Lookahead Placement Optimization with Cell Library-based Pin Accessibility Prediction via Active Learning*

Tao-Chun Yu[1], Shao-Yun Fang[1], Hsien-Shih Chiu[2], Kai-Shun Hu[2], Philip Hui-Yuh Tai[2], Cindy Chin-Fang Shen[2], and Henry Sheng[2]

[1]Department of Electrical Engineering, National Taiwan Univer
[2]Synopsys Taiwan Co., Ltd., T
{d10407012, syfang}@mail

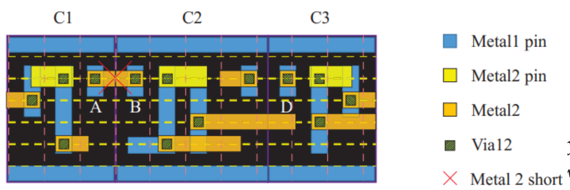


Figure 1: Bad pin accessibility on C_2 influenced by C_1.

ABSTRACT

With the development of advanced process nodes of semiconductor, the problem of pin access has become one of the major factors to impact the occurrences of design rule violations (DRVs) due to complex design rules and limited routing resource. Many state-of-the-art works address the problem of DRV prediction by adopting supervised machine learning approaches. However, those supervised learning approaches extract the labels of training data by generating a great number of routed designs in advance, giving rise to large effort on training data preparation. In addition, the pre-trained model could hardly predict unseen data and thus may not be applied to predict other designs containing cells that are not used in the training data. In this paper, we propose the first work of cell library-based pin accessibility prediction (PAP) by using active learning techniques. A given set of standard cell libraries is served as the only input for model training. Unlike most of existing studies that aim at design-specific training, we propose a library-based model which can be applied to all designs referencing to the same standard cell library set. Experimental results show that the proposed model can be applied to predict two different designs with different reference library sets. The number of remaining DRVs and M2 shorts of the designs optimized by the proposed model are also much fewer than those of design-specific models.

CCS CONCEPTS

• **Hardware → Placement**.

KEYWORDS

pin accessibility prediction, placement optimization, active learning

ACM Reference Format:
Tao-Chun Yu[1], Shao-Yun Fang[1], Hsien-Shih Chiu[2], Kai-Shun Hu[2], Philip Hui-Yuh Tai[2], and Cindy Chin-Fang Shen[2], and Henry Sheng[2]. 2020. Lookahead Placement Optimization with Cell Library-based Pin Accessibility Pre ym-
posi van.
ACI 562

*This work was partially supported by Synopsys, TSMC, and MOST of Taiwan under Grant No's MOST 108-2636-E-011-002.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ISPD '20, March 29-April 1, 2020, Taipei, Taiwan
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7091-2/20/03…$15.00
https://doi.org/10.1145/3372780.3375562

1 INTRODUCTION

Due to the shrinking of modern process nodes of semiconductor, the pin access problem of standard cells has become more harder to be coped with, especially on the lower metal layers. Fig. 1 gives an example of the bad pin accessibility of cell $C2$ affected by the adjacent cell $C1$. Note that the horizontal yellow dashed lines and vertical red dashed lines represent metal 2 (M2) tracks and metal 3 (M3) tracks, respectively. Obviously, the M2 routing segment from pin A can only go right to access the M3 track since there is an M2 pin occupying the routing resource on the left. It can be observed that the access points of Pin B are then blocked by the M2 routing segment routed from Pin A, so an M2 short DRV will be induced when dropping a Via12 on Pin B, resulting in bad pin accessibility. This example reveals that pin accessibility has strongly correlation with adjacent cells.

To resolve the pin access problem, many existing works try to enhance the pin accessibility during layout design of standard cells [9–15]. [9, 12, 15] focus on self-aligned double patterning (SADP) lithography and consider pin access planning under design constraints. [14] and [10] solve the pin access optimization problem with integer linear programming (ILP). Then, the constraints are relaxed to the objective function and the formulation is solved by using an iterative Lagrangian relaxation algorithm to speedup the overall runtime. For better pin accessibility,[11] proposes a heuristic placement co-optimization approach to determine whether a cell requires an extra whitespace or needs to be redesigned, and [13] proposes a dynamic programming-based detailed placement approach to iteratively refine cell positions. However, the aforementioned studies optimize pin accessibility based on human cognition and only focus on the layout of a single cell, making them suffer from a disadvantage that their pre-defined objectives may miss some critical but complicated factors having a great correlation with the pin access problem, especially for extremely large-scale designs in advanced process node.

