Eh?Predictor: A Deep Learning Framework to Identify Detailed Routing Short Violations From a Placed Netlist

Aysa Fakheri Tabrizi, Nima Karimpour Darav, Logan Rakai, *Member, IEEE*, Ismail Bustany, Andrew Kennings, and Laleh Behjat

***Abstract*—Detailed routing is one of the most challenging aspects of the physical design process. Many of the violations that occur during the detailed routing stage stem from the placement of the cells. In this paper, we propose a deep learning framework to identify short violations that can occur during detailed routing from a placed netlist. One of the advantages of our technique is that by using the proposed deep learning-based predictor, global routing is no longer required as frequently and hence the total runtime for place and route can be significantly reduced. In this paper, we discuss the proposed framework and the methodology for analyzing the extracted features. The experimental results show that the average sensitivity, specificity, and accuracy of Eh?Predictor is above 90%. In addition, we show that Eh?Predictor is up to 14 times faster than NCTUgr for smaller designs and up to 96 times faster for larger designs.**

***Index Terms*—Data mining, design automation, imbalanced data, machine learning, physical design, placement, routing.**

I. Introduction

ADVANCES in very large scale integrated circuits (VLSI), adds to the challenges in electronic design automation (EDA). Routability is one of the most challenging issues in modern designs. New technology node scales, technology constraints and increasing design rules add to the complexity of physical design. Remedial procedures after detailed routing, such as rip up and reroute process are not always adequate to resolve occurrences of violations, such as shorts, pin access problems, and other detailed routing violations. In these cases, the layout is considered unroutable and the placement and routing stages have to be repeated. This process can take up to several days. Therefore, routability-driven placement algorithms are inspired to avoid unroutable layouts.

Manuscript received June 14, 2018; revised October 24, 2018 and February 1, 2019; accepted March 17, 2019. Date of publication May 16, 2019; date of current version May 22, 2020. This work was supported in part by the Natural Sciences and Engineering Council of Canada, in part by Micro-Electronics Corporation, in part by Compute Canada, and in part by Mentor Graphics. This paper was recommended by Associate Editor Bustany_Chu. *(Corresponding author: Aysa Fakheri Tabrizi.)*

A. F. Tabrizi, L. Rakai, and L. Behjat are with the Department of Electrical and Computer Engineering, Schulich School of Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: afakheri@ucalgary.ca).

N. K. Darav is with Microsemi Corporation, Kitchener, ON N2H 6R2, Canada.

I. Bustany is with Xilinx Inc., San Jose, CA 95124 USA.

A. Kennings is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada.

Digital Object Identifier 10.1109/TCAD.2019.2917130

A. Motivation

Developing routability-driven placement algorithms requires a realistic estimation of routability and prediction of problematic areas. Conventionally, global routers are used to estimate and evaluate the routability of a placement solution during the placement process. However, global routers ignore the effects of local nets. Local nets contribute a high percentage of the total nets and can affect the quality of the final routing solution [1]–[3]. There are a few methods that take into account local nets during global routing such as [4] that incorporate local nets in a global-routing-based congestion analyzer using Steiner tree wire length estimations and pin density. Another example is presented in [5], where local net density is modeled as a regression problem and machine learning is used to solve it. Invoking a global router several times can be time consuming. At the same time, using global routing solutions along with local net modeling only generates a better congestion map and does not directly predict the routing violations. The motivation of this paper is to develop a fast predictor for identifying detailed routing pin short violations without using global routing.

B. Contributions

In this paper, we propose Eh?Predictor, a deep learning framework to detect the short violations from a placed netlist without using a global router. The proposed framework learns from actual detailed routing violations and predicts the short violation prone areas at the placement stage.

In designing this framework, the factors that can contribute to the routing violations are determined and multiple relevant quantitative features are defined and extracted from the placed netlist. A deep learning model is trained using a training set that includes feature values and labels, determined by actually routing the design. Then, the trained model is used for predicting the presence of shorts using the feature values at the placement stage.

The experimental results show that the proposed detection method is able to predict on average 90% of the shorts on previously unseen data with only 5% false alarms (FAs)

and considerably reduces the computational time. Our main contributions include the following.

1) Proposing a machine learning framework to predict the shorts before routing.
2) Significantly decreasing the prediction time by not using a global router.
3) Modeling the short prediction problem as a binary classification problem with imbalanced data.
4) Defining and extracting effective features.
5) Analyzing the extracted features and their correlation with short violations.
6) Customizing a deep learning model that fits the problem and its imbalanced nature.

Eh?Predictor is evaluated by empirical experiments on advanced designs, and has shown improvement on predicting short violations.

The remainder of this paper is organized as follows. In Section II, backgrounds related to placement, routing, and machine learning are reviewed. The proposed modeling and methodology are presented in Section III. The experimental results are discussed in Section IV. Finally, Section V concludes this paper.

II. Preliminary

A. Placement and Routing

1) Placement: It is a key step in physical design synthesis [6]. The process of placement tries to find an optimal solution to determine the location of each module (cell) inside a die surface such that a variety of objectives are optimized while several constraints are satisfied. Some of these objectives include wire length, performance, power consumption, circuit area, reliability, and routability. However, wire length is considered as the main objective of placement because placers leading to less wire length can better cope with emerging challenges in modern technology processes [7]. The placement problem is NP hard [8], [9] even if only one objective such as wire length is optimized.

2) Routing: Routability has become critical in modern VLSI placement [10], [11]. Wire length-driven placement methods, which are not aware of actual routes, may result in unroutable solutions. Therefore, a placer should not only minimize wire length; but also must consider the routability of the final solution during the placement process. Due to its complexity, the routing step is performed into two main steps: 1) global routing and 2) detailed routing [12], [13].

Global routing in its core utilizes a grid of tiles (called g-cells) with the same size. In global routing, all connections inside a g-cell (intraconnections or local nets) and some metal layer constraints are ignored while all connections among g-cells (interconnections or global nets) are routed. Each g-cell, g_i, is composed of four routing edges. Two vertical edges determine the available horizontal routing capacities of the g-cell while two horizontal routing edges provide the available vertical routing capacities of the g-cell [14]. From the placement viewpoint, global routing can also be utilized to estimate routing overflows and to improve routability during the placement process [15], [16].

In contrast with the global routing step, the detailed routing step must consider local nets and technology constraints during routing. Due to complicated design rules in modern technology processes, the detailed routing is very time consuming. For example, the detailed routing process for the benchmark *mgc_superblue11_a* released in the ISPD 2015 contest [17] takes more than three days using a commercial tool while the runtime of its placement process using Eh?Placer [15] is less than 40 min.

3) Technology Constraints: The technology constraints are as follows. *Fence region* constraints are defined to address issues, such as isolating separate voltage regions and preserving space for incorporating global repeaters [17], [18]. Cells assigned to a fence region have to be placed inside the boundary of the region while other cells have to be placed outside of the boundary. Violating fence region constraints makes the placement unacceptable. These constraints are generally considered as *hard constraints*. A design may have several joint or disjoint fence regions. The *target density* constraint controls how uniform cells should be distributed over the chip area [15]. *Pin shorts* occur where prerouted wires including power/ground (PG) [19] mesh or primary inputs/outputs (PIOs) and cell pins are on the same metal layer. PIOs should be taken into account because they are considered as fixed objects [17], [20]. Nonedefault routing (NDR) rules are defined for some nets to dedicate different wire width and spacing from what are defined for regular nets.

B. Machine Learning

Machine learning is a filed of artificial intelligence (AI) and a set of techniques that enables computers to learn and act without explicitly being programmed [21].

1) Supervised Learning: Supervised machine learning refers to the techniques used for developing functions from labeled training examples that can be used to label new data [21].

In a supervised machine learning technique, first a training set is developed and its labels are determined. The next step is to determine the features that would represent the learned functions. These features are essentially the most important part of a machine learning process. Once the features are determined, the type of the learned function is determined. Some of the most well-known examples of the models are logistic regression, decision trees, support vector machines, and neural networks. A learning algorithm is then performed on the training set and the learned functions are obtained. Finally, the performance of the learned functions are evaluated using a separate test set.

2) Deep Learning: The idea of neural networks existed for decades. However, it has caught more attention in recent years, due to more data availability and computational scale. Deep learning is mostly referred to large neural networks with multiple hidden layers [22]. By using more data for training, the performance of old learning algorithms plateaus, where, the performance of very large neural networks or deep learning can improve and make deep learning a popular choice.

3) Imbalanced Data Evaluation Metrics and Terms: Imbalanced data classification problem is a supervised learning problem, where the proportion of the number of data in classes is skewed. In a binary imbalanced data classification problem, the majority of the data belongs to one of the

classes [23], [24]. In the following a brief summary of the metrics that are commonly used for imbalanced data evaluation are given.

Confusion Matrix: A confusion matrix is a table that presents the performance of a classifier and is widely used to assess imbalanced data. The rows of a confusion matrix present actual classes while columns show the predictions. The confusion matrix presents the following four cases.

1) *True Positive (TP/TN):* The number of instances that are correctly classified as positive/negative.
2) *False Positive (FP/FN):* The number of instances that are negative/positive and incorrectly classified as positive/negative.

