Problem C: Static IR Drop Estimation Using Machine Learning

Gana Surya Prakash Kadagala and Vidya A. Chhabria
Arizona State University
Supported by: Steel Perlot and the OpenROAD Project

0. Version History

First version: 3/23/2023
Second version: 5/31/2023 : Benchmarks released on website. Fake and real circuit benchmarks added.Please use these benchmarks for the contest.
Third version: 6/12/2023

(a) Clarification on .sp dbu units for information to covert to .csv file.
(b) Fake and real circuit benchmarks on the website are updated to eliminate those with high values of voltage drop and negative voltage drop. Testcase11 and Testcase12 updated in real circuits and 12, 18, 21, 29, 31, 26, 42, 44, 62, 63, 67, 69, 96 updated in fake circuits.

Fourth version: 6/22/2023

(a) Clarification on evaluation metrics with example
(b) Removal of duplicate files in fake circuit benchmarks
(c) Update Python version to 3.6
(d) Update on executable command: output file path is an argument to the executable to generate the output predicted csv.

1. Introduction

Power delivery network (PDN) analysis is crucial to successful IC design closure, particularly for designs implemented in lower technology nodes that suffer from large wire parasitics and high power densities. The on-chip PDN is responsible for transmitting voltages and currents to each cell in the design. However, due to the parasitics in the PDN, a voltage drop is induced between the power pads and the cells in the design. Large voltage drops in the PDN can hurt chip performance and, in the worst case, its functionality. Consequently, it is essential to check that the worst-case IR drop values in the PDN are within specified limits.

A structure of the on-chip PDN is shown in Fig. 1(a). The PDN can be modeled as a network of voltage sources, current sources, and resistances, where the wires are a network of resistances, the power pad (C4 bumps) are voltages sources connected to the PDN wires, and the current sources are the cells/instances that draw current as shown in Fig. 1(b).

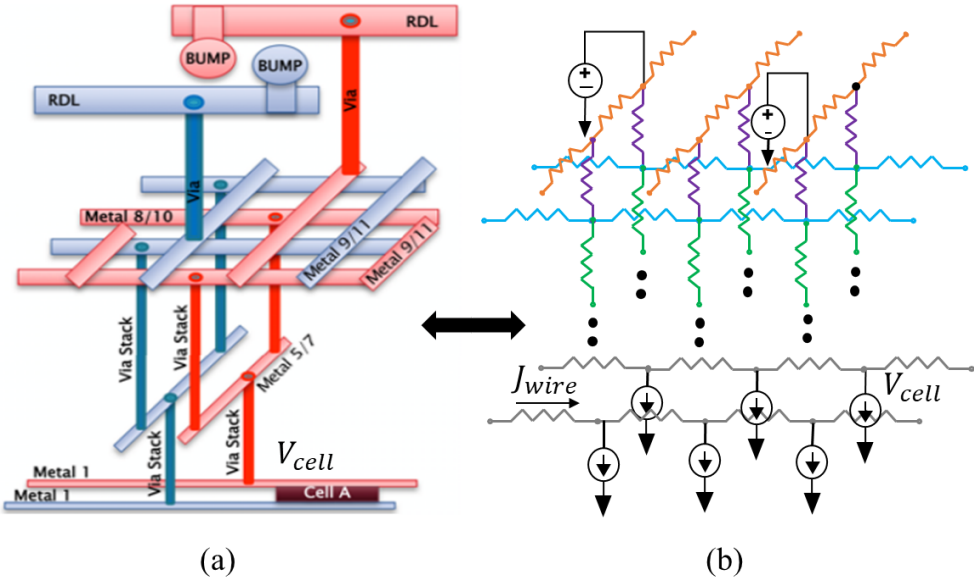

Fig 1. PDN structure and model

The goal of IR drop simulation is to find the voltage at the nodes of the PDN that connect to the instances (current sources). Traditionally, finding the voltage at every node amounts to solving a system of linear equations of the form $GV = J$ where G is a conductance matrix, V is the unknown vector of voltages, and J is the vector of currents [1]. However, solving this system of equations is computationally very expensive with millions of PDN nodes.

