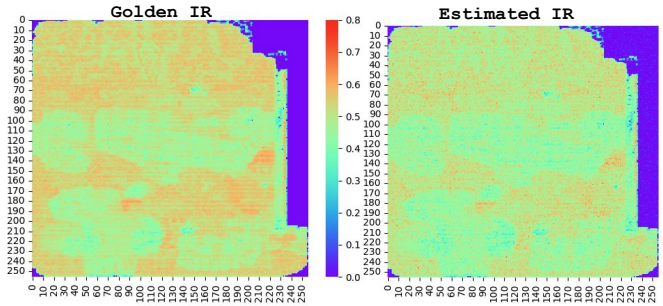
  - 3D Power feature map (time-sliced, 256x256x20) :

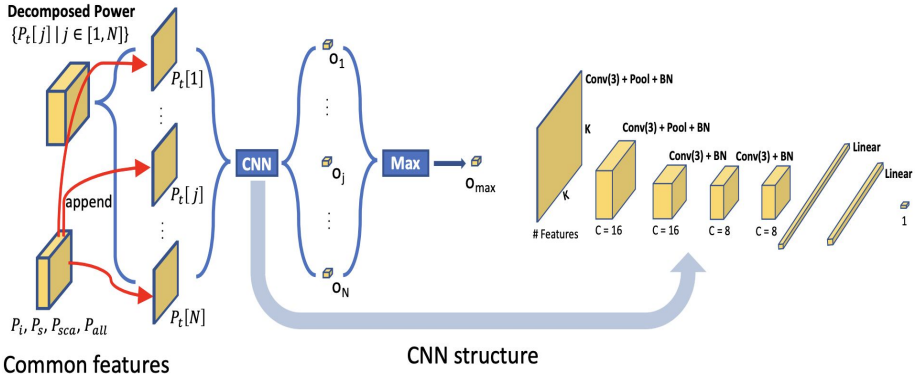
```
'2D_scaled_overall_power_0t',...,'2D_scaled_overall_power_20t'
```

  - Label 2D map:

```
'instance_DIR_drop'
```

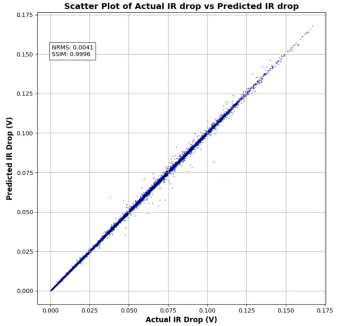
  - Results:





Common features

CNN structure



* Xie, Zhiyao, et al. "PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network." 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2020.

Findings based on key results

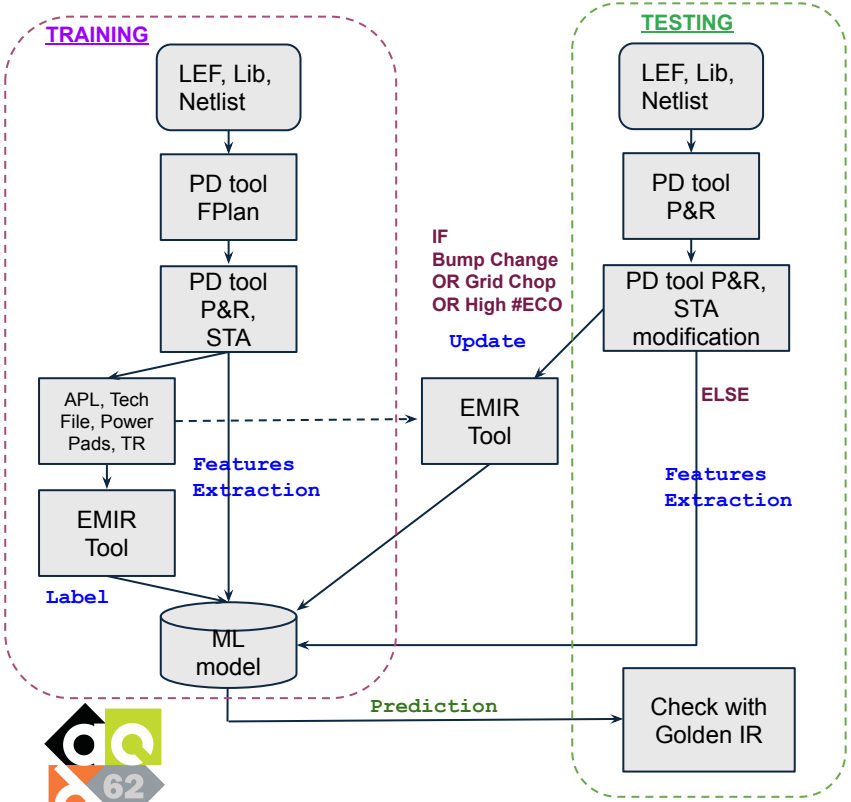
1. While trying to solve static IR prediction for the same, **we observed XGBoost to have better overall correlation and lower RMSE, as compared to image-based UNet approach**