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

$$\text{TPR} = \frac{\text{TP}}{P} = \frac{\text{TP}}{\text{TP} + \text{FN}} \times 100.$$

Specificity (SPC): 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

$$\text{SPC} = \frac{\text{TN}}{\text{TN} + \text{FP}} \times 100.$$

FA: Describes cases where the developed function mistakenly identifies an event. In our problem, FA describes the tiles with no shorts that are incorrectly identified as having shorts. It is defined as

$$\text{FA} = \frac{\text{FP}}{\text{TN} + \text{FP}} \times 100 = 100 - \text{SPC}.$$

Accuracy (ACC): It shows the overall performance of the classifier and is defined by

$$\text{ACC} = \frac{\text{TP} + \text{TN}}{\text{All}} \times 100.$$

All is the number of instances.

Matthews Correlation Coefficient (MCC): Is a metric that is used for binary classification of imbalanced data. MCC returns a value between -1 and $+1$. An MCC of $+1$ represents perfect prediction. MCC is calculated as

$$\text{MCC} = \frac{\text{TP} \times \text{TN} - \text{FP} \times \text{FN}}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}}.$$

4) Machine Learning in EDA: Data mining and machine learning techniques have been applied to EDA in recent years [25]. In [5] and [26]–[29] machine learning is used for lithography, hot-spot detection, timing estimation, modeling the local nets and routability prediction using global router.

III. Proposed Methodology

Eh?Predictor is developed to be used as a guide during the placement by predicting the short violations without using a conventional routing estimator.

We have modeled the problem of predicting the short violations of a design from a placed netlist as a binary classification problem with imbalanced data and used the supervised machine learning approach to perform the classification. The framework proposed to predict the shorts along with the details of its designing are presented in this section.

A. Problem Formulation

The goal of this paper, is to detect the areas in the circuit that are prone to the occurrence of short violations, before actually routing the circuit. For this purpose, we divide the circuit layout into small grids, called tiles. Then, the possibility of short occurrence in each tile is investigated. We modeled this problem as a binary classification problem with the following inputs and outputs.

Input: $(\vec{x}^{(i)}, y^{(i)})$, where $x_j^{(i)} \in \mathbb{R}$ and $y^{(i)} \in \{0, 1\}$.

Output: $\hat{y}^{(i)} \in \{0, 1\}$, where, i is the index of the tile, $\vec{x}^{(i)}$ is the feature vector of tile $t^{(i)}$ from the after placement layout of a design and $x_j^{(i)}$ is the jth feature in $\vec{x}^{(i)}$. $y^{(i)}$ is the binary label of tile $t^{(i)}$. Here, $y^{(i)} = 0$ indicates the absence of short violations and $y^{(i)} = 1$ indicates their presence. $\hat{y}^{(i)}$ is the predicted binary label of tile $t^{(i)}$, where $\hat{y}^{(i)} = 1$ indicates the prediction of short violations. A tile $t^{(i)}$ is referred to as an *instance* in general. A tile with short violations ($y^{(i)} = 1$) is a *positive instance*, and a tile without short violations ($y^{(i)} = 0$) is a *negative instance*.

Since the number of tiles with short violations using a fair placer is much lower than the number of normal tiles, the majority of instances belong to negative class. This problem is a binary classification problem with imbalanced data.

B. Framework Flow

The flow of the proposed framework for violation prediction and its integration in physical design flow is presented in Fig. 1. In this figure, the blocks represent the processes and the clouds represent the inputs and the outputs. The black dotted box and the red solid box indicate the training phase and the prediction phase, respectively. The dotted arrows are only performed during the training phase.

The input to our predictor is a *Placed Netlist*, generated by a *Placer* from a *Netlist* of a design. During the *Generating tiles* process, the layout of the placed design is divided into nonoverlapping rectangular areas, called tiles $\mathbb{T}$ and the tiles that completely overlap the macros are discarded. The features of each tile $t^{(i)} \in \mathbb{T}$ are extracted during the *Feature Extraction* process. These features are fed into the trained *Learning Model*. The binary output $\hat{y}^{(i)}$ shows the prediction of the presence or absence of short violations in tile $t(i)$.

When training the model, the *Placed Netlist* of the training designs is also given to a *Router* to be routed, and the *violations*, along with their locations, are extracted. Next step is *Labeling* the tiles, using the location of the violations. The tiles that have short violations reported by detailed router are labeled as positive instances. All other tiles are labeled as negative instances. The tile features ($\vec{x}^{(i)}$) and labels (y^i) are fed to

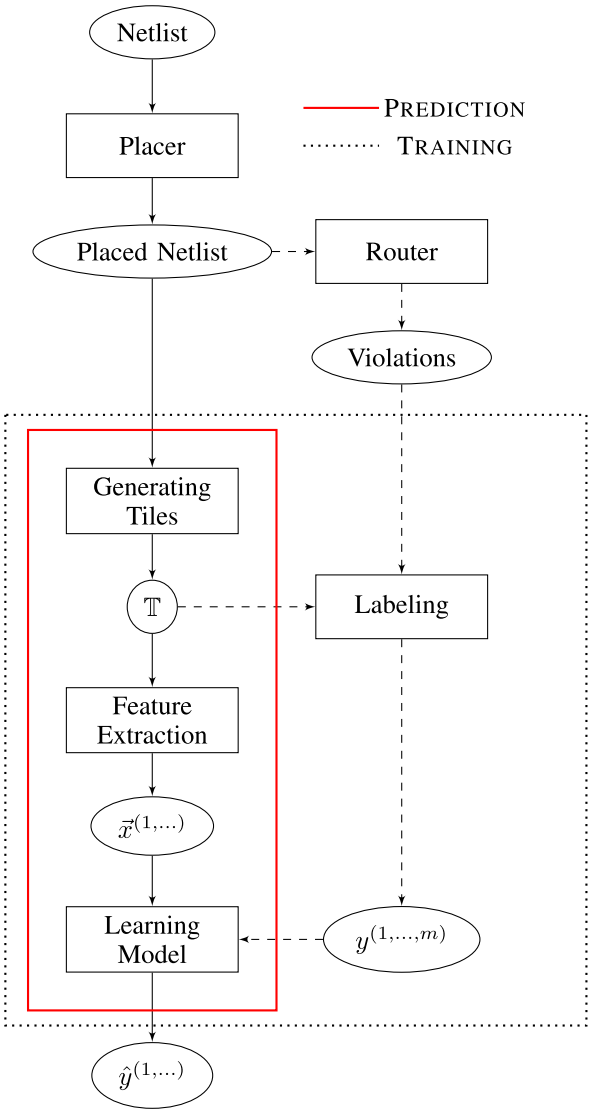

Fig. 1. Outline of Eh?Predictor.

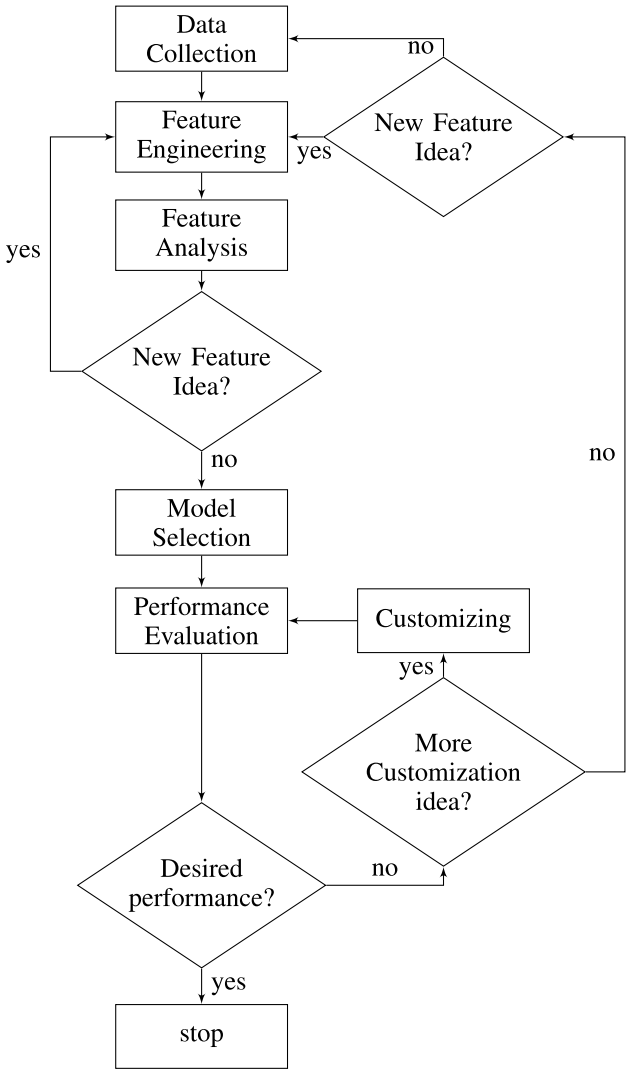

Fig. 2. Design flow: the process of designing Eh?Predictor.

the *Learning Model* for training. Once the model is trained, it can be used as a short violation predictor on new, previously unseen tiles.

C. Framework Design

The process of designing the proposed framework is shown in Fig. 2.

The first step is collecting data and developing a data set that can be used for training. The next step is designing the initial features. The initial features are designed based on factors that can influence routing of a design. These factors are determined by thorough research and experience in developing detailed routing aware placers. Then, the initial feature values for the data set are calculated and analyzed. Based on these analyses, new features are designed. In the next step an appropriate learning model with initial settings is selected. The model is trained by the data set with designed features and its performance is evaluated. The model is customized through many iterations until a satisfactory performance is achieved. If satisfactory performance is not achieved and no more customization idea can improve the performance, more features can be hand-engineered, extracted and added to feature set. New features can be defined by analyzing the instances with false prediction. Collecting more data and adding to the data set for training is the next step to further improve the performance of the predictor. Sufficient data, effective feature set, and appropriate learning model are the key factors in developing a successful predictive method. The details of obtaining each of these factors are described in the following.

