Detailed Routing Violation Prediction During Placement Using Machine Learning

Aysa Fakheri Tabrizi*, Nima Karimpour Darav*, Logan Rakai*, Andrew Kennings†, and Laleh Behjat*
*Department of Electrical and Computer Engineering, University of Calgary, Canada
†Department of Electrical and Computer Engineering, University of Waterloo, Canada

***Abstract*—The complexity of design rules at $22nm$ and below precludes direct incorporation of detailed routing (DR) rules into a placement algorithm. However, ignoring routability rules during the placement process may result in infeasible designs. The congestion estimated by a global router is conventionally used for routing estimation during placement, but it does not include real detailed routing violations, which adversely affect the routability of a design. Presently, there are no methods that directly aim to predict detailed routing violations. In this paper we propose a machine learning based method to predict the shorts that are a major component of detailed routing violations. The proposed method can be integrated into a placement tool and be used as a guide during the placement process to reduce the number of shorts happening in the detailed routing stage. Empirical results show that our method is successful in predicting 88% of the shorts with only 16% incorrectly predicting shorts in no short violation area.**

I. Introduction

With the growth of very large-scaled integration (VLSI) technology, more technology constraints are added, making the design automation and manufacturing more challenging. One of the challenges is the routability of a circuit. The increasing number and complexity of design rules at 22nm mean that routability rules need to be considered during the placement process otherwise unroutable designs can be produced. Therefore, detailed routability-driven placement has become a very challenging and important problem in design automation and has been the subject of the two of the recent ACM/IEEE International Symposium on Physical Design (ISPD) contests [1], [2].

In this paper, we propose a machine learning based algorithm that learns from previous data and predicts the real detailed routing (DR) pin shorts that are a major part of detailed routing violations. The main contributions of this paper are:

- Proposing a method for predicting DR violations during the placement process by learning from real DR information.
- Developing a DR violation prediction technique that can be used to improve the placement in order to directly aim to reduce shorts during placement.
- Eliminating the need to use a global router during the placement which results in saving runtime.
- Proposing and extracting appropriate features and using appropriate classification techniques for imbalanced data to model the correlation between placement data and DR pin shorts.

The rest of this paper is organized as follows. In Section II, a literature review on routing violation prediction and estimation and imbalanced data classification is given. In Section III, the proposed method is described in detail. Experimental results are reported in Section IV. Finally, conclusions and future work are presented in Section V.

II. Preliminary

A. Detailed-routing driven placement

There are many factors leading to detailed routing violations. Explicit modeling of all of these factors is a complex task. Increasing design rules in modern technology, such as pre-routed wires, cell edge spacing, and blockages in modern circuits add complexities to the placement that are hard to address using traditional placement techniques [3], [4]. These constraints imposed by design rules must be satisfied to successfully complete detailed routing; otherwise a design cannot be routed because of pin shorts and pin access issues. Pin shorts occur where cell pins and pre-routed wires are placed on the same metal layer or where two neighboring cells are too close to each other. Pin access issues occur where a cell pin is placed under a pre-routed wire or a highly-congested area, and the pin cannot be accessed without imposing other DR violations [5], [4]. In order to mitigate routing violations, a global routing estimation is conventionally used as a guide during the placement process. However, the correlation between the global routing estimation and DR violations is not easy to perceive, and there is a growing gap between the congestion estimated by a global router and real detailed routing violations [6], [3], [7], [5]. With the lack of appropriate local congestion modelling and increasing DRC rules, optimizing the conventional metrics is not sufficient for targeting detailed routability-driven placement.

B. Imbalanced data classification

Learning from skewed data is challenging and has been the subject of the work of many researchers in recent years. Most of the classifiers classify the majority of the instances to the overpresented class [8].

Data sampling and boosting are some of the approaches proposed to improve the classification of imbalanced datasets.

Data sampling, which includes oversampling and undersampling, balances the distribution of the classes in the training dataset. Oversampling adds data to the minority class and undersampling extracts data from the majority class.

978-1-5090-3969-2/17/$31.00 ©2017 IEEE

Boosting is an ensemble method that combines a number of weak classifiers to generate an effective classifier. Weak classifiers are trained using different instance distributions in each iteration. These classifiers are then weighted based on their performance on the current iteration. The final classifier is a weighted combination of the weak classifiers.

RUSBoost is the state-of-the-art method for imbalanced data classification that combines data sampling and boosting [9], [10]. This method eliminates the data distribution imbalances between the classes by data sampling and improves the classification performance of the weak classifiers by boosting. The data sampling method used in RUSBoost is random undersampling (RUS) that removes instances of the majority class from the training dataset until the intended balanced class distribution is achieved.

III. Methodology

In order to construct a model that predicts the DR shorts during placement, we use (i) the placement information i.e. features, (ii) the actual detailed routing shorts, i.e. targets, and (iii) a machine learning technique to learn the correlation.