2. Also, we tried to re-implement a popular dynamic IR prediction which uses CNN approach. Our observations were as follows:
   a. Although, this model is not specific for a design, **it is not very intuitive to guide physical designer while implementing small-to-medium ECO changes**
   b. It reads 2D image-based arrays of time-based overall power maps- some of which are very sparse. This might force a very optimistic dynamic IR drop inference.
   c. The overall feature extraction time to parse DEF, timing window file, power reports to arrange training dat2D & 3D feature & label tensors input to model is very time consuming (even after parallelizing). Especially for designs >2M, **the feature extraction time itself starts to take #hours**
      - Thus, for a good speedup, intuitive prediction response & accuracy performance: **We chose XGBoost Approach with a comprehensive feature set for solving Dynamic IR drop prediction**

Proposed Flow & Details

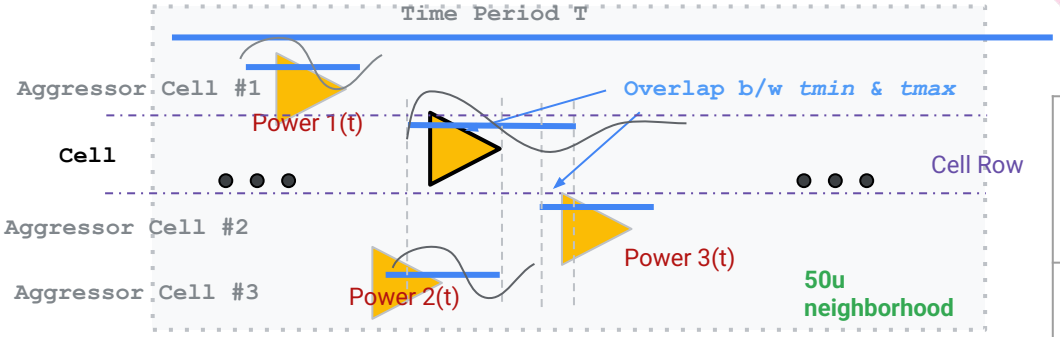
SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm

Training & Testing Flow: Overview



- **Input:** .DEF, .LEFs, .APLs, timing window file, Power Pad (.plocs), EMIR instance voltage stats
- **Output:** A learning-based model to predict dynamic IR drop
- **Training:** Extract relevant & unique power features, timing features from reference & N-1, N-2 versions
- **Objective:** Minimize Root Mean Absolute Error (RMSE)
- **Testing:** Extract same features for test-case (DUT)
- ML model might need an update in case PG changes (bump movement, major grid cuts done in DUT)

Input Features: Description



'cell_type','self_avg_power','self_peak_power','peak_switching_time','slew','load_cap','cell_delay'	Self drop Features of aggressors per instance
'agg_on_top','agg_at_bottom','agg_on_left','agg_on_right','agg_avg_power_5u','agg_peak_power_5u','agg_peak_power_in_5u_row','agg_avg_power_in_5u_row','agg_peak_power_50u','agg_avg_power_50u','agg_peak_power_in_50u_row','agg_avg_power_50u_in_row'	Dynamic Power Features of aggressor drop per instance
'arrival_time_5u','arrival_time_50u','cap_5u','cap_50u'	Arrival Time & Capacitance Features of aggressor drop per instance