D. Data Collection

To generate the data set, a set of designs are selected and placed by a placer. For each placed circuit, tiles are generated. The data set consists of information about the netlist of a design before placement, floorplanning, tiling, and placement of the design. For labeling the training tiles, the placed netlists are also routed using a detailed router. The exact location of short violations are extracted and mapped to the tiles.

The tiles with one or more short violations are labeled as positive and the rest of the tiles are labeled as negative. We have used academic placer, Eh?Placer for placement and Mentor Graphic's Olympus router for labeling.

In the designs that macros are routing blockages, the tiles that completely overlap the macros are discarded. The reason is that there is no routing on these tiles and hence there is no short violations and there is no need for prediction. Our predictor can easily predict these tiles correctly. Including macros results in high specificity value for designs with macros and could be misleading.

E. Feature Extraction

Selection and extraction of effective features can significantly impact the performance of the predictor. In order to generate the feature vector, first the factors that contribute to violations are determined through experimentation and the features are extracted as shown in Fig. 3.

In this process, several quantitative features are derived from each individual factor or a combination of multiple factors and are calculated for each tile based on the Netlist, floorplan, placement, and tiling information. These features form the initial feature vector. The features are studied and analyzed and more features are introduced based on these analyses and added to feature vector. Then feature analyses are repeated for the new features. Since the range of the values in feature vector varies, the values of features are standardized to contribute equally. The generated feature vector is fed to the deep learning model. The layers of deep learning model generates more complicated features by combining them. Performance of predictor on different tiles are studied and examined. New features are introduced based on these studies and are added to feature vector and the process is repeated.

1) Effective Factors: The factors that can contribute to violations are determined as follows.

Global Characteristics: The general characteristics of the design, such as density of the designs, macros, regions. These features can be informative in making decisions, as similar designs may have similar patterns. These factors are similar for the tiles of each design. For example, a design with higher density is more prone to short violations.

Routing Accessibility: The area covered by macros, and routing blockages, such as power grids, in different metal layers affect the accessibility of routing and potential violations.

Narrow Channels Effects: When macros are placed close to each other or close to the border of the design, the narrow channels are formed. As the macros act as routing blockages, the nets are forced to go around the macros which in turns results in high density of routs in narrow channels that can result in detailed routing violations such as shorts. The tiles over the narrow channels are good candidates for detailed routing violations such as shorts.

Utilization and Distribution: The cell density, pin density, and pin distribution of a design affects the performance of both the placer and the router.

Nets and Routes: The nets that are being routed are the main factor of occurrence of shorts. As the goal is detecting the shorts before actually routing the design, different features are considered to estimate the routing and track availability.

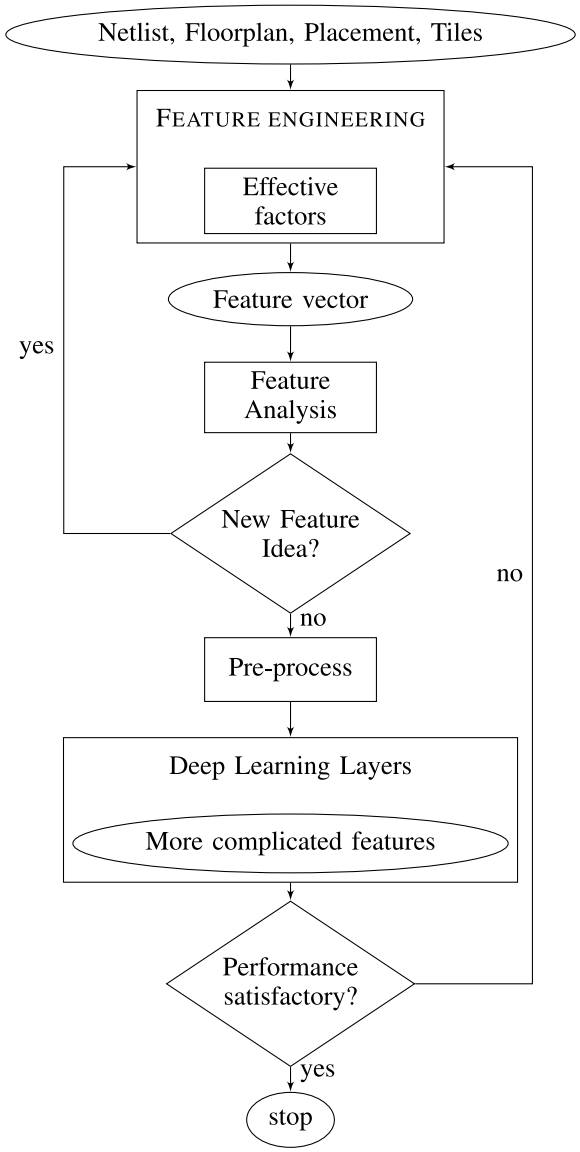

Fig. 3. Feature extraction process.

Spacial Pins and Nets: It has been observed that presence of some special pins and nets, such as IO pins, clock networks, and NDR nets could affect the routability.

Relative Location of the Tile in the Design: Tiles located in the center of a design are prone to become congested as more global nets cross over the area. Also the design borders are more prone to short occurrence due to input/output pins.

Neighborhood: The neighborhood of the bin that the occurrence of short is investigated can have a significant role in the occurrence of the short. For example, routing results and occurrence of possible violations can be different for exact same tiles with neighbors that have different pin density.

The following are the potential factors from each category that impact detailed routing and are considered in the feature extraction phase of this paper.

TABLE I
LIST OF FEATURES. T REPRESENTS TILE AND NH REPRESENTS NEIGHBORHOOD

Feature #	Feature name	Calculation
1	Region (Design) Density	$RD(DD) = Region\ (Design)\ Cell\ Area/Region\ (Design)\ Area$
2	Macro Density	$MD = Macro\ Area/Design\ Area$
3	Macro - T	$MT = [Macro \cap t]\ area/t\ area$
4	Macro - NH	$MNH = [Macro \cap nh_t]\ area/nh_t\ area$
5	Placement Blockage - T	$PBT = [placement\ blockage \cap t]\ area/t\ area$
6	Placement Blockage - NH	$PBNH = [placement\ blockage \cap nh_t]\ area/nh_t\ area$
7	Routing Blockage - T	$RBT = [Routing\ blockage \cap t]\ area/t\ area$
8	Routing Blockage - NH	$RBNH = [Routing\ blockage \cap nh_t]\ area/nh_t\ area$
9-15	Metal layers Blockages - T	$M_iBT = [M_i\ blockage \cap t]\ area/t\ area, , i = \{1...7\}$
16-22	Metal layers Blockages - NH	$M_iBNH = [M_i\ blockage \cap nh_t]\ area/nh_t\ area, , i = \{1...7\}$
23	Location on x-axis	$X = x_t/design\ width$
24	Location on y-axis	$Y = y_t/design\ height$
25	Cell Density - T	$CDT = cell\ area/t\ area$
26	Cell Density - NH	$CDNH = cell\ area/nh_t\ area$
27	Pin Density - T	$PDT = pin\ area/t\ area$
28	Pin density - NH	$PDNH = pin\ area/nh_t\ area$
29	M1 and M2 Pin Density - T	$M1M2PDT = M1M2\ pin\ area/t\ area$
30	M1 and M2 Pin Density - NH	$M1M2PDNH = M1M2\ pin\ area/nh_t\ area$
31	Number of PIO Pins	$IO = \#PIO\ pins$
32	Pin Distribution on x-axis	$PdistX = std(x_p \vert p \in \mathbb{P}_t)$
33	Pin Distribution y-axis	$PdistY = std(y_p \vert p \in \mathbb{P}_t)$
34	Local Net - T	$LNetT = \#\forall net \vert (\#p_{net} \in \mathbb{P}_t) > 1$
35	Local Net - NH	$LNetNH = \#\forall net \vert (\#p_{net} \in \mathbb{P}_{nh_t}) > 1$
36	Global Net - T	$GNetT = \#\forall net \vert (\#p_{net} \in \mathbb{P}_t) > 0\ \&\ (\#p_{net} \notin \mathbb{P}_t) > 0$
37	Global Net - NH	$GNetNH = \#\forall net \vert (\#p_{net} \in \mathbb{P}_{nh_t}) > 0\ \&\ (\#p_{net} \notin \mathbb{P}_{nh_t}) > 0$
38	Horizontally Passing Net	$PNetH = \#\forall net \vert (\exists pin_{net} \vert x_{pin_{net}} > x_t - \frac{w_t}{2}\ \&\ (\exists pin_{net} \vert x_{pin_{net}} < x_t + \frac{w_t}{2})$
39	Vertically Passing Net	$PNetV = \#\forall net \vert (\exists pin_{net} \vert y_{pin_{net}} > y_t - \frac{h_t}{2}\ \&\ (\exists pin_{net} \vert y_{pin_{net}} < y_t + \frac{h_t}{2})$
40	Maximum Tracks	$MaxTrackH/V = max\ tracks\ \#$ using linescan algorithm
41	Number of Clock pins - T	$ClkT = \#clkpins \in \mathbb{P}_t$
42	Number of Clock pins - NH	$ClkNH = \#clkpins \in \mathbb{P}_{nh_t}$
43	NDR nets	$NDRT = \#NDRpins \in \mathbb{P}_t$

2) Quantified Features: The list of features that are extracted based on the above factors, are presented in Table I and are explained in the following.