Due to the difficulty of analyzing DRV occurrences caused by bad pin accessibility, recent works try to directly predict the DRV locations by means of supervised machine learning-based techniques [1–8]. [1] and [7] propose support vector machine (SVM)-based methods to respectively predict the locations of DRVs and the routability of placements. [2] and [4] predict the number of DRVs and congested regions of a given cell placement by considering global routing (GR) congestion maps. [8] proposes a convolutional neural network (CNN)-based routability predictor to forecast overall routability of a placement with accuracy similar to that of global

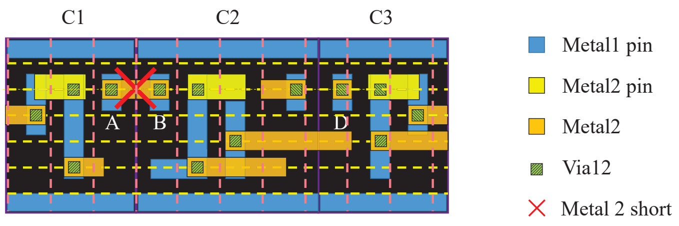


Figure 1: Bad pin accessibility on C_2 influenced by C_1.

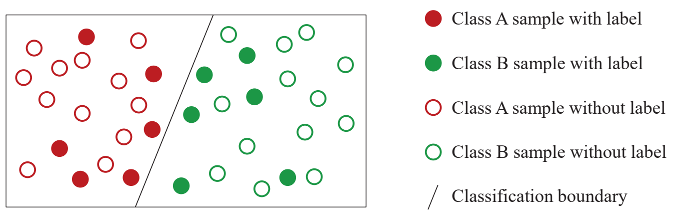

Class A sample with label
Class B sample with label
Class A sample without label
Class B sample without label
Classification boundary

Figure 2: The concept of active learning.

router while using less runtime. [6] and [3] respectively use the ensemble RUSBoost algorithm and a weighted neural network to predict whether a given tile has M2 short violations or not. [5] applies the pin pattern image as the main feature instead of using GR congestion map and pin density map to catch exact pin patterns having bad pin accessibility. The experimental results not only show the dramatic reductions in both DRV counts and M2 short counts but also show its better prediction accuracy of DRV occurrence than the previous works.

Deriving a pin access score with a given pin pattern, [5] has the ability to identify cell combinations with bad pin accessibility; however, it may suffer from two weaknesses: (1) the model can only be trained after generating a great number of routed designs due to the labeling requirement of supervised learning approaches. This may take many days or even few weeks for collecting training data. (2) The trained model is not necessarily applicable to other designs using different cells or different reference cell libraries. To facilitate the management of standard cells, standard cell library designers usually develop a set of standard cell libraries, each of which has different characteristics such as low/high drivability and low/high threshold voltage. An industrial design may reference to not only one standard cell library. Since an arbitrary design for training is very likely to only use a part of cells of a given reference library set, the trained model may hardly analyze unseen cells in other designs even if they use the same library set.

Note that not only [5] but also all the other existing studies [1–4, 6–8] suffer from the aforementioned drawbacks. Thus, we are inspired to train a cell library-based pin accessibility prediction model that does not require large effort to prepare routed designs, which is applicable to all designs referencing to the same cell library set, and can be adopted to optimize cell placement. The key idea of our work is to simply consider the cell combinations from a given cell library set to obtain the training data. However, the number of cell combinations derived from thousands of library cells is astronomical, and thus it is prohibitively expensive to label every cell combination using EDA tools. This challenge motivates us to adopt active learning to extract critical cell combinations that can effectively and efficiently train our PAP model. The major contributions of this paper are summarized as follows:

- To the best of our knowledge, this is the first work that applies active learning to train a PAP model. With the proposed good querying strategies, the model is able to query the labels of the most representative and informative cell combinations. Thus, the performance of the model can be iteratively improved by training on those critical data.
- Unlike existing design-specific models that aim at training and predicting on the same design, we proposed a library-based training flow to train the PAP model, which can be applied to predict other designs referencing to the same cell library set.
- Different from the design-specific model that can only be trained after preparing a lot of routed designs, the proposed cell library-based PAP model can be trained at the earlier stage in a process development flow: once the cell libraries are provided.
- The experimental results show that the proposed cell library-based pin accessibility prediction model can be successfully applied to predict two real industrial designs using different reference library subsets. Furthermore, the reductions in DRVs and M2 shorts contributed by the proposed model after placement re-legalization are also superior than those contributed by the design-specific model proposed in [5].

The rest of this paper is organized as follows: Section 2 introduces the preliminaries of this work and the problem formulation. Section 3 introduces the proposed pin accessbility evaluator and the adopted model architecture. In Section 4, the proposed active learning flow is detailed followed by the introduction to the placement optimization flow. Experimental results are presented in Section 5. Finally, we conclude our work in Section 6.

2 PRELIMINARIES

In this section, we first briefly introduce the basic of active learning. Then, the problem formulation is given.