Algorithm

1. Read all process node .lef files

2. Load timing session & extract cell_name, worst slew, cell delay, load cap, freq, op. voltage ∀ instances
3. Read in .DEF, .ipf & use (1) to extract cell_loc(x,y), gcell(x,y), total_avg_power p_{sca} = toggle_rate *(p_{int} + p_{swi}) + p_{lkg} ∀ instances

4. Determine peak current ∀ switching cells using slew, op. voltage from (2) & (3) interpolated from *.apl current* files
5. Determine #switching instances & query peak current time & peak power for all
6. For each gcell(x,y):
   For each switching instance in it:
   a. add avg. power & peak power of neighbor inst in (5u, 50u regions) with overlapping timing window (tmin & tmax- both for rise & fall)
   b. expand time window tail-end by +5, +50, +100 ps (assuming F_{max}> 1Ghz) & repeat (a)
   c. add cap for non switching cells (from *.apl cap* data)
   d. skip instances with missing timing window

FEATURES EXTRACTION

6. Concat all input features in (2)–(6), to get an accurate spatial & temporal representation

7. Initiate XGBoost model X with Loss function J
   for ∀ decision-trees ∈ [1, N]:
   shuffle j ∈ inst:
   Y=X(j)
   Gradient Descent -= ∇J(Y,IRj])

TRAINING

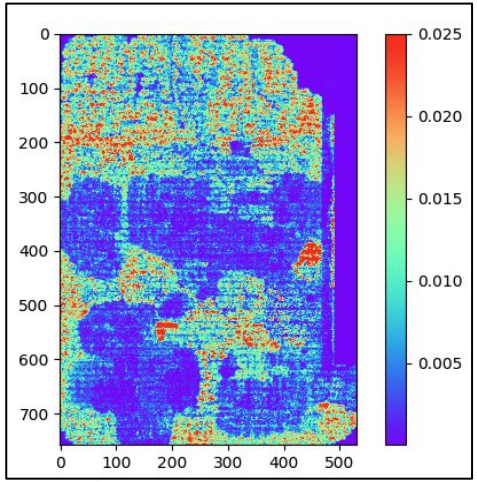
PREDICT

8. Inference for instances in DUT using Trained model X from (6), if no major changes to grid in DUT as compared to training databases

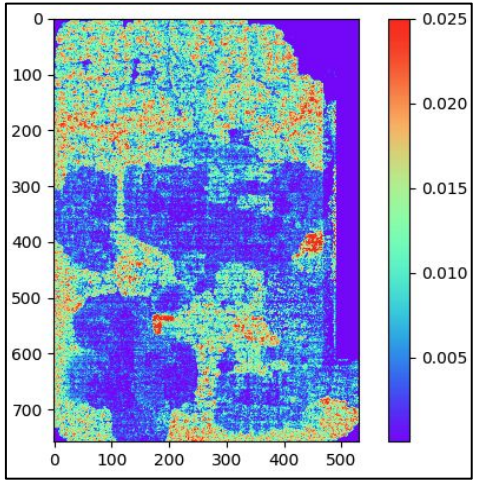
Incremental IR Results on smaller ECO (# 500 cell additions)

Circuit Design	Instance Count#	RMSE (mV)	Correlation Coeff.	Golden IR runtime	Feature Ext.+Inference runtime
Design #1	7367516	0.029 mV	0.76	4 hours, 38 mins @156 workers	~15 mins @1 worker (run on same machine as Golden IR)
Design #2	1883416	0.039 mV	0.84	1 hour, 58 mins @156 workers	~5 mins @1 worker (run on same machine as Golden IR)