1) *Region Density (RD)—Design Density (DD):* This feature is derived based on the *global characteristics* and *utilization* factors and presents the cell density of the region or the design (in case the design does not have any region) to which the tile belongs.
2) *Macro Density (MD):* This feature is derived based on *narrow channel effect* and *global characteristics* factors and is defined as the area of macros in the design divided by area of the design.
3) *Macro—Tile/Neighborhood Overlap (MT/MNH):* These features capture *routing accessibility* and *narrow channel effect*. They calculate the tile and neighborhood area overlap with macros, respectively.
4) *Placement Blockage—Tile/Neighborhood Overlap (PBT/PBNH):* These features capture the *utilization* factor. They are calculated as the tile and neighborhood area overlap with any placement blockages, respectively.
5) *Routing Blockage—Tile/Neighborhood (RBT/RBNH):* These features capture the *routing accessibility* and *narrow channel effect* factors. They are calculated as the tile and neighborhood area overlap with any routing blockages, respectively.
6) *Metal Layers Blockages—Tile/Neighborhood (M_iBT/M_iBNH):* These features capture the *routing accessibility* and calculate the routing blockages in different layers over the tile and the neighborhood area, respectively.
7) *Location on x-Axis/y-Axis (X/Y):* These features show the *Relative location of the tile in the design*. To calculate these values, the *x*-axis and *y*-axis positions of the center of the tile are obtained and are normalized to the range [0 1] for each design.
8) *Cell Density—Tile/Neighborhood (CDT/CDNH):* Cell density captures the *utilization* factor and is the ratio of the total area of the cells within tile/neighborhood to tile/neighborhood area.
9) *Pin Density—Tile/Neighborhood (PDT/PDNH):* Pin density also captures the *utilization* factor and is calculated as the total pin area within the tile/neighborhood divided by tile/neighborhood area. As the number and shapes of pins on different cells vary, pin density and cell density values can be independent.
10) *M1 and M2 Pin Density—Tile/Neighborhood (M1M2PDT/M1M2PDNH):* This feature captures the *utilization* and *routing accessibility* factors and is calculated as the total pin area within the tile/neighborhood on M1 and M2 metal layers divided by tile/neighborhood area.
11) *Number of PIO Pins (IO):* This feature shows the number of the PIO pins within the tile and captures *routing accessibility* and *special pins and nets* factors.
12) *Pin Distribution on x-Axis/y-Axis (PdistX/PdistY):* This feature is computed as the standard deviation of the *x* and *y* locations of a pins in the tile and captures *utilization* factor.
13) *Local Net in Tile/Neighborhood (LNetT/LNetNH):* Local net is calculated as the total number of nets that have at least two pins in a tile/neighborhood and captures *connectivity factor*.
14) *Global Net in Tile/Neighborhood (GNetT/GNetNH):* Global net is calculated as the number of nets that have

at least one pin in a tile and at least one pin outside the tile. Global net captures connectivity factor.

15) *Horizontally/Vertically Passing Net (PNetH/PNetV):* This features also capture the *connectivity* and *effects of narrow channels* factors by estimating the number of the nets passing the tile horizontally/vertically. It is computed as follows: for each column/row of the constructed tile grid, the number of nets having at least one pin on both sides of the column/row is calculated and divided by the number of the tiles in that column/row.
16) *Maximum Horizontal/Vertical Tracks (maxTrackH/maxTrackV):* These features are estimations of maximum number of horizontal/vertical tracks using the line scan algorithm [30].
17) *Number of Clock Pins in Tile/Neighbothood (ClkT/ClkNH):* This feature is calculated as the number of clock pins in tile and captures the spacial pins factor.
18) *NDR Nets (NDRT):* This feature represents the effects of NDR nets, and is equal to the number of pins with NDR nets. NDR nets are the nets that need double-wide width and spacing when routing.

3) Correlation Analysis: Some of the defined features have direct impact on the presence of these violations, where others do not have a direct correlation with violations but the occurrence of multiple features for the same instance can affect the routability. The correlation between some of the features and violations are analyzed in this section.

The density of the region that the tile is located within affects the short violation occurrence. However, there are many more factors involved. Fig. 4 shows the short rate in the regions with different densities. The top graph shows the higher density regions, the middle graph shows the regions from the designs with macros, and the bottom graph shows the regions with lower density and no blockages. It can be seen that the highest rates belong to either high density regions or the regions from the designs with macros with average density. The regions with lower density from designs with or without macros, and the regions with average density from the designs without macros have the lowest short rate.

There are some regions from the designs with blockages with average density that have lower short rate. By looking at different features we realized that these designs have a bigger circuit area. This is shown in Fig. 5. In this figure, the x- and y-axis show the density of the region and the short rate, respectively. The size of the circles represent the area size of the design and the color shows the type of the design. The designs with bigger circuit area with similar region density and blockage coverage are the ones discussed before and have lower short rate.

Once all the global features that were common for tiles within the same region were studied, the individual features of individual tiles are investigated. In Fig. 6, the tiles are divided based on their coverage by macros. The y-axis shows the short rate. The highest short rate belongs to the tiles that are 30%–40% covered by the macros. Tiles with higher pin density that are partly covered by macros have the higher short rate. Although the tiles with NDR form about 30% of our data

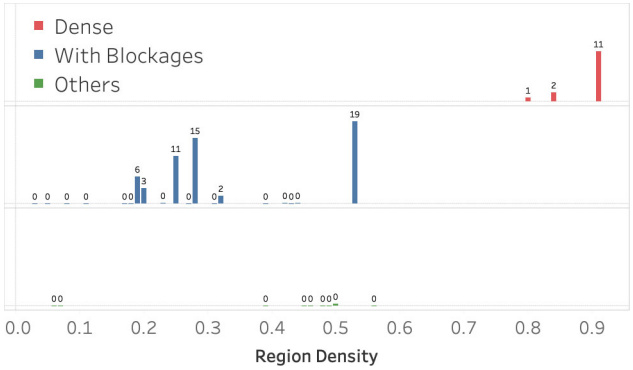

Fig. 4. Shorts per hundred tiles versus Region density.

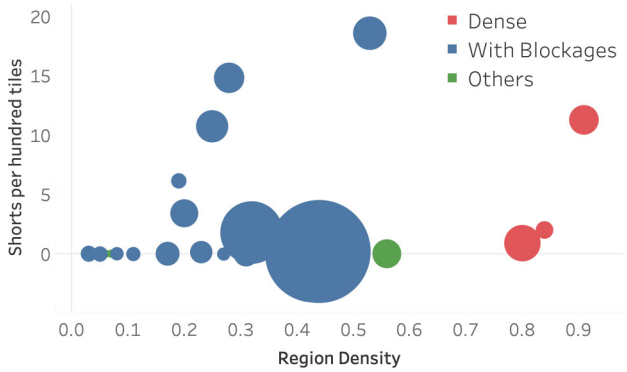

Fig. 5. Plot of shorts per hundred tiles for Region density. Color shows details about the type of design. Size shows the area size of design.

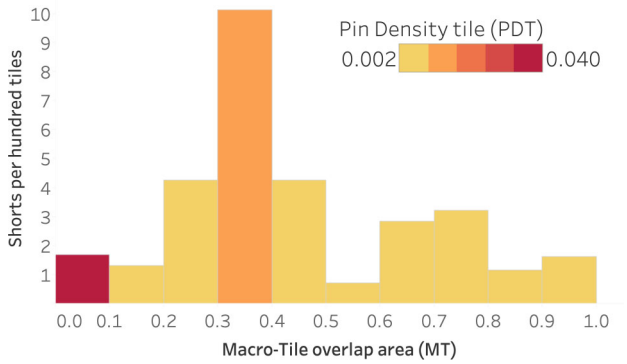

Fig. 6. Trend of shorts per hundred tiles for macro-tile overlap area (MT). Color shows average of tile pin density.

set, more than half of the shorts belong to the tiles with NDR. The probability of shorts in tiles with NDR pins (2.8%) is more than twice as the tiles without NDR pins (1.3%).

4) Feature Normalization: Before inputting the instances into the network, the features were normalized based on the mean and standard deviation of each feature in the data set.

5) Evaluation: The actual routing and placement of the tiles are analyzed using different data visualization techniques. After obtaining each round of results, we analyzed the results based on their performance with special attention given to regions with underperforming predictions. The differences between these regions and the regions with high-preforming results are analyzed to find out the specific characteristics that

might have been missed. For example, clock pins were added as a feature after these analysis. The feature extraction process terminated when introducing new features did not result in change in the performance of the predictor.

F. Learning Model Selection and Development

A supervised machine learning approach is used to perform the classification. In selecting a suitable model, the complexity of the model and its fit to the data must be balanced. The model must be complex enough to capture the features and simple enough not to over fit. Also, since our problem is an unbalanced classification problem, a model that can handle the unbalanced data should be selected [31]. We have selected some candidate models, customized the model by modifying the architecture and tuning the hyperparameters until satisfactory performance is achieved. Finally, we have selected a customized deep learning model as our main learning model that outperforms the other candidate models. This model is applied to all the circuits. The details of customized model is given in Section IV-A.

IV. Experimental Setup and Results

The experiments are performed on the ISPD15 routability driven benchmark set designs [17]. The salient characteristics of the ISPD 2015 benchmark circuits with 45 nm/28 nm design rules are high area utilization, routing layer blockages, fence regions, and fixed macros and narrow channels [17]. We have used Eh?Placer, which includes a complete flow [15], for placing the designs and generating the placed netlists. Mentor Graphics Olympus router [32] is used for routing the generated placed netlists and labeling the training data.