This contest aims to bypass the computational challenge of static IR drop using machine learning (ML) techniques. ML algorithms have previously found some success in addressing this problem [2, 3, 4, 5, 6]. In this context, the contest aims to serve three goals:

1) Lower the barriers to entry for non-EDA experts by converting a traditional EDA problem to an ML solvable problem and incentivizing the use of novel ML techniques to improve accuracy.
2) Explore the use of transfer learning to address the limited dataset problem in the EDA community.
3) Establish a state-of-the-art ML model for IR drop prediction.

Contest Objective: The contestants need to train an ML model to predict static IR drop with the highest possible accuracy (mean absolute error) and F1 score on the test data with the least inference runtime and model size.

2. Background:

The static IR drop distribution across the chip depends on the following three factors:

1) The location/distributions of all voltage sources (power pads) in the design
2) The topology of the PDN and resistance values of each resistor (via/metal layer) in the network
3) The distribution of current sources (power) in the design

Prior ML techniques have modeled the above inputs as images [2, 3, 6] and used image-based ML models to predict IR drop. For example, [2] represents the above three features as three distributions where the current source distribution is modeled as a current map (Fig. 2(a)); the PDN topology is modeled as a function of the density (spacing between power stripes per unit area) of the power grid (Fig. 2(b)) in different regions ,and the voltage source distribution is modeled as an effective distance map which is the distace from each PDN node to the PDN node with the voltage source (Fig. 2(c)). The output is a distribution of IR drop at every node in the lowermost layer of the PDN, which can be represented as an IR drop map (Fig. 2(d)). Using such image-based representations, prior techniques have used CNNs and U-Nets, to perform IR drop prediction using a model trained on a diverse set of such data points (three inputs and one output). The trained ML model can perform inference on unseen testcases.

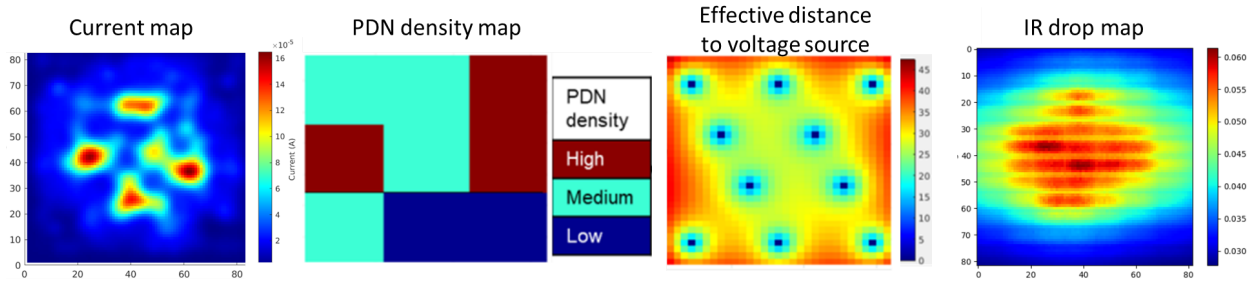

Fig. 2: Current map, PDN density, voltage source, and IR drop distributions across the die.

While these techniques have found success, another underlying challenge exists – the unavailability of sufficient training data to build these models. The work in [8] used generative adversarial models to generate thousands of fake yet realistic current maps. With the availability of golden ground-truth open-source $GV = J$ static IR drop solvers, [8] can create a large training dataset for the static IR drop problem for different configurations of power grids and voltage sources.

For this contest, we are leveraging the fake dataset from [8] to create a large training dataset and performing test (inference) on real-circuit designs. There will also be a few data points of real circuits in the training dataset. One suggestion is for contestants is to use transfer learning, where the initial models are trained using fake data and weights are fine-tuned using real circuit training data, and then the model is tested on validation data (real circuit data only).