2.1 Active Learning

Active learning is a suitable machine learning technique to tackle the problem with a large number of unlabeled data and expensive labeling cost, bringing a great opportunity to save the labeling cost without degrading the performance of trained models. Fig. 2 gives the brief concept of applying active learning to solve a classification problem. The red and green solid circles respectively represent the labeled data of class A and class B, and the red and green hollow circles respectively represent the unlabeled data of class A and class B. The goal of active learning is to query the labels of fewer but more important data to train a classification boundary which leads to good performance for both labeled and unlabeled data sets.

The procedure of active learning is composed of two stages:(1) iteratively sampling and labeling some queried data, and (2) retrain the model based on the queried data. For each iteration, it queries either one or a batch of samples for labeling, which are expected to be able to improve the performance of the current classification model. There are two query strategies, informativeness and representativeness, which are widely used in the previous researches working on active learning. Informativeness is regarded as the ability of a sample to improve the accuracy or reduce the overall loss of a classification model, and to achieve less uncertainty during the training of next iteration. Representativeness describes the correlation between a sample and the underlying unlabeled data.

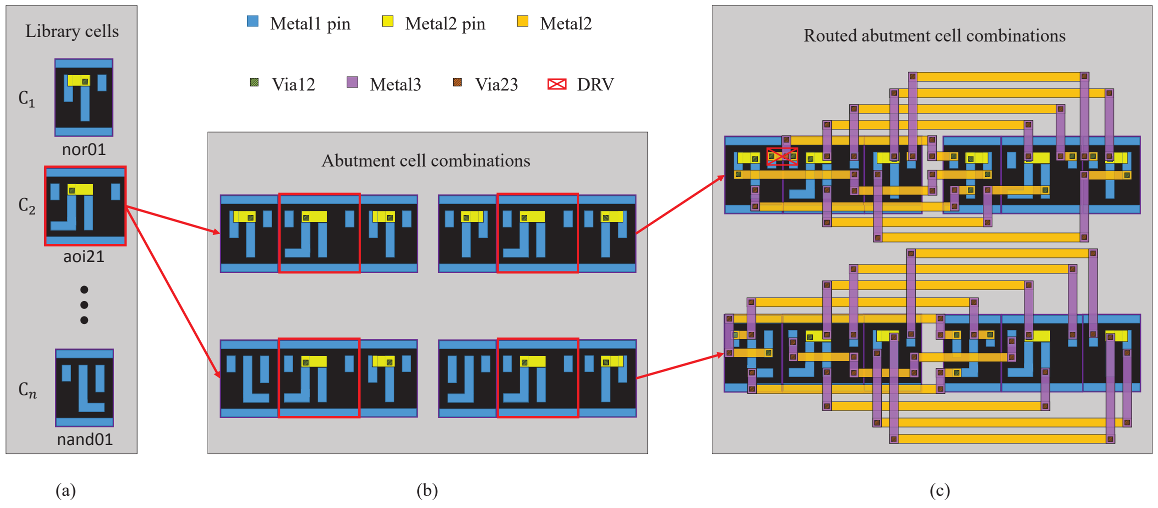


Figure 3: The procedure of evaluating pin accessibility for C_2 considering abutting cells. (a) Library cells. (b) Two examples of cell combinations. The upper one is (C_1, C_2, C_1) and the lower one is (C_n, C_2, C_1). (c) Routing results of the two cell combinations.

Many previous works have applied informativenes querying strategies to sampling and labeling data [16–26]. On the other hand, several previous works focus on representativeness querying strategies [27–31]. [26] integrates both informativeness and representativeness into one formula and proposes a quadratic programming formulation to solve the two-sample discrepancy problem. The experimental results show that the combined querying strategies is superior than those only adopting a single informativeness or representativeness querying strategy. According to the good results obtained by [26], we are inspired to consider both informativeness and representativeness querying strategies to effectively query more important cell combinations in the given cell libraries.

2.2 Problem Formulation

The goal of this paper is to train a cell library-based pin accessibility prediction model. As the name implies, the input of our approach is a set of cell libraries, $L = \{l_0, l_1, ..., l_{m-1}\}$, where m denotes the number of cell libraries. Let $C_i = \{c_0^i, c_1^i, ..., c_{n-1}^i\}$ be a set of standard cells in l_i, where n represents the number of standard cells in C_i. Note that the number of standard cells in each cell library may be different. Having the cell libraries with the corresponding standard cells, we can give the problem formulation as follows:

Problem 1: Given a set of cell libraries L, the task is to develop an active learning-based training flow to train a pin accessibility prediction model, which can be applied to predict DRV occurrences due to bad pin accessibility for designs referencing to the same cell library set.

3 PIN ACCESSIBILITY EVALUATOR AND MODEL BACKGROUNDS

In this section, the pin accessibility evaluation methods for cell combinations are detailed, and the adopted feature extraction and model architecture are introduced.