The feature extraction process is developed as an add-on to Eh?Placer. The source codes for feature extraction are written in C++ and compiled by gcc version 4.8.5 under Linux CentOS 7.3 with two Intel Xeon E5-2620 processors running at 2.00 GHz and 64 GB of RAM. The source codes of our supervised learning model is developed in Python using tensorflow library [33]. Training and testing are performed under the same Linux machine the feature extraction source codes were developed.

In this section, first we present the details of the implementation of our model in Section IV-A. Then, in Section IV-B we elaborate the prediction results, runtime, and comparisons.

A. Setup

In this section, the details of implementation, including determining the grid size, customizing the learning model, and generating training and test sets are elaborated to enable the reader to regenerate the codes.

1) Determining the Grid Size and the Neighborhood Size: The grid size is selected based on a tradoff between the size of data set and the noise posed by a small grid size. A big grid size leaves us with a smaller data set and very low number of positive instances for training. Also, large tiles do not capture local issues. Moreover, prediction on a big tile might not be very useful and could be even counter productive as the process to alter the placement to avoid the predicted violation can result in new violations. On the other hand, the small grid size results in noise in data and the data extracted from such a small grid are not meaningful as there would not be enough distinction for feature values. In order to benefit from advantages of both small and large grid size, we selected our grid size relatively small and considered a larger neighborhood area of each tile to extract the features from. Empirically, we have realized that we can achieve the best results by selecting three rows by three rows as the most suitable grid size and nine rows by nine rows as the most suitable neighborhood size.

2) Customizing the Learning Model: The process of developing a deep learning model for a certain application is an empirical process and consists of the iterative process of idea, implementation, and experiment [21]. The final setting used for evaluation of our method is as follows.

We have modeled the problem as a binary classification with one output node. Customizing our model includes setting the network architecture, tuning the hyperparameters, and planning to avoid overfitting.

Emperically, we have found four layers network most suitable for modeling our problem. Using more layers tends to over-fit for the available data set and less layers results in weaker performance. Input layer consists of 43 nodes which is equal to the size of feature vector. Each of three hidden layers of our customized network consists of 100 nodes. Output layer has one node for binary classification. Relu function [21] is used as activation function for the nodes. Relu has constant gradient which results in faster training compared to other activation functions such as sigmoid or tanh functions, where vanishing gradient problem can occur.

Initial weights are random values from a truncated normal distribution. The generated values follow a normal distribution with mean 0 and STD of 1, except that values whose magnitude is more than 2 STD from the mean are dropped and repicked. Initial bias values are set to 0.

We have used weighted cross entropy as loss function with weight of $1/\text{ratio} \approx 54$ for positive error, where ratio is the ratio of the number of tiles with shorts to the normal tiles. This allows us to make up for the imbalance in our data and tradeoff sensitivity and specificity by giving a higher cost to a positive error relative to a negative error in calculation of loss function. Our model penalizes the miss-classification of positive instances by 54 times more than negative instances.

The solver used for optimizing the loss function is Adam optimizer with the learning rate of 0.05 [34]. Adam optimization method is a gradient-based optimization method with adaptive learning rate, which uses the running averages of both the gradients and the second moments of the gradients and results in faster convergence. The gradient descent is too slow and mini-batch methods are not appropriate choices for this problem due to the imbalance in data. In the mini-batch optimization methods, a batch of data is randomly selected in each iteration and their derivatives are calculated. Since in random the chance of having samples from positive class is very low, it is not suitable for our problem and the classifier tends to assign most of the samples to negative class. Whole batch optimization is used in these experiments. In order to avoid overfitting, dropout method with keeping rate of 0.95 is

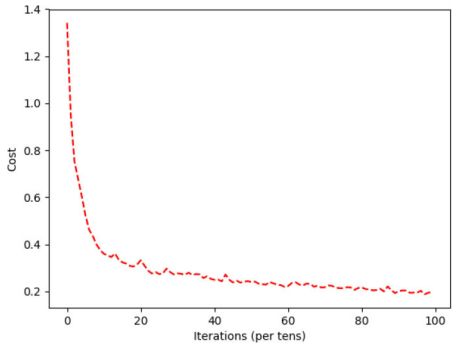

Fig. 7. Training cost versus iteration number.

applied on all hidden layers. Early stopping is applied when the MCC value of the validation set starts decreasing.

The maximum number of training iterations is set to 1000. In order to set this parameter, the change in loss function is monitored in several experiments and the number of iterations is selected. Fig. 7 shows the plot of the training loss or cost versus number of iterations.

3) Generating Training and Test Sets: In order to train and evaluate our model, the instances are divided into training set and test set. Data from each design is split randomly into training set (80%) and test set (20%). Test set is only used for evaluation after the training is completed. Hold-out validation is applied to training set to test the generalization of the model while training. 20% of the tiles in training set is randomly selected and assigned to validation set and used for customizing the model and tuning the hyperparameters. It should be noted that *mgc_edit_dist_a* and *mgc_superblue16a* designs are not included in the training and test sets as they could not be detailed routed.

B. Results

The results of detecting short violations using Eh?Predictor are summarized in Table II as confusion matrices. In this table, the rows of the matrices present actual normal tiles and tiles with violations, while each column shows the number of prediction of each class. The total number of generated instances are about 24k, where 4.5k of them contain short violations. Overall, Eh?Predictor is able to predict 98% of violations. Also 95% of the tiles without violations are correctly predicted as negative, which means the FA rate is less than 5%. Eh?Predictor detects 99% of the shorts with 5% FA rate in training data, and 90% of the shorts with only 5% FAs on the previously unseen instances.

The performance of the proposed model on all data, as well as individual designs, is evaluated and analyzed using the results presented in Table III and Fig. 8.

In Table III, in columns 1 and 2, the design name and the total number of instances are presented. In columns 3–6 the total number of positive instances, i.e., tiles with shorts, the percentage of positive instances, the number of correctly predicted positive instances and the TPR values are given. In columns 7 and 8 the total number of correctly predicted negative instances and the SPC values are given. The last two

TABLE II
CONFUSION MATRICES FOR THE TRAINING SET, TEST SET, AND ALL DATA

Training		Prediction		Total
		N	P	
Actual	N	180,610 (% 95)	8,686	189,296
	P	21	3,574 (% 99)	3,595
				192,891

Test		Prediction		Total
		N	P	
Actual	N	45,078 (% 95)	2,304	47,382
	P	81	769 (% 90)	850
				48,232

All data		Prediction		Total
		N	P	
Actual	N	225,688 (95%)	10,990	236,678
	P	102	4343 (98%)	4,445
				241,123

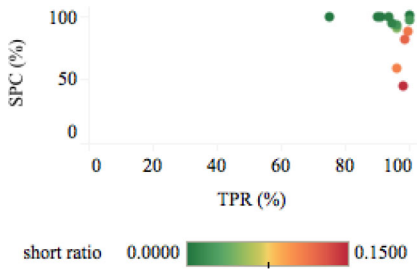

Fig. 8. Specificity versus sensitivity: each circle represent a design and color represent the ratio of positive instances to all instances of the design.

columns show the ACC and MCC values, respectively. In order to better interpret the results, the designs are ordered based on the percentage of positive instances in the design (*P%*). TPR and SPC are not applicable on the design without short violations.

In Fig. 8, each circle represents a design, where the color represents the relative short number, i.e., the ratio of the number of the tiles with shorts to the number of all tiles. Circles with the shades of green to yellow has lower number of shorts and circles with shades of yellow to red are the designs with relatively higher short number.

The overall sensitivity and specificity of the predictor on all designs are 98% and 95%, respectively. According to the table and the graph, our predictor is highly successful in predicting the shorts of the majority of the designs and can achieve a high specificity for the designs with lower relative short number.

However, this value is lower for two of the designs with the highest relative short number. That means the FA rate is higher in actual unroutable designs. The reason is that FAs mostly occur around the actual shorts. Although there are no real shorts in those tiles, they have similar conditions to their adjacent tiles that have real shorts and hence are prone to short violations. These are the tiles that might need improvements as well, during the placement to resolve the neighbor short violations. Therefore, this issue does not negatively affect the ultimate goal of the prediction, that is to improve the placement to avoid short occurrence. In Fig. 9, the predictions versus the real after routing shorts of the two design with lowest specificity rates are presented. These are the designs with high relative short number and with the shorts scattered

TABLE III
PREDICTION RESULTS OF PROPOSED FRAMEWORK ON ISPD 2015 BENCHMARK DESIGNS

Design	Instances #	P #	P %	TP #	TPR %	TN #	SPC %	ACC %	MCC [-1:1]
mgc_des_perf_b	10000	0	0	0	NA	9975	100	100	NA
mgc_superblue11_a	71152	12	0	12	100	71100	100	100	0.48
mgc_pci_bridge32_b	9791	4	0	3	75	9674	99	99	0.14
mgc_superblue12	66010	113	0	102	90	65450	99	99	0.41
mgc_matrix_mult_a	16512	33	0	30	91	16249	99	98	0.32
mgc_fft_2	3249	16	0	15	94	3204	99	99	0.56
mgc_matrix_mult_1	8281	76	1	72	95	7685	94	94	0.33
mgc_matrix_mult_b	21433	429	2	413	96	19471	93	93	0.43
mgc_fft_1	1936	39	2	39	100	1834	97	97	0.61
mgc_pci_bridge32_a	3569	163	5	157	96	3084	91	91	0.53
mgc_fft_a	6491	696	11	669	96	3370	58	62	0.34
mgc_des_perf_1	5476	617	11	615	100	4261	88	88	0.67
mgc_des_perf_a	11452	1394	12	1377	99	8159	81	83	0.58
mgc_fft_b	5771	853	15	835	98	2169	44	52	0.31
All	241123	4445	2	4343	98	225688	95	95	0.51
Training set	192891	3595	2	3574	99	180610	95	95	0.53
Test set	48232	850	2	769	90	45078	95	95	0.46