In Section III-A we explain the proposed modeling for our problem as a classification problem. In Section III-B the feature extraction step is explained and in Section III-C selecting an appropriate learning model is discussed.

A. Problem modeling

In the proposed approach, we model the problem of predicting routing violations during the placement as an imbalanced data binary classification problem.

In this model, the circuit area is divided into very small tiles and the occurrences of shorts are investigated inside these tiles. Each tile is described by a feature vector and considered as an instance. Instances in the training set are labeled based on the detailed routing outcome of the circuit.

The size of the tile is selected to be small enough such that there are not too many violations inside one tile. However, there might still be tiles that have more than one violation in their area. Since the purpose of predicting the violation in a tile during the placement is to inform the detailed placer that more effective moves are needed in the area, binary classification is proposed to be sufficient without the complexity of predicting the exact number of violations occurring.

The overview of the model is presented in Figure 1. Solid arrows indicate the training flow. Initially, the circuits chosen for training are placed using a placer. The solutions must be legal and must satisfy all design rules such as fence regions [5]. Then, the solutions are fed to two different steps: (i) feature extraction step; and (ii) routing step.

In the feature extraction step, the area of each circuit under training is divided into a grid of rectangular areas called tiles T. For each tile $t \in T$, several features are extracted (see Section III-B).

In the routing step, each circuit under training is routed using a detailed routing tool, and the locations of any identified shorts are collected. Each short issue is assigned to the same tiles T built in the feature extraction step. An instance belongs to Negative (N) class and is labeled with the target value of 0 if there is no short in its tile area; and belongs to Positive (p) class and is labeled with the target value of 1 if any short violation is observed in its tile area after detailed routing.

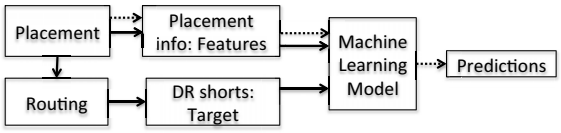

Fig. 1. Overview of the framework

Finally, the collected data is given to a learning algorithm.

The constructed model is used to predict violations on the validation data which are the previously unseen tiles. Dotted arrows show the flow of prediction on new data. Once the prediction on validation data is successful, the generated model can be integrated into the placement tool and can be used as a DR violation estimation to guide the placement.

The main stages of the modeling i.e. feature extraction and learning model selection are described in the following sections.

B. Feature extraction

One of the most important steps of constructing this model is extracting appropriate features that contribute to routing violations.

In the proposed modeling, circuit layout is divided into several non-overlapping tiles (similar to G-Cells in Global routing) and features of each tile are extracted. Each tile is referred to as an instance.

- Cell and pin densities directly influence the routability of a design [11]. Hence, for each tile t, both cell and pin densities are computed and included in the features of t.
- Blockages are grouped into two categories: (i) placement blockages, and (ii) routing blockages. The former includes fixed cells and blocked areas defined in the given placement. In addition, similar to [4], [5], all pre-routed wires on layer M1 and M2 are considered as placement blockages. Although cells are not placed on sites overlapping with placement blockages; they are important as they may cause issues such as narrow channels [11] and may degrade the routability of neighboring tiles. In addition, routing blockages on layers above M2 play a key role in causing shorts. The area of each tile t overlapping with placement and routing blockages is extracted and considered as one of the features of tile t.
- The connectivity of a tile is also considered as a feature. The connectivity is defined as the sum of the degree of the nets connecting to each pin in the tile.
- Relative position of tiles on the circuit area is another feature that is used in the training of the circuit.
- According to our experiments, a tile t may be affected by the characteristics of its neighborhood area. The neighborhood area of tile t is defined as an area overlapping

with tile t and its adjacent tiles. For this reason, the features (mentioned above) of the neighborhood area of tile t are also considered as another set of features to be taken into account for tile t.

In our proposed method, the feature vector $v(t)$ of each tile t is built upon the features mentioned above. $v(t)$ is divided into two subsets: the first subset represents the individual characteristics of tile t while the second subset define the characteristics of neighborhood area of tile t. For each subset, cell and pin densities with values between zero and one are computed. In addition, for placement and routing blockages, the areas of blockages overlapping with tile t (the neighborhood area) are divided by the area of tile t (the neighborhood area) to be added into the first (second) subset of $v(t)$. The relative positions of tiles (neighborhood areas) are normalized between zero and one to ensure that circuits with different sizes can be used in our proposed model. Finally, the net conductivities of tile t and its the neighborhood area are added into both subsets.

C. Model selection

There are many supervised learning algorithm that can be used for classification. However, generally, when a placed circuit can be routed by a detailed router, there are many more instances without violation than instances with violation. Therefore, we need to carefully select a learning algorithm that performs well on an imbalanced dataset. We use ensemble RUSBoost algorithm for our prediction modeling [9], [10].