3.1 Pin Accessibility Evaluator

As shown in Fig. 1, the pin accessibility of standard cells may have a great correlation with neighboring cells. Fig. 3 shows how we evaluate the pin accessibility of a cell in combination with two neighboring cells in this paper. Suppose that we focus on the pin accessibility of cell C_2. In order to observe the pin accessibility of C_2 affected by adjacent cells, we randomly select two library cells and randomly abut them to the two sides of the target cell C_2 without PG rail violation. Then, we duplicate the created cell combination and flip the abutting cells by the y-axis. Fig. 3(b) shows two examples of cell combinations, where the upper one is (C_1, C_2, C_1) and the lower one is (C_n, C_2, C_1). Since the pin counts of the same type standard cells are the same, the netlist can be defined by connecting each pin in the left cell combination to the corresponding pin in the right (duplicated) cell combination, as shown in Fig. 3(c). After routing and design rule check, we determine whether a cell combination has poor pin accessibility by checking DRV occurrences.

3.2 Feature Extraction and Model Architecture

Throughout this paper, an input training data indicates a cell combination mentioned in Section 3.1. Empirically, global routing congestion and pin density are no more highly correlated with DRC occurrences in advanced processes, and thus using them as the major features to train the DRV prediction model may not be effective. In addition, the informations of routing congestion and the pin density in a tile are not available when we analyze a given cell library set. Thus, we adopt the feature extraction method proposed in [5] due to the good performance of using pin pattern as the main feature to train the DRV prediction model, and thus cell combinations are converted into pin patterns. To feed pin patterns as the input training data, it is necessary to quantify them as numerical values. Fig. 4 illustrates the process. Each pin pattern is first split into a two-dimensional pixel array, as shown in Figs. 4(a) and (b). Then, The value of each pixel is computed by the corresponding occupied pin ratio, which is computed by the the area covered by the overlapped pin pattern out of the pixel area, as shown in Figs. 4(c) and (d), where Fig. 4(c) is computed by the pin shapes in M1 and Fig. 4(d) is computed by the pin shape in M2.

The CNN-based architecture of the PAP model proposed in [5] is used since the CNN-based model is suitable for pattern recognition problems, as shown in Fig. 5. During the CNN computation, several

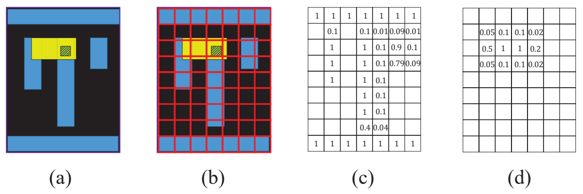


Figure 4: Feature extraction method. (a) Input pin pattern. (b) Sliced input pin pattern. (c) Numerical pixels of M1. (d) Numerical pixels of M2.

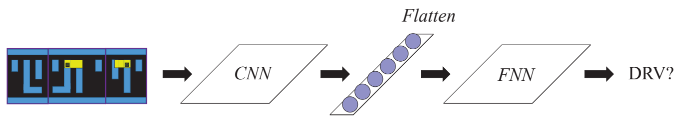


Figure 5: PAP model architecture.

representative features such as line and geometric shape of an input pin pattern will be extracted by a set of trainable filters. Furthermore, these features will be flattened into a one-dimensional vector and fed into a fully connected neural network (FNN) for binary classification. The output will be a probability of DRV occurrence of the input pattern. By pre-defining a threshold value, we can easily determine whether the input pin pattern has DRV or not. Note that if the probability is greater than the pre-defined threshold value, the input pattern will be regarded as a DRV pattern; otherwise, the input pattern will be regarded as a DRV-clean pattern. The number of used channels is determined by the maximum layers of the input pin patterns. In the case shown in Fig. 4(a), we will distinguish M1 and M2 pin patterns and assign them to two different channels. After all pin patterns are quantified into numerical values, they can be fed into the PAP model for training.

To feed pin patterns as the input training data, it is necessary to quantify them to numerical values. Therefore, each pin pattern is first split into two dimensional pixel array, as shown in Fig 4 (b). Then, The value of each pixel is computed by the corresponding occupied pin ratio, which is computed by the the area covered by the overlapped pin out of the pixel area, as shown in Fig 4 (c) and Fig 4 (d), where Fig 4 (c) represents the pin shapes in M1 and Fig 4 (d) represents the pin shape in M2. The number of used channels is based on the maximum used layers of input pin pattern set. In the case shown in Fig 4 (a), we will distinguish M1 and M2 pin patterns and assign them to channel one and channel two, respectively. After all pin patterns are quantified into numerical values, they can be fed into PAP model for training.

4 PROPOSED METHODOLOGIES FOR MODEL TRAINING AND PLACEMENT OPTIMIZATION

In this section, we introduce how to train the proposed library-based PAP model. The algorithm flow is first explained. Then, the initial cell combination generation for training the initial model and two query strategies are detailed. Finally, the model-guided placement refinement is proposed to further optimize the given legalized placement.