TABLE IV
COMPARISON OF THE RESULTS OF THE PROPOSED MODEL TO DIFFERENT LEARNING METHODS WITH DIFFERENT FEATURES [35], [36]

Design	(TPR %, SPC %)			MCC [-1:1]		
	[35]	[36]	Eh?Predictor	[35]	[36]	Eh?Predictor
mgc_des_perf_b	(NA , 71)	(NA , 100)	(NA , 100)	NA	NA	NA
mgc_superblue11_a	(69 , 81)	(83 , 100)	(100 , 100)	0.01	0.34	**0.48**
mgc_pci_bridge32_b	(100 , 85)	(75 , 86)	(75 , 99)	0.02	0.03	**0.14**
mgc_superblue12	(55 , 65)	(86 , 99)	(90 , 99)	0.01	**0.42**	0.41
mgc_matrix_mult_a	(30 , 96)	(15 , 98)	(91 , 99)	0.01	0.05	**0.32**
mgc_fft_2	(90 , 47)	(100 , 95)	(94 , 99)	0.02	0.29	**0.56**
mgc_matrix_mult_1	(81 , 50)	(68 , 92)	(95 , 94)	0.03	0.20	**0.33**
mgc_matrix_mult_b	(78 , 95)	(81 , 94)	(96 , 93)	0.17	0.40	**0.43**
mgc_fft_1	(78 , 78)	(95 , 87)	(100 , 97)	0.10	0.33	**0.61**
mgc_pci_bridge32_a	(64 , 73)	(95 , 79)	(96 , 91)	0.11	0.36	**0.53**
mgc_fft_a	(77 , 84)	(96 , 50)	(96 , 58)	0.11	0.28	**0.34**
mgc_des_perf_1	(96 , 53)	(95 , 85)	(100 , 88)	0.21	0.60	**0.67**
mgc_des_perf_a	(96 , 86)	(97 , 57)	(99 , 81)	0.37	0.35	**0.58**
mgc_fft_b	(82 , 83)	(95 , 48)	(98 , 44)	0.17	0.31	**0.31**
All	(89 , 89)	(93 , 93)	(98 , 95)	0.20	0.42	**0.51**
Training set	(89 , 89)	(94 , 94)	(99 , 95)	0.21	0.44	**0.53**
Test set	(68 , 76)	(90 , 93)	(90 , 95)	0.02	0.41	**0.46**

throughout the circuit. It can be seen that the majority of the shorts are predicted in these designs and the FAs are around the actual shorts. In such designs, it is very important to obtain a good sensitivity, as these are the designs that are actually unroutable and detecting their shorts in the placement stage gives the opportunity to plan for avoiding them. The results illustrate that our predictor is highly successful in predicting the shorts of all the designs including the designs with high relative short number. There is only one design with sensitivity of less than 90% which has only four tiles with shorts and our predictor predicted three of them and resulted in 75% sensitivity. Generally, missing only a few shorts in such a small number of shorts results in big decrease in the sensitivity percentage. However, those designs are considered routable and generally, detailed routers are capable of fixing a small number of the shorts in circuits with very few shorts. The FA rate for the routable designs are low and as result, using this predictor as a guide at the placement stage will not result in major changes in the designs that are already routable.

In Table IV, the performance of the Eh?Predictor is compared to the performance of RUSBoost ensemble method using less features in [35] and a shallow ANN in [36]. In this table, first, the TPR, SPC, and MCC of the proposed method are compared to those of the RUSBoost model. It can be seen that the proposed model outperforms RUSBoost for the majority of the designs in terms of MCC as a single metric and can maintain a better tradeoff between sensitivity and specificity. Since the tiling method of the two methods are different, the SPC values for designs with macros are not comparable. In calculation of the SPC of the proposed model the area used by macros are excluded. The next set of columns show the prediction results of the method in [36]. Again, it can be observed that the proposed method outperforms this model by comparing the MCC metric.

ANN is based on combining the features and uses all the features. It can be more accurate in problems where there are no dominant features. RUSBoost is based on decision trees and does not use the features it does not find useful.

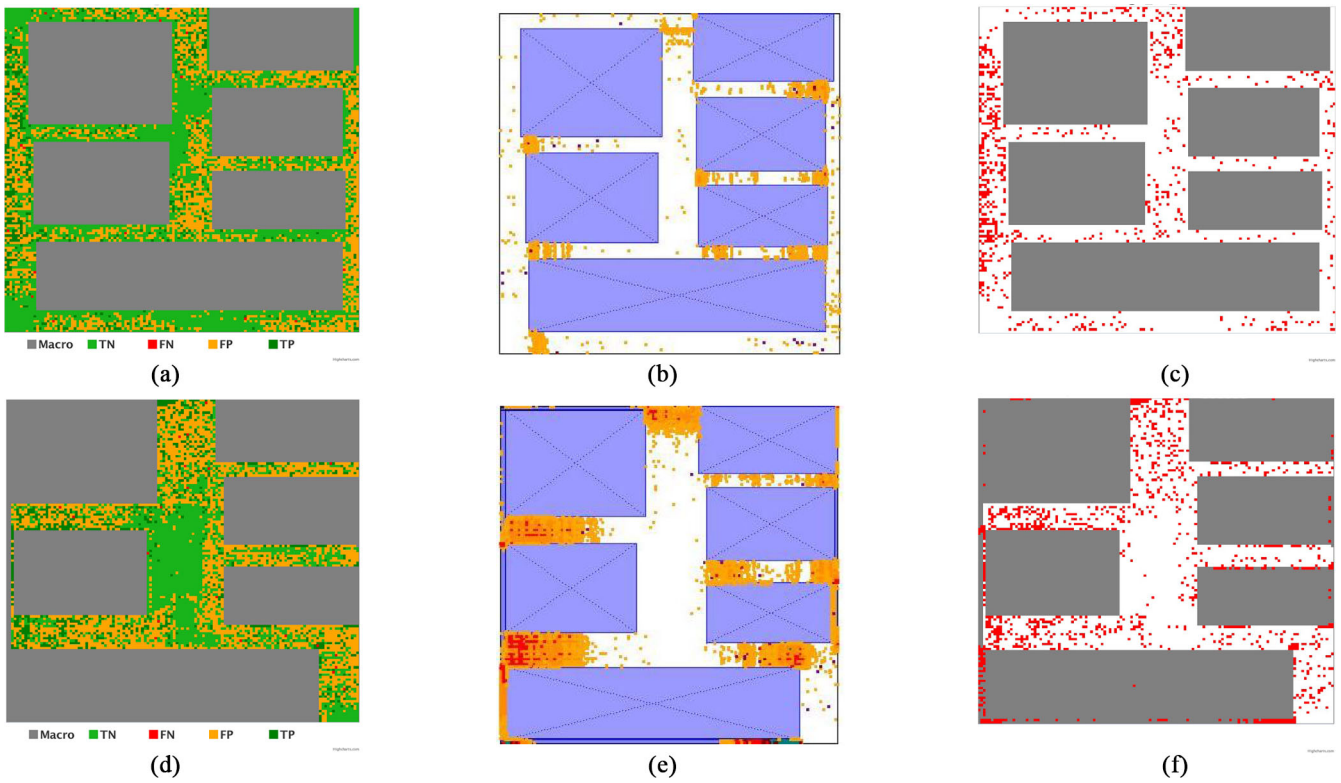

Fig. 9. mgc_fft_a and mgc_fft_b floorplans showing (a) Eh?Predictor prediction results, (b) global router congestion map, and (c) real detailed routing short violations. Here, dark green in (a) represents the shorts that are correctly detected and the light green shows the tiles correctly predicted as normal tiles. Yellow represents the incorrectly predicted shorts or FAs and red shows the shorts that are missed by the predictor. (b) Congestion map generated by Mentor Graphics global router and (c) real short violations after detailed routing by Mentor Graphics Olympus router.

TABLE V
COMPARISON OF THE RESULTS OF THE PROPOSED FRAMEWORK TO A DIFFERENT LEARNING METHODS WITH SAME FEATURES

	$(TPR\ \%, SPC\ \%)$			MCC [-1:1]		
Design	RUSBoost	Shallow ANN	Eh?Predictor	RUSBoost	Shallow ANN	Eh?Predictor
mgc_des_perf_b	(NA , 99)	(NA , 94)	(NA , 100)	NA	NA	NA
mgc_superblue11_a	(50 , 100)	(92 , 100)	(100 , 100)	0.35	0.25	**0.48**
mgc_pci_bridge32_b	(25 , 100)	(75 , 92)	(75 , 99)	0.07	0.05	**0.14**
mgc_superblue12	(53 , 99)	(90 , 99)	(90 , 99)	0.27	0.38	**0.41**
mgc_matrix_mult_a	(3 , 98)	(58 , 93)	(91 , 99)	0.00	0.09	**0.32**
mgc_fft_2	(100 , 97)	(100 , 91)	(94 , 99)	0.37	0.22	**0.56**
mgc_matrix_mult_1	(13 , 95)	(84 , 90)	(95 , 94)	0.04	0.22	**0.33**
mgc_matrix_mult_b	(76 , 91)	(94 , 83)	(96 , 93)	0.31	0.28	**0.43**
mgc_fft_1	(90 , 81)	(95 , 90)	(100 , 97)	0.25	0.37	**0.61**
mgc_pci_bridge32_a	(20 , 96)	(89 , 72)	(96 , 91)	0.16	0.28	**0.53**
mgc_fft_a	(100 , 21)	(98 , 41)	(96 , 58)	0.16	0.25	**0.34**
mgc_des_perf_1	(99 , 82)	(97 , 79)	(100 , 88)	0.58	0.52	**0.67**
mgc_des_perf_a	(100 , 41)	(99 , 51)	(99 , 81)	0.28	0.33	**0.58**
mgc_fft_b	(100 , 21)	(98 , 28)	(98 , 44)	0.19	0.22	**0.31**
All	(91 , 92)	(97 , 91)	(98 , 95)	0.38	0.38	**0.51**

In order to evaluate the efficiency of the proposed learning model and the proposed feature set in Eh?Predictor, other learning models with same feature sets and also same learning model with different feature sets are implemented as well, and the results are compared.