3. Input and Output Formats

We will provide the data in two formats:

(a) Image-based inputs and outputs: These are direct image-based representations of the three inputs and the IR drop output, as described in Section 2. These can serve as a direct dataset for the application of image-related ML models (such as CNNs, U-Nets, etc.)

(b) SPICE-based inputs and outputs for the detailed structure of PDN and additional features: These are SPICE netlists that also provide the value of each resistor in the PDN resistance network. These inputs can be used to create graph-based ML models (such as GNN, GCNs, etc.)

(a) Image-based data: The image-based data will be provided as a matrix in .csv format where every value in the file represents an element in the current, effective distance, PDN density matrices, and voltage drop of the PDN node in the lowermost metal layer in a 1um x 1um area of the chip represented as a matrix (Note than the PDN node pitch in the lowermost layer will be larger)
E.g.: A 80um X 80um area chip will consist of 80x80 values in the current_map.csv

```
4.43E-08    5.83E-08    8.05E-08    1.02E-07    . . . .     1.15E-07

7.93E-08    1.04E-07    1.44E-07    1.82E-07    . . . .     2.05E-07

1.54E-07    2.03E-07    2.80E-07    3.54E-07    . . . .     3.97E-07

2.71E-07    3.56E-07    4.92E-07    6.21E-07    . . . .     6.95E-07

.           .           .           .           .           .
.           .           .           .           .           .
.           .           .           .           .           .


4.14E-07    5.45E-07    7.54E-07    9.53E-07    . . . .     1.07E-06
```

The above matrix can be represented as an 80x80 image where every pixel represents the current in a 1um x 1um region. A similar format is used for other input effective_distance.csv, pdn_density.csv, and output voltage_map.csv The corresponding voltage_map.csv

```
2.91E-02    2.95E-02    2.99E-02    3.02E-02    . . . .     3.07E-02

2.91E-02    2.96E-02    3.00E-02    3.03E-02    . . . .     3.08E-02

2.92E-02    2.97E-02    3.00E-02    3.04E-02    . . . .     3.09E-02

2.93E-02    2.97E-02    3.01E-02    3.05E-02    . . . .     3.10E-02

.           .           .           .           .           .
.           .           .           .           .           .
.           .           .           .           .           .


2.93E-02    2.98E-02    3.02E-02    3.06E-02    . . . .     3.11E-02
```

Example input features and output label (plotting the provided matrices in the .csvs) is shown in Fig. 2.

Dataset and its diversity in the provided benchmarks:
The size of the matrices can vary across the dataset as they represent chips of different sizes. However, a single datapoint will have all four matrices (current, PDN, voltage source, and IR drop) of the same size.
(i) Current map: These will vary in maximum, average, and peak values with different hotspot locations.

(ii) PDN density map: The PDN will be a region-wise uniform PDN where the density in a specific die region is constant. Each region can use one of three possible densities. Across the training and test dataset, all data points will use a combination of the three PDN densities in different regions [3].
(iii) Effective distance to voltage source map: The number of voltage sources can vary, and the distribution of the voltage sources can also vary across the die.
(iv) IR drop maps: These are the ground-truth output voltages obtained by solving $GV = J$ system of linear equations using [8].

(b) SPICE-based data: In addition to the above data, for each datapoint we also provide the PDN model as shown in Fig. 1(b) as a SPICE netlist. The SPICE netlist encodes the node locations, the value of resistances between nodes, the current source nodes and their locations, and the voltage source nodes and locations. The format is described below with the following SPICE netlist snippet:

```
R645 n1_m1_108000_17920 n1_m1_102600_179200 0.14
R646 n1_m1_113400_179200 n1_M3_113400_179200 4.23
I7 n1_m1_113400_179200 0 4.24901e-08
V0 n1_m7_81000_106230 0 1.1
```

The generic convention is <electric component> <node1> <node2> <value>.

The node is defined using the following convention:
<netname>__<layer-idx>_<x-cordinate>_<y-cordinate>

In the above example, R646 is via since they share the same x and y coordinate but the layer changes. I7 is a current source connected to the node at location (113400, 179200), and V0 is the voltage source at node (81000, 106230). The locations in the spice file can be converted to the csv file by using a 2000dbu. For instance in the above example, 113400 is location 113400/2000 = 56.7 um. The matrix is in the .csv files are at a 1um granularity.