4.1 Algorithm Flow

The overview of the proposed approach is given in Fig. 6, which consists of two parts: (1) the active learning-based model training flow and (2) the placement optimization flow. Fig. 6(a) shows the proposed active learning-based model training flow, which can be divided into two stages: initial pattern generation, and querying strategies. As the standard cell libraries are given from foundries or design houses, we first randomly generate some cell combinations in the initial pattern generation stage. Then, the generated cell combinations are fed into ICC2 router to query the labels and all labeled cell combinations are used to train an initial PAP model. After that, we will move on to the querying strategies, which requires lookup table (LUT)-based representativeness evaluation and informativeness computation. With the two quantitative analyses, we can obtain a set of critical and unlabeled cell combinations. Finally, all labeled cell combinations will be fed into the PAP model for retraining until the performance of the model is good enough. After the optimal PAP model is obtained by the proposed model training flow, we further apply the model to guide the legalizer of ICC2 to optimize cell placements. Fig. 6(b) shows the placement optimization flow. Input is a legalized placement result, the trained model is first adopted to compute the required placement spacing rules among placed cells. Then, the required placement spacing rules are integrated into the legalizer to avoid generating bad cell combinations during the post legalization stage. Finally, routing and design rule check (DRC) are run for checking the quality of results of the post-legalized placement.

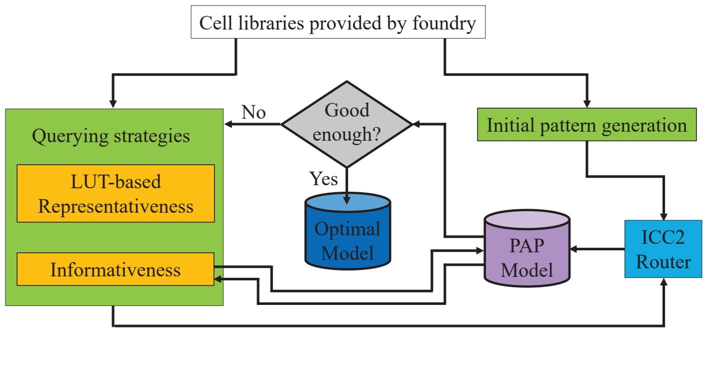
4.2 Initial Cell Combination Generation

Before starting the iterative active learning-based flow, the initial model should be ready. Since we do not have any information to smartly choose the good samples from the given cell libraries, we can only randomly compose a batch of cell combinations from the given cell libraries. The task is to train a PAP model which can predict the pin accessibility of every cell of the given cell libraries considering abutting cells. To let the initial model make full consideration of all cells, each type of library cell is regarded as the target cell and is placed at the center position of a cell combination for pin accessibility evaluation (Sec. 3.1). Then, we randomly select two library cells with random orientations without PG rail violation abutting to the left and the right side of the target cell. With the netlist construction mentioned in Sec. 3.1, we place all cell combinations onto the prepared empty designs and call ICC2 router [32] to complete the routing and DRC check for deriving the labels. As all cell combinations are labeled, we use all the cell combinations and their labels to train the initial model.

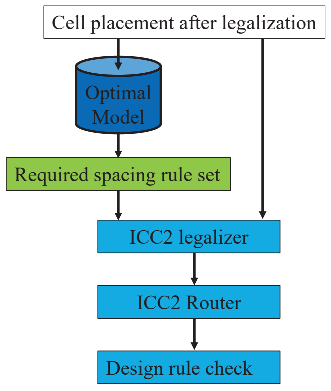
4.3 Querying Strategies

Although the initial model has considered the pin accessibility of all cells, while the model suffers from low accuracy as it only considers few abutment combinations for each cell. Based on the pre-trained initial model, the labels of more cell combinations are required to enhance the model. Since it is prohibitively expensive to query astronomical cell combinations, we propose two querying strategies to smartly query new unseen samples: (1) LUT-based representativeness evaluation, and (2) informativeness computation, which are detailed in the following.

4.3.1 LUT-based representativeness evaluation. In this subsection, we try to pick out more representative library cells with higher probabilities of DRV occurrence and determine the exact number of queries for each library cell. Before detailing the representative


Cell libraries provided by foundry
Querying strategies
LUT-based Representativeness
Informativeness
No
Good enough?
Yes
Optimal Model
Initial pattern generation
PAP Model
ICC2 Router

(a)



(b)

Figure 6: The proposed algorithm flow. (a) Active learning-based training flow. (b) Placement optimization flow.

Table 1: The notations for explaining querying strategies.