In Table V, the performance of the proposed model is compared to the performance of a model using RUSBoost and a shallow ANN using same feature set. The weak learner used with RUSBoost method is a decision tree with the leaf number of 30 and learning rate of 0.1. The shallow ANN used in these experiments is a 2-layer network with 20 nodes in its hidden layer. For minimizing the loss function of ANN Adam optimizer with the learning rate 0.25 [34] is used. The number of iterations is set to 3000. The weight parameter is set to 20. That is to say the model penalizes the miss-classification of positive instances by 20 times more than negative instances. Comparing the MCC value of these models as a single metric, the proposed model outperforms the other two models.

In Table VI, the performance of the proposed model is compared to the performance of the same model using different set of features to show the importance of inclusive feature set. In this table, the first set of columns show sensitivity

TABLE VI
COMPARISON OF THE RESULTS OF THE PROPOSED FRAMEWORK TO THE SAME LEARNING MODEL WITH DIFFERENT FEATURE SET

	(TPR %, SPC %)			MCC [-1:1]		
Design	Feature set 1	Feature set 2	Eh?Predictor	Feature set 1	Feature set 2	Eh?Predictor
mgc_des_perf_b	(NA , 93)	(NA , 100)	(NA , 100)	NA	NA	NA
mgc_superblue11_a	(83 , 100)	(50 , 100)	(100 , 100)	0.36	0.21	**0.48**
mgc_pci_bridge32_b	(100 , 93)	(100 , 96)	(75 , 99)	0.07	0.1	**0.14**
mgc_superblue12	(90 , 99)	(84 , 99)	(90 , 99)	0.37	0.31	**0.41**
mgc_matrix_mult_a	(73 , 97)	(85 , 98)	(91 , 99)	0.17	0.26	**0.32**
mgc_fft_2	(100 , 98)	(100 , 99)	(94 , 99)	0.41	0.55	**0.56**
mgc_matrix_mult_1	(89 , 94)	(93 , 94)	(95 , 94)	0.31	**0.33**	**0.33**
mgc_matrix_mult_b	(98 , 90)	(98 , 93)	(96 , 93)	0.39	**0.44**	0.43
mgc_fft_1	(100 , 89)	(95 , 96)	(100 , 97)	0.37	0.56	**0.61**
mgc_pci_bridge32_a	(94 , 87)	(99 , 91)	(96 , 91)	0.46	**0.55**	0.53
mgc_fft_a	(97 , 54)	(98 , 54)	(96 , 58)	0.32	0.32	**0.34**
mgc_des_perf_1	(99 , 88)	(1 , 89)	(100 , 88)	0.66	-0.1	**0.67**
mgc_des_perf_a	(99 , 80)	(100 , 85)	(99 , 81)	0.57	**0.64**	0.58
mgc_fft_b	(99 , 44)	(99 , 43)	(98 , 44)	**0.31**	**0.31**	**0.31**
All	(99 , 94)	(98 , 95)	(98 , 95)	0.47	0.49	**0.51**

TABLE VII
PREDICTION RESULTS OF PROPOSED FRAMEWORK ON ISPD 2015 BENCHMARK DESIGNS WITH A DIFFERENT PLACEMENT SOLUTION THAT ARE NOT USED IN TRAINING

Design	Instances #	P #	P %	TP #	TPR %	N #	TN #	SPC %	ACC %	MCC [-1:1]
mgc_des_perf_b	10000	0	0	0	NA	10000	9783	98	98	NA
mgc_superblue11_a	71152	7	0	6	86	71144	71081	100	100	0.27
mgc_pci_bridge32_b	9791	8	0	4	50	9782	9375	96	96	0.07
mgc_superblue12	66010	189	0	59	31	65820	65153	99	99	0.15
mgc_matrix_mult_a	16512	903	5	93	10	15608	15121	97	92	0.09
mgc_fft_2	3249	16	0	14	88	3232	3203	99	99	0.53
mgc_matrix_mult_1	8281	0	0	0	NA	8280	7863	95	95	NA
mgc_matrix_mult_b	21433	629	3	449	71	20803	19585	94	94	0.41
mgc_fft_1	1936	39	2	38	97	1896	1830	97	96	0.59
mgc_fft_a	6491	356	5	312	88	6134	3368	55	56	0.19
mgc_des_perf_1	5476	701	13	622	89	4774	4365	91	91	0.69
mgc_des_perf_a	11452	765	7	745	97	10686	8529	80	81	0.44
mgc_fft_b	5771	468	8	418	89	5302	2693	51	54	0.22

and specificity values for different configurations and the second set of columns show the corresponding MCC values. In each set of columns, first column shows the results of model using feature set 1. Feature set 1 includes all the features except global features (feature number 1, 2). Second column shows the results of predictor using feature set 2. Feature set 2 includes the features from main feature vector excluding features related to connectivity and special pins (feature number 31–43). Third column shows the results of Eh?Predictor using the proposed feature set. The results show that using the proposed feature set results in better predictions.

The number of shorts in our dataset is limited and the shorts obtained from the benchmark circuits are not evenly distributed between all the designs. For example, more than 25% of the shorts in dataset is obtained from design mgc_des_perf_a. Moreover, as presented in Fig. 4, most of the shorts are obtained from designs with high density, and designs with blockages. These characteristics are modeled to global features of the tiles and different designs with different global features are needed to train the model. For example, more than 80% of the shorts from high density designs are obtained from design mgc_des_perf1. Therefore, excluding all tiles of one design from training set results in loss of global feature information and model cannot achieve its best performance. In Table VII, the performance of the predictor on completely unseen designs whose placement and routing information are not used in training are presented. In generating this table, the model is trained using all the designs and tested on the designs with different placement solution that are generated by using a different version of Eh?Placer [15]. This model can be constantly updated by adding more labeled data if available, i.e., different placed and routed designs and can be used for predicting new designs with similar global characteristics.

C. Runtime

Many placement tools, such as Eh?Placer, exploit a routing estimator as a guide to reduce global routing congestion and detailed routing issue during the placement process. For this reason, we compare the runtime of our predictor with NCTUgr, which is used in Eh?Placer during the ISPD 2015 contest.

In Table VIII, the runtime of predictor is compared to the runtime of NCTU global routing estimator. It takes a fraction of a second to call the predictor, where each call of NCTUgr can take up to 22 s predictor can save a significant runtime of

TABLE VIII
COMPARISON OF THE RUNTIME (S) OF THE PROPOSED PREDICTOR TO ONE CALL OF NCTU GLOBAL ROUTER

Design	NCTUgr	predictor	
mgc_des_perf_1	0.55	0.08	7 X
mgc_des_perf_a	0.81	0.11	7 X
mgc_des_perf_b	0.66	0.10	7 X
mgc_fft_1	0.13	0.01	14 X
mgc_fft_2	0.14	0.01	11 X
mgc_fft_a	0.21	0.06	3 X
mgc_fft_b	0.21	0.06	3 X
mgc_matrix_mult_1	0.71	0.09	8 X
mgc_matrix_mult_a	0.99	0.11	9 X
mgc_matrix_mult_b	0.97	0.12	8 X
mgc_pci_bridge32_a	0.16	0.02	10 X
mgc_pci_bridge32_b	0.22	0.07	3 X
mgc_superblue11_a	11.72	0.25	48 X
mgc_superblue12	22.33	0.23	96 X

placement. For example, 100 calls of the predictor instead of global routing estimator in superblue12 can save up to 40 mins of placement runtime. According to the table, the runtime of global router increases rapidly with the increase in the size of circuit (number of cells). Eh?predictor is up to 14 times faster than NCTUgr for smaller designs and up to 96 times faster for larger designs.

The total runtime spent for training in this experiment using the proposed method is 1532 s. Training is a one-time task, and once a model is trained the only runtime added to the placer is a few seconds of prediction time. This can save hours of prediction during placement.

V. CONCLUSION

In this paper, we proposed a deep learning framework to identify short violations that can occur during detailed routing from a placed netlist. The major contributions of this paper are proposing a deep learning framework to predict the shorts before routing, significantly decreasing the prediction time by not using a global router, modeling the short prediction problem as a binary classification problem with imbalanced data, defining and extracting effective features after placement, analyzing the extracted features and their correlation with short violations, customizing a deep learning model that fits the problem and its imbalanced nature. The results of this paper show significant improvement in the accuracy of the prediction and we show that Eh?Predictor is up to 14X faster than NCTUgr for smaller designs and up to 96X faster for larger designs.

ACKNOWLEDGMENT

The authors would like to thank the Natural Sciences and Engineering Council of Canada, Canada Micro-Electronics Corporation, and Compute Canada for financial and equipment support of this paper. They would also like to thank the Mentor Graphics for their generous contribution to this paper.