IV. Results

A. Setup

The dataset is generated from the ISPD 2015 benchmarks [3]. The benchmark circuits are placed by EhPlacer [5] (achieved 2nd place in the ISPD 2015 contest [2]) and routed by Mentor Graphics Olympus router [12].

In order to evaluate our proposed method, nine circuits from the ISPD 2015 benchmark suite are selected as the training set while the other five designs (marked with * in the first column of the Table II) are only included in the test set. There are two other circuits, called mgc_edit_dist_a and mgc_superblue16a, which are not included in the training and test sets as the detailed routing step was not performed for them due to high global routing congestion. In selecting the training set, 25% holdout validation is applied on the circuits of the training set, i.e. 75% of the instances randomly selected from these data are used for training while the remaining 25% are used for validation. In the presented experiments, the weak learner used with RUSBoost method is a decision tree with the leaf number of 30 and learning rate of 0.1.

B. Validation and discussion

Since the nature of our data is imbalanced and less than 1% of the data belong to the positive class, evaluating our model using the traditional metrics for classification such as accuracy is insufficient and can be misleading. In order to evaluate our model's performance, the metrics that are used for case of class imbalance are assessed.

TABLE I
Confusion Matrix

Actual DR	Prediction Outcome: Normal	Prediction Outcome: Shorts
Normal	408987 (89%)	51802 (11%)
Shorts	352 (11%)	2865 (89%)

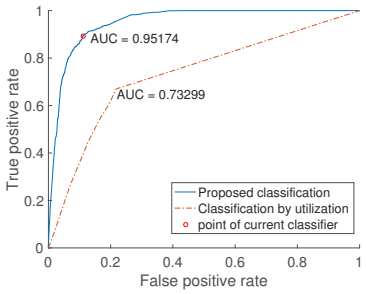

Fig. 2. ROC curve

Our proposed model is trained using the training data described in section IV-A. In Table I, the confusion matrix of the validation set is presented. A confusion matrix is a table that presents the performance of a classifier and is widely used to assess imbalanced data. The rows of the matrix shown in Table I present actual normal tiles and tiles with violations while each column shows the number of prediction of each class. The confusion matrix presents the following four cases:

True Positive/Negative (TP/TN): The number of instances that are correctly classified as positive/negative.

False Positive/Negative (FP/FN): The number of instances that are negative/positive and incorrectly classified as positive/negative.

It can be seen from the matrix that the model is able to detect 89% of the violations on the previously unseen instances with only 11% rate of false alarm (false violation detection). The rate of detection can even increase by changing the threshold of the model in the cost of higher false violation detection rate. This is illustrated using the Receiver Operating Characteristic (ROC) curve given in Figure 2. ROC curves show the trade-off between detection rate and false alarm rate. The red circle on the curve shows the classifier that is selected for the experiments in this paper. In order to assess the performance of the proposed model, we have compared the generated ROC curve with the ROC curve generated by classification based on utilization as a baseline. In generating this curve, different threshold values between 0 and 1 are considered and the tiles with utilization rates higher and lower than the threshold are assigned to positive and negative class, respectively, and the true positive rate and false positive rate are calculated. It can be seen that for similar true positive rate in two curves, the false positive rate of the proposed model is much lower.

The area under the ROC curve (AUC) is often used as a

TABLE II
PREDICTIONS

Circuit	Instances	P	TP	TPR	SPC	ACC
des_perf_1	49284	2252	2169	96 %	53 %	53 %
des_perf_a	202500	5985	5727	96 %	86 %	86 %
des_perf_b	90000	0	0	NA	71 %	71 %
fft_1*	17424	90	70	78 %	78 %	78 %
fft_2	29240	20	18	90 %	47 %	47 %
fft_a	160000	778	598	77 %	84 %	84 %
fft_b	160000	1543	1262	82 %	83 %	83 %
matrix_mult_1*	75624	129	105	81 %	50 %	50 %
matrix_mult_a	562500	70	21	30 %	96 %	96 %
matrix_mult_b	562500	1472	1152	78 %	95 %	95 %
pci_bridge32_a	40000	747	478	64 %	73 %	73 %
pci_bridge32_b*	160000	9	9	100 %	85 %	85 %
superblue11_a*	1767768	32	22	69 %	81 %	81 %
superblue12*	732720	223	122	55 %	65 %	65 %
Average	329254	954	840	77 %	75 %	75 %
Total	4609560	13350	11753	88 %	84 %	84 %

single numeric measure to evaluate the performance of the model constructed for imbalanced datasets. The AUC of the proposed model is 0.95, which is very close to the ideal AUC value of 1 and is by far larger than the AUC of the baseline.