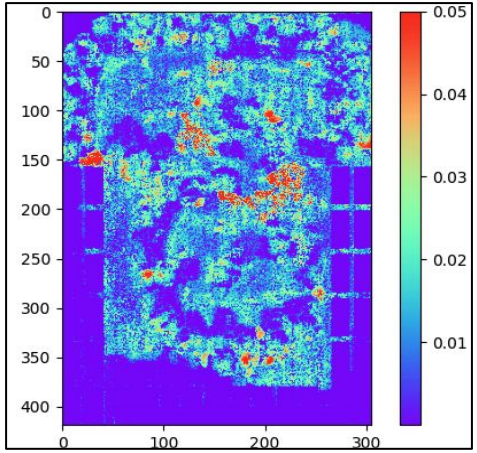
Design #1-Golden IR



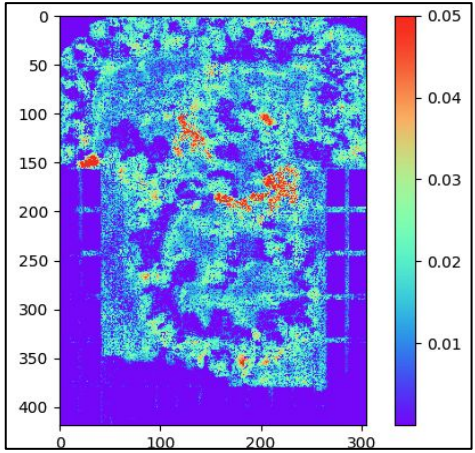
Design #1-ML Estimated IR



Design #2-Golden IR

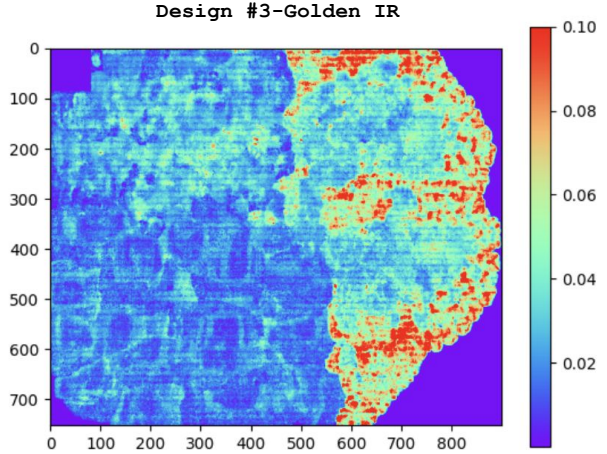


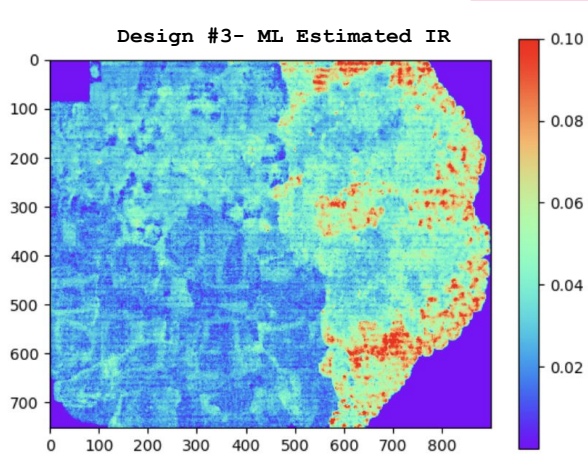
Design #2-ML Estimated IR

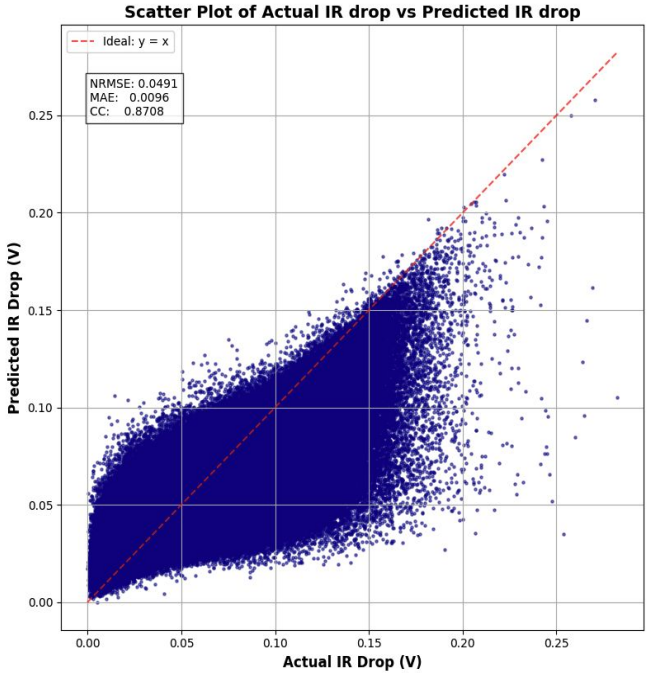


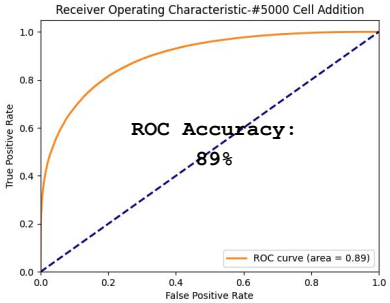
Incremental IR Results on larger ECO (#4000 cell additions)

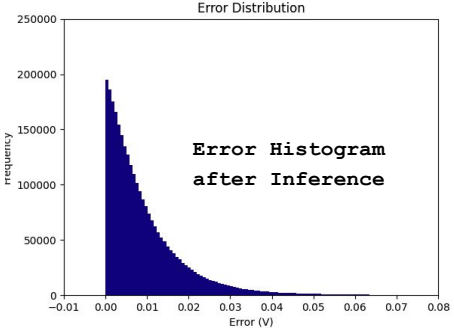
- Runtime is ~5 mins for Feature Extraction+Inference Time (Design #3 Inst Count: 2.9M)











- MAE ~9mV between actual vs predicted IR drop

Conclusion

SPONSORED BY CEDA IEEE Council on Electronic Design Automation sigda acm

Summary & Comparison with Recent Work

- As results show, as tested on industrial SoC designs with >2M instance count, the proposed pretrained ML model estimates full-chip dynamic IR with a speedup of >15x & a good accuracy of <1 mV Root Mean Square Error with low to moderate ECOs. This accuracy however drops when large ECOs (order of 1000s) to ~ 9mV Mean Average Error.

- Correlation Coefficient of ~0.88 with a receiver operating curve accuracy of ~90.0 area indicates very close approximation of predicted vs actual IR drop (calculated from golden IR signoff tool) for three test designs as can be observed from scatter plots

Method	Type of IR	ML model	Features Used	Objective
PowerNet: Transferable Dynamic IR Drop Estimation via Maximum Convolutional Neural Network [2020]	Vectorless Dynamic IR	Convolutional Neural Network	power switching, toggle rate, timing window	IR mitigation
A Fast and Accurate Per-Cell Dynamic IR-drop Estimation Method for At-Speed Scan Test Pattern Validation [2012]	Scan shift Dynamic IR	Linear Regression	power switching	Scan shift IR mitigation
Machine-learning-based Dynamic IR Drop Prediction for ECO [2018]	Vectored Dynamic IR	Convolutional Neural Network + XGBoost Regression	power switching, toggle rate, timing window, resistance, toggle rate, cell count	IR mitigation
This work	Vectorless Dynamic IR	XGBoost Regression	42 power & timing features, mentioned in Slide(3). No resistance features used	PG variant regression analysis for optimal PG

Prateek Pendyala

Silicon Physical Design Engineer, Power Grid Methodology

Google

AI

Security

Systems

EDA

Design

62 THE CHIPS TO SYSTEMS CONFERENCE

SPONSORED BY CEDA® IEEE Council on Electronic Design Automation

sigda acm