REFERENCES

[1] M. Pan and C. Chu, "IPR: An integrated placement and routing algorithm," in *Proc. DAC*, San Diego, CA, USA, 2007, pp. 59–62.

[2] J. A. Roy, N. Viswanathan, G.-J. Nam, C. J. Alpert, and I. Markov, "CRISP: Congestion reduction by iterated spreading during placement," in *Proc. ICCAD*, San Jose, CA, USA, 2009, pp. 357–362.

[3] H. Shojaei, A. Davoodi, and J. T. Linderoth, "Congestion analysis for global routing via integer programming," in *Proc. ICCAD*, 2011, pp. 256–262.

[4] Y. Wei *et al.*, "GLARE: Global and local wiring aware routability evaluation," in *Proc. DAC*, San Francisco, CA, USA, 2012, pp. 768–773.

[5] J. Melchert, B. Zhang, and A. Davoodi, "A comparative study of local net modeling using machine learning," in *Proc. GLSVLSI*, 2018, pp. 273–278.

[6] L.-T. Wang, Y.-W. Chang, and K.-T. Cheng, *Electronic Design Automation: Synthesis, Verification, and Test* (Systems on Silicon). Burlington, MA, USA: Elsevier Sci., 2009.

[7] I. L. Markov, J. Hu, and M.-C. Kim, "Progress and challenges in VLSI placement research," in *Proc. ICCAD*, 2012, pp. 275–282.

[8] M. Garey and D. Johnson, "*Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: Freeman, 1990.

[9] W. E. Donath, "Complexity theory and design automation," in *Proc. DAC*, Minneapolis, MN, USA, 1980, pp. 412–419.

[10] Y. Ding, C. Chu, and W. Mak, "Pin accessibility-driven detailed placement refinement," in *Proc. ISPD*, 2017, pp. 133–140.

[11] W. Liu, C. Koh, and Y. Li, "Optimization of placement solutions for routability," in *Proc. DAC*, 2013, pp. 1–153.

[12] C. Alpert and G. Tellez, "The importance of routing congestion analysis," in *Proc. DAC Knowl. Center*, 2010, pp. 1–14.

[13] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, 1st ed. Dordrecht, The Netherlands: Springer, 2011.

[14] R. Kastner, E. Bozorgzadeh, and M. Sarrafzadeh, "Predictable routing," in *Proc. ICCAD*, 2000, pp. 110–114.

[15] N. K. Darav, A. Kennings, A. Tabrizi, D. Westwick, and L. Behjat, "Eh?placer: A high-performance modern technology-driven placer," *ACM Trans. Design Autom. Elect. Syst.*, vol. 21, no. 3, pp. 1–37, 2016.

[16] X. He *et al.*, "Ripple 2.0: High quality routability-driven placement via global router integration," in *Proc. DAC*, 2013, pp. 1–6.

[17] I. S. Bustany, D. Chinnery, J. R. Shinnerl, and V. Yutsi, "ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement," in *Proc. ISPD*, 2015, pp. 157–164.

[18] Q. Ma and E. F. Y. Young, "Multivoltage floorplan design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 4, pp. 607–617, Apr. 2010.

[19] K. Wang and M. Marek-Sadowska, "Power/ground mesh area optimization using multigrid-based technique," in *Proc. DATE*, 2003, pp. 850–855.

[20] N. K. Darav, I. Bustany, A. Kennings, and R. Mamidi, "ICCAD-2017 cad contest in multi-deck standard cell legalization and benchmarks," in *Proc. ICCAD*, Irvine, CA, USA, 2017, pp. 867–871.

[21] N. G. Andrew. (2018). *Machine Learning Yearning, Technical Strategy for AI Engineers in the Era of Deep Learning*. Accessed: Sep. 2018. [Online]. Available: http://www.mlyearning.org

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[23] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[24] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: Special issue on learning from imbalanced data sets," *SIGKDD Explor. Newslett.*, vol. 6, no. 1, pp. 1–6, Jun. 2004.

[25] L.-C. Wang, "Experience of data analytics in EDA and test—Principles, promises, and challenges," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 6, pp. 885–898, Jun. 2017.

[26] J. A. Torres, "ICCAD-2012 cad contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. ICCAD*, San Jose, CA, USA, 2012, pp. 349–350.

[27] Y.-T. Yu, G.-H. Lin, I. H.-R. Jiang, and C. Chiang, "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *Proc. DAC*, Austin, TX, USA, 2013, pp. 1–67.

[28] L.-C. Wang, P. Bastani, and M. S. Abadir, "Design-silicon timing correlation: A data mining perspective," in *Proc. DAC*, San Diego, CA, USA, 2007, pp. 384–389.

[29] L.-C. Chen, C.-C. Huang, Y.-L. Chang, and H.-M. Chen, "A learning-based methodology for routability prediction in placement," in *Proc. VLSI-DAT*, 2018, pp. 1–4.

[30] A. F. Tabrizi, N. K. Darav, L. Rakai, A. Kennings, W. Swartz, and L. Behjat, "A detailed routing-aware detailed placement technique," in *Proc. ISVLSI*, Jul. 2015, pp. 38–43.

[31] M. Kearns, Y. Mansour, A. Ng, and D. Ron, "An experimental and theoretical comparison of model selection methods," *Mach. Learn.*, vol. 27, no. 1, pp. 7–50, Apr. 1997.
[32] "Olympus-SoC place and route for advanced node designs," Mentor Graph., Inc., Wilsonville, OR, USA, Rep. MGC 05-16 3181:070813. Accessed: Feb. 16, 2015. [Online]. Available: www.mentor.com/products/ic_nanometer_design/place-route/olympus-soc
[33] M. Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Accessed: Sep. 24, 2018. [Online]. Available: http://tensorflow.org/
[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015. [Online]. Available: http://arxiv.org/abs/1412.6980
[35] A. F. Tabrizi, N. K. Darav, L. Rakai, A. Kennings, and L. Behjat, "Detailed routing violation prediction during placement using machine learning," in *Proc. VLSI-DAT*, 2017, pp. 1–4.
[36] A. F. Tabrizi *et al.*, "A machine learning framework to identify detailed routing short violations from a placed netlist," in *Proc. DAC*, 2018, Art. no. 48.

Aysa Fakheri Tabrizi received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Tabriz, Tabriz, Iran, in 2007 and 2010, respectively, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Calgary, Calgary, AB, Canada, in 2013 and 2018, respectively.

Her current research interest includes machine learning applications in engineering problems.

Dr. Tabrizi was a member of the team that won first and second place in the ISPD 2014 and ISPD 2015 Routability Driven Placement Contest.

Nima Karimpour Darav received the B.Sc. degree in computer engineering from Shahid Beheshti University, Tehran, Iran, in 2004, and the M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, in 2007, and the Ph.D. degree in electrical and computer engineering from the University of Calgary, Calgary, AB, Canada, in 2017.

In 2017, he joined Microsemi Corporation, Aliso Viejo, CA, USA, researching on analytic placement for field-programmable gate arrays. Since 2013, he has been researching on several collaborative projects on physical designs, such as "Eh?Placer," "En?Legalizer," and "Eh?Predictor." Eh?Placer achieved first and second place in the ISPD 2014 and ISPD 2015 High-Performance Routability Driven Placement Contest.

Logan Rakai (M'12) received the B.Sc. degree (Hons.) in computer engineering and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Calgary, Calgary, AB, Canada, in 2007, 2008, and 2012, respectively.

He is currently an Adjunct Professor with the Department of Electrical and Computer Engineering, Schulich School of Engineering, University of Calgary, and he strives to bring out the best in students. His current research interests include using distributed and high-performance computing, machine learning, and specialized solvers to solve engineering problems in cloud computing, computer-aided design, and power systems.

Ismail Bustany received the M.S. and Ph.D. degrees in electrical engineering from University of California at Berkeley, Berkeley, CA, USA.

He is a Distinguished Engineer with Xilinx Inc., San Jose, CA, USA. His current research interests include physical design, computationally efficient optimization algorithms, sparse matrix computations, hardware acceleration, and machine learning.

Dr. Bustany has served on the technical programming committees for the ISPD, the ISQED, and DAC. He organized the 2014 and 2015 ISPD detailed routing-driven placement contests and co-organized the 2017 ICCAD detailed placement contest. He has chaired various technical sessions, and was the General Chair of the 2019 ISPD.

Andrew Kennings received the B.A.Sc., M.A.Sc., and Ph.D. degrees in electrical engineering from the University of Waterloo, Waterloo, ON, Canada, in 1992, 1994, and 1997, respectively.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, University of Waterloo. His current research interests include all aspects of computer-aided design for field-programmable gate arrays and application-specified integrated circuits.

Laleh Behjat received the B.Sc. degree from Tehran University, Tehran, Iran, in 1996 and the M.Sc. and Ph.D. degrees from the University of Waterloo, Waterloo, ON, Canada, in 1999 and 2002, respectively.

She joined the University of Calgary, Calgary, AB, Canada, in 2002, where she is a Professor with the Department of Electrical and Computer Engineering, Schulich School of Engineering. Her current research interests include developing EDA techniques for physical design and application of large-scale optimization in EDA. She has been developing new and innovative methods to teach Computer Science and EDA to students.

Dr. Behjat's research team has won several awards, including first and second places in ISPD 2014 and ISPD 2015 High Performance Routability Driven Placement Contests and third place in DAC Design Perspective Challenge in 2015. She is an Associate Editor of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and *Optimization in Engineering* (Springer).