L	A set of cell libraries provided by foundries or design houses.
l_i	The i^{th} cell library in L.
C_i	A set of cells in l_i.
c_j^i	The j^{th} cell in C_i.
$num\_drv(c_j^i)$	The number of DRV occurrences of c_j^i in its labeled cell combinations.
$num\_drvc(c_j^i)$	The number of DRV-clean cell combinations of c_j^i in its labeled cell combinations.
num_lib_cells	The number of library cells in L.
CC_resource	The total resource for cell combinations which can be placed into the prepared empty blocks.

querying strategy, some notations are defined in Table 1. After the initial model is trained, we will have the labels of trained cell combinations. Thus, we can create a lookup table (LUT) to store the DRV occurrence history of each library cell, where the DRV occurrence history is used to record the numbers of cell combinations with and without DRVs associated with each library cell. With the DRV occurrence history of each lib cell, we can intuitively compute the probability of DRV occurrence for each library cell as follow:

$$P_{drv}(c_j^i) = \frac{num\_drv(c_j^i)}{num\_drv(c_j^i) + num\_drvc(c_j^i)}, \tag{1}$$

where $P_{drv}(c_j^i)$ represents the estimated probability of DRV occurrence of c_j^i. It should be emphasized again that the goal of the PAP model is to identify those cell combinations with DRV occurrence due to bad pin accessibility. Therefore, we aim at querying more cell combinations for those library cells with higher probabilities of DRV occurrence. However, for the cells whose sampled cell combinations are all DRV-clean during the initial training, we cannot filter out these cells immediately. It is because these cells still have opportunities to suffer from DRVs when abutting with other neighboring cells that are not considered in the initial model. As a consequence, we compute the representative score of each library cell by taking zero-mean from its current probability, and then the sigmoid function is applied to ensure that the score is in the range of $[0, 1]$ and will not be zero for those cells without DRV occurrence in the initial training. The scoring function is formulated as follows:

$$Score(c_j^i) = \sigma(P_{drv}(c_j^i) - \frac{\sum_{l_i \in L} \sum_{c_j^i \in C_i} P_{drv}(c_j^i)}{num\_lib\_cells}), \tag{2}$$

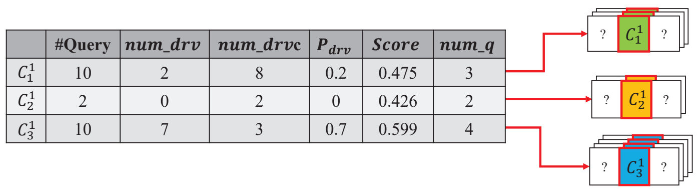
$$\sigma(x) = \frac{1}{1 + e^{-\alpha x}}, \tag{3}$$

where α is a user-defined parameter, and the larger α is, the smoother the curve of the sigmoid function will be. After the representative scores of all library cell are obtained, we can compute the required number of queries of each library cell as follows:

$$num\_q(c_j^i) = \lfloor \frac{Score(c_j^i)}{\sum_{l_i \in L} \sum_{c_j^i \in C_i} Score(c_j^i)} \times CC\_resource \rfloor. \tag{4}$$

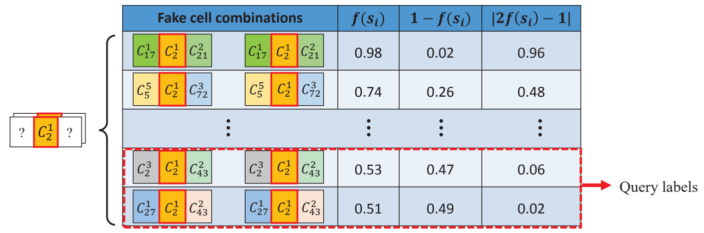
Note that the term $CC\_resource$ is a fixed number depending on the number of available memory usage. With more machines and more memory on each machine, we can create more empty blocks for ICC2 to route unlabeled cell combinations. The floor function in Equation (4) is used to prevent the insufficient resource for unlabeled cell combinations. Fig. 7 gives a simple example for explaining the LUT-based representativeness querying strategy. Note that α and $CC\_resource$ in this example are set to be 1 and 10, respectively. According to the DRV history LUT, we can easily find the number of num_drv and num_drvc of each library cell. Thus, the terms P_{drv}, $Score$, and num_q can be respectively computed by Equations (1), (2), and (4). With the computed num_q of each library cell, we can get the exact number of queries for each lib cell in each iteration, as indicated in the rightmost column in Fig. 7.

4.3.2 Informativeness. Having the number of queries of each library cell, we aim at picking out the most informative cell combinations for each library cell which can best improve the model performance. That is, we focus on those cell combinations in which the model does not have enough confidence, and thus querying the labels of these combinations is desirable and able to improve the accuracy of DRV prediction. We take c_2^1 in Fig. 7 as an example to explaining the informativeness querying strategy, as shown in Fig. 8. The required number of queries for c_2^1 has been obtained from



	#Query	*num_drv*	*num_drvc*	P_{drv}	*Score*	*num_q*
C_1^1	10	2	8	0.2	0.475	3
C_2^1	2	0	2	0	0.426	2
C_3^1	10	7	3	0.7	0.599	4

Figure 7: An example for explaining the LUT-based representativeness querying strategy.