The SPICE netlist has all the information needed for the three features, and the labels for IR drop are in the form of the csv as described in the previous image-based data section.

Required Output Format:
For fair evaluation, all contestants must generate outputs as voltage drop matrices (.csv) of the same size as the input matrices (even if the features are taken from SPICE netlist and not the image-based data).

Benchmarks to be used:

Please use the benchmarks attache as a part of the contest. http://iccad-contest.org/Problems.html

4. Evaluation
The contestants will be provided with training (hundreds of fake data points and few real-circuit data points). However, the contestants' ML models will be evaluated on an unseen test dataset (real circuit data) and will be evaluated for the following metrics:

(a) Mean absolute error (MAE): The average of the absolute difference between a prediction and the actual value, calculated for each example in a dataset. The goal is to have a low normalized MAE. This is the difference between the predicted and original value csvs.

Example:
Predicted IR drop matrix is:

```
4.43E-03    5.83E-03

5.93E-03    1.04E-03
```

Actual IR drop matrix is:

```
4.63E-03   5.23E-03

5.93E-03   0.04E-03
```

MAE matrix is:

```
0.2E-03    0.6E-03

  0          1E-03
```

MAE = 0.45mV

(b) F1 score: A binary classification metric that uses 10% of the maximum IR drop of each benchmark as the threshold to perform classification.

$$F1\ score\ =\ 2\ *\ Precision\ *\ Recall\ /\ (Precision\ +\ Recall)$$
$$Precision\ =\ TP\ /\ (TP\ +\ FP)$$
$$Recall\ =\ TP\ /\ (TP\ +\ FN)$$

Where TP = True positive, FP = False positive, TN = True negative, and FN = False negative. The positive class is nodes with the top 10% of IR drop. The goal is to have a high F1 score. An element in the ir_drop_map.csv is considered as an IR drop hotspot if its value is greater than 90% of the maximum ir drop of that testcase. For the above example in (a), we define a hotspot matrix as an element which has an IR drop value larger than 90% of 5.93mV = 5.337mV. Please note that the maximum value will be different for each testcase

The actual hotspot matrix for the actual IR drop matrix above in (a):

```
0          1

1          0
```

The predicted hotspot matrix for the predicted IR drop matrix above in (a):

```
0          0

1          0
```

Therefore, TP = 1; TN = 2; FP = 0; FN = 1

$Precision = TP / (TP + FP)$ = 1/(1) = 1;

$Recall = TP / (TP + FN)$ = 1/(2) = 0. 5;

$F1\ score = 2 * Precision * Recall / (Precision + Recall)$ = 1/1.5 = 0.67

(c) Run time: Inference time of the ML model. The goal is to have low runtime and a fast inference.

The overall score is a function of MAE, F1 score, and inference runtime. We will use a 60% weightage to MAE and 30% to F1 and 10% to runtimes. Each participating team will be given a score for each metric. For each hidden testcase, the team with the best MAE will receive 60 points for that testcase, the team with the best F1 score will receive 30 points for that testcase, and the team with the best runtime will receive 10 points for that testcase. The scores for the rest of the teams will be scaled based on a normal distribution for each metric (relative scoring). This will lead to a single score for each testcase (out of 100) for each team. The scores will be added across all testcases and the top three teams with the highest will win the contest. Since there are 10 hidden testcases, the maximum score a team can get is 1000 points if they have the best MAE, F1 score and runtime across ALL testcases.