The total runtime spent for training in this experiment is 98 seconds, which is a one-time task, and once the model is trained the only runtime added to the placer is the prediction time, which takes fraction of seconds. This can save thousands of seconds of estimation using a global router estimator or going back and forth between placement and routing. The model can be updated by adding new training data to the system and retraining the system to achieve even better prediction.

The performance of the proposed model on individual circuits is presented in Table II. In columns 1-3, the names of the designs, the total number of tiles in the designs (Instances), the total number of tiles containing shorts (P), and the number of tiles correctly predicted as tiles with shorts (TP) are presented, respectively. The remaining columns indicate the following metrics:

Sensitivity or True Positive Rate (TPR): Sensitivity or True Positive Rate shows the ability of the model to classify positive class. In our problem, TPR presents the percentage of the tiles with short violation who are correctly identified as having violation. Sensitivity is defined by:

$TPR = TP/P = TP/(TP + FN)$

Specificity (SPC): Specificity measures the ability of the model to classify the negative class. In our problem, SPC presents the percentage of normal tiles that are correctly identified as not having violation. Specificity is defined by:

$SPC = TN/(TN + FP)$

Accuracy (ACC): Accuracy shows the overall performance of the classifier and is defined by: $ACC = (TP + TN)/All$

In Table II, the circuits that are marked with * are completely unseen circuits. None of the tiles in these circuits participated in the training of the model. The training instances are selected from the rest of the circuits. Note that only a portion of the data of these circuits is used for training, and the rest of data is previously unseen instances. According to the table, our model successfully predicted 88% of the tiles with violations in total, with the average of 77% on each design, while the false alarm is only 16% in total with the average of 25% on each design. The ACC value of the total is 84%. The ACC values are very close to SPC values as expected by considering the imbalance in the dataset. However, our model could only detect 30% of the violations of matrix_mult_a. We realized that the violations in this design are scattered over the design and are caused by global features, where our model is based on local features. This issue will be addressed in our future work.

V. Conclusion and future work

In this paper, we propose a machine learning based method that predicts the shorts that are a major component of detailed routing violations. Empirical results show that our method is successful in predicting the 89% of shorts in unseen parts of the training circuits with only 11% false alarm, and successfully predicted 88% of the tiles with violations in total, while the false alarm is only 16%. Future work includes using the predictions as a guide during the placement process and proposing moves that can resolve the predicted violations to reduce the number of shorts happening in detailed routing stage.

References

[1] ISPD 2014 detailed routing-driven placement contest. http://www.ispd.cc/contests/14/ispd2014_contest.html. Accessed: 2016-10-03.

[2] ISPD 2015 blockage-aware detailed routing-driven placement contest. http://www.ispd.cc/contests/15/ispd2015_contest.html. Accessed: 2016-10-03.

[3] Ismail S. Bustany, David Chinnery, Joseph R. Shinnerl, and Vladimir Yutsi. ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement. In *Proc. of ISPD*, pages 157–164, 2015.

[4] A. Kennings, N. Karimpour Darav, and L. Behjat. Detailed placement accounting for technology constraints. In *Proc. of VLSI-SoC*, pages 1–6, 2014.

[5] N. Karimpour Darav, A. Kennings, A. Fakheri Tabrizi, D. Westwick, and L. Behjat. Eh?placer: A high-performance modern technology-driven placer. *ACM TODAES*, 21(3):37:1–37:27, April 2016.

[6] V. Yutsis, I. S. Bustany, D. Chinnery, J. R. Shinnerl, and W.-H. Liu. ISPD 2014 benchmarks with sub-45nm technology rules for detailed-routing-driven placement. In *Proc. ISPD*, pages 161–168, 2014.

[7] A. F. Tabrizi, N. K. Darav, L. Rakai, A. Kennings, W. Swartz, and L. Behjat. A detailed routing-aware detailed placement technique. In *2015 IEEE Computer Society Annual Symposium on VLSI*, pages 38–43, July 2015. ISSN 2159-3469.

[8] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 21(9):1263–1284, September 2009. ISSN 1041-4347.

[9] Chris Seiffert, Taghi M. Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. Rusboost: Improving classification performance when training data is skewed. In *ICPR*, 2008.

[10] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano. Rusboost: A hybrid approach to alleviating class imbalance. *Trans. Sys. Man Cyber. Part A*, 40(1):185–197, January 2010. ISSN 1083-4427.

[11] J. Cong, Guojie Luo, K. Tsota, and Bingjun Xiao. Optimizing routability in large-scale mixed-size placement. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pages 441–446, Jan 2013.

[12] Mentor Graphics, Inc. Olympus-SoC place and route for advanced node designs. Technical report, www.mentor.com/products/ic_nanometer_design/place-route/olympus-soc, 2015.