Fake cell combinations	$f(s_i)$	$1-f(s_i)$	$\|2f(s_i)-1\|$
C_{17}^1 C_2^1 C_{21}^2 C_{17}^1 C_2^1 C_{21}^2	0.98	0.02	0.96
C_5^5 C_2^1 C_{72}^3 C_5^5 C_2^1 C_{72}^3	0.74	0.26	0.48
⋮	⋮	⋮	⋮
C_2^3 C_2^1 C_{43}^2 C_2^3 C_2^1 C_{43}^2	0.53	0.47	0.06
C_{27}^1 C_2^1 C_{43}^2 C_{27}^1 C_2^1 C_{43}^2	0.51	0.49	0.02

Figure 8: An example for explaining the informativeness querying strategy.

the LUT-based representativeness evaluation. In the informativeness computation, we first randomly create a batch of unseen cell combinations. Then, all cell combinations are fed into the model to evaluate the score of DRV prediction $f(s_i)$, where s_i denotes the i^{th} cell combination and f represents the hypothesis function for evaluating the probability of DRV occurrence [5]. As all DRV probabilities are computed, the DRV-clean probability can be easily computed by $1 - f(s_i)$. Finally, the cell combinations with the smallest differences between its DRV probabilities and DRV-clean probabilities are chosen to be the candidates for querying their labels. As shown in Fig. 8, for c_2^1, the last two cell combinations are queried due to their smallest differences between $f(s_i)$ and $1 - f(s_i)$.

4.4 Model-guided Placement Refinement

Having the optimized DRV prediction model, we adopt the model to refine placements and thus reduce the number of DRVs. Inspired by the approach proposed in reference [5], setting placement spacing rules is an effective method to guide the placer to avoid generating pin patterns with bad pin accessibility. Given a legalized placement, we first adopt the trained model to inference the DRV probabilities of all cell combinations among the given placement. Then, for each cell combination whose DRV probability is greater than the predefined threshold value, the required placement spacing between the two cells is computed by adopting the method proposed in [5]. Unlike [5] that proposes their own dynamic programming-based detailed placement algorithm considering the prepared spacing rules, we directly apply the build-in feature in the ICC2 placer to refine the placement performance, which takes a set of placement spacing rules as the input and re-legalizes a given legalized placement based on the spacing rules. The reason is that the detailed placement algorithm proposed in [5] only considers the cell overlap constraint, while it fails to consider many other important design rules in modern processes, such as coloring rules and multi-domain-related rules of power/ground nets, leading to impractical placement results. ICC2 placer is mature in DRV-aware legalization and can also tackle additional spacing rules. Therefore, we derive the required spacing rules for a given legalized placement and then re-legalize the placement based on the spacing rules to avoid generating undesirable cell abutments with bad pin accessibility.

5 EXPERIMENTAL RESULTS

The EDA tool ICC2 is used to support the placement and routing requirements [32]. The industrial cell library set and benchmarks are used for experiments, which cannot be revealed due to the proprietary limitation. The active learning flow is implemented by C++. The library-based model training flow is written in Python with Keras neural networks API and then transferred into C++ interface with Tensorflow C++ API [33] to speedup the evaluation time. The model-guided placement refinement is also implemented by C++. Experiments were conducted on a Linux machine with 2.6 GHz CPU and 264 GB memory. The testcases for experiments are modified by real industrial designs referencing to the same cell library set. Experimental results are divided into three parts: (1) the performance comparison between the design-specific models proposed by [5] and the library-based model proposed in this paper, (2) the model-guided placement comparison between the optimizations based on the design-specific models and the library-based model, and (3) runtime analyses between the design-specific flow and the library-based flow.

5.1 Model Performance Comparison

Table 2: Comparison of five quantitative metrics.

	designA	*designB*	ours
Accuracy	92.12%	99.90%	79.26
True positive rate	79.97%	100.00%	67.57
Positive predictive value	80.85%	99.18%	61.78
False positive rate	5.06%	0.00%	13.05
F-measure	80.41	99.48	64.54

To compare the performance of the proposed library-based model with design-specific models, we compute five quantitative metrics for both models, which are accuracy, true positive rate, positive predictive value, false positive rate, and F-measure. Accuracy gives the probability that the model will produce a correct prediction. True positive rate is used to measure the proportion of actual cell combinations with DRVs which are correctly identified by a model. Positive predictive value is used to measure the proportion of predicted cell combinations with DRVs that are really DRV cell combinations. False positive rate measures the proportion of predicted cell combinations with DRVs that are actually DRV-free cell combinations. F-measure is the harmonic average of the true positive rate and positive predictive value, which is larger if both the two values are maximized and the difference between the two values is minimized. A binary classifier has better performance if its false positive rate is lower and the other four metrics are higher.