5. Submission Details and Platform

Participants are recommended to use **python3.6** for all software development. The submission should contain the following items:

- Executable file
- Requriements.txt
- Model checkpoint
- README (optional)
- Source code (optional): Needed for any open-source bonus prize money

(a) Binary executable file:

Package your Python script into a standalone binary executable to facilitate evaluation. Follow these steps:

- Step 1: Make sure this script takes 5 input files (i_map.csv, pdn_density.csv, eff_dis.csv, xyz.sp and chk_pnt.file) as input arguments and generates a single output file ir_drop_map.csv. If you code does not use .sp file as input you can make this argument optional.
- Step 2: Use a packaging tool such as PyInstaller, py2exe, or cx_Freeze to create the binary executable. Refer to the documentation of your chosen tool for detailed instructions.
- Step 3: Use your requriements.txt file to create environment. Test the generated binary executable on your own in a new system to ensure it works as expected with following command.

```
./ir_predictor.exe        current_map.csv        pdn_density.csv
eff_dist_map.csv             netlist.sp             chk_pnt.file
path_to_output_predicted_ir_drop.csv
```

The chk_pnt.file is the name of your checkpoint of your trained ML model that will be used to perform inference on unseen/hidden designs. The last argument is the path for the generated predicted output csv.

(b) Requriements.txt (requirements.txt):

This file can be automatically generated from the virtual envvironemtn in which you are developing your IR drop predictor by using pip3 freeze > requirements.txt

Sample example of file contents:

```
numpy==1.19.5
torch==1.7.1
```

This file will be installed by doing pip3 install requirements.txt when evaluating your submission in the organizers virtual environment

(c) Model Checkpoint File (chk_pnt.file)

Include your trained model checkpoint file.. This file should contain the trained model weights or parameters in whatever format you prefer as long as your executable binary reads this file.

(d) README

Clear instructions on inferring your model using executable on one sample case with step-by-step process. Expectation is that this process should match our submission guidelines

6. Platform and Access to GPUs for Training models

More information on access to Google cloud will be provided to contestants after the alpha submission as we cannot support compute resources for all registered teams currently.

6. References

[1] Y. Zhan, S. V. Kumar, and S. S. Sapatnekar, “Thermally-aware design,” Found. Trends Electron. Des. Automat., vol. 2, no. 3, pp. 255–370, March 2008.

[2] V. A. Chhabria, V. Ahuja, A. Prabhu, N. Patil, P. Jain, and S. S. Sapatnekar, “Thermal and IR Drop Analysis Using Convolutional Encoder-Decoder Networks,” Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC), 2021.

[3] Chia-Tung Ho and Andrew B Kahng. “IncPIRD: Fast Learning Based Prediction of Incremental IR Drop,” in the IEEE/ACM International Conference on Computer-Aided Design (ICCAD). 2019.

[4] Zhiyao Xie, Haoxing Ren, Brucek Khailany, Ye Sheng, Santosh Santosh, Jiang Hu, and Yiran Chen, “PowerNet: Transferable Dynamic IR Drop Estimation via Maximum Convolutional Neural Network” in Asia and South Pacific Design Automation Conference (ASP-DAC), 2020.
[5] Chi-Hsien Pao, An-Yu Su, and Yu-Min Lee, “XGBIR: An xgboost-based IR drop predictor for power delivery network,” Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020.
[6] V. A. Chhabria, Y. Zhang, H. Ren, B. Keller, B. Khailany, and S. S. Sapatnekar, “MAVIREC: ML-Aided Vectored IR-Drop Estimation and Classification,” Proceedings of Design, Automation, and Test in Europe (DATE), 2021.
[7] V. A. Chhabria, A. B Kahng, M. Kim, U. Mallappa, S. S. Sapatnekar, and B. Xu, “Template-based PDN Synthesis in Floorplan and Placement Using Classifier and CNN Techniques,” Proceedings of Asia and South Pacific Design Automation Conference (ASP-DAC), 2020.
[8] V. A. Chhabria, K.Kunal, M. Zabihi, and S. S. Sapatnekar, “BeGAN: Power Grid Benchmark Generation Using a Process-portable GAN-based Methodology,” Proceedings of IEEE/ACM International Conference On Computer Aided Design (ICCAD), 2021.
[9] V. A. Chhabria and S. S. Sapatnekar, “PDNSim,” 2021, https://github.com/The-OpenROAD-Project/OpenROAD/tree/master/src/psm.