The computed five quantitative metrics are summarized in Table 2. The *designA*-specific model and the *designB*-specific model are trained and tested by using their own training and testing data sets (derived from routed designs), and our library-based model is tested by both the two testing sets. It can be observed from Table 2 that all the quantitative metrics of design-specific models outperform the library-based model. However, the good prediction performances of design-specific models do not necessarily result in better placement refinement results, which will be shown in the next experiment. In addition, we found that the design-specific training approaches

Table 3: Comparisons between $designA$ default placement, design-specific model trained by $designA$, our library-based model, and design-specific model trained by $designB$.

	Default $designA$		$DesignA$-specific model			Our library-based model			$DesignB$-specific model		
	#DRVs	#M2 sh	#DRVs	#M2 sh	Avg dis.	#DRVs	#M2 sh	Avg dis.	#DRVs	#M2 sh	Avg dis.
P^1_0	7007	409	684	58	0.02	195	18	0.04	1347	27	0.04
P^1_1	6313	404	513	43	0.02	136	11	0.04	1288	29	0.04
P^1_2	6246	343	431	33	0.02	188	8	0.04	853	17	0.04
P^1_3	6138	359	459	36	0.02	237	26	0.04	109	5	0.04
P^1_4	7306	479	531	42	0.02	148	12	0.04	473	15	0.04
P^1_5	6138	362	699	66	0.02	172	14	0.04	232	3	0.04
P^1_6	6997	410	473	36	0.02	116	9	0.04	307	8	0.04
P^1_7	6314	399	536	45	0.02	170	14	0.04	2352	70	0.04
Avg	6557	395	536	45	0.02	170	14	0.04	870	22	0.04
Comp.	1.00	1.00	0.08	0.11	1.00	0.03	0.035	2.00	0.36	0.17	2.00

Table 4: Comparisons between $designB$ default placement, design-specific model proposed by [5], and our library-based model.

	Default $designB$		$DesignB$-specific model			Our library-based model			$DesignA$-specific model		
	#DRVs	#M2 sh	#DRVs	#M2 sh	Avg dis.	#DRVs	#M2 sh	Avg dis.	#DRVs	#M2 sh	Avg dis.
P^2_0	2348	126	727	15	0.14	763	19	0.06	NA	NA	NA
P^2_1	1782	101	987	31	0.14	223	6	0.06	NA	NA	NA
P^2_2	3937	157	1893	48	0.13	468	10	0.06	NA	NA	NA
P^2_3	1646	116	656	9	0.14	175	5	0.07	NA	NA	NA
P^2_4	1777	111	1282	32	0.14	575	14	0.06	NA	NA	NA
P^2_5	3777	174	926	27	0.13	677	12	0.07	NA	NA	NA
P^2_6	2055	128	481	10	0.13	1991	54	0.07	NA	NA	NA
P^2_7	2262	130	182	2	0.13	893	17	0.07	NA	NA	NA
Avg	2448	130	892	22	0.135	721	17	0.065	NA	NA	NA
Comp.	1.00	1.00	0.36	0.17	1.00	0.29	0.13	0.48	NA	NA	NA

may suffer from the overfitting issue, especially when training data and testing data contain similar DRV patterns.

5.2 Model-Guided Placement Comparison

To compare the performances of DRV reduction by using the two models, we further generate eight different legalized placements for each of the two industrial designs, optimize the placements by applying the model-guided placement refinement, and compare the numbers of DRVs in the routing results. Note that we apply the same placement refinement approach to both the design-specific model and the library-based model for fair comparison. The experimental results are summarized in Tables 3 and 4. The numbers of total DRVs and M2 shorts are respectively denoted as "#DRVs" and "#M2 sh". "Avg dis." denotes the average cell displacement in sites. The eight placement testcases for each of the two different designs are marked as P^1_0–P^1_7 and P^2_0–P^2_7, respectively. The first, second, and third block of the tables separately show the statistics of the default design, the refined placements by the design-specific model [5], and the refined placement by the proposed library-based model. Compared to the default placements of *designA*, the placement refinements done by [5] can respectively reduce 92% total DRVs and 89% M2 shorts on average, while our model can reduce 97% total DRVs and 96.5% M2 shorts on average. In addition, compared to the default placements of *designB*, the placement refinements done by [5] can respectively reduce 64% total DRVs and 83% M2 shorts on average, while our model can further reduce 71% total DRVs and 87% M2 shorts on average. The total wirelengths of all routing results are not explicitly shown because placement refinement does not lead to wirelength degradation on average. In fact, for about half of the cases, the total wirelengths are even improved due to better routability.

It is worth to note that although Table 2 shows that the performance of the design-specific training models is superior than our library-based training model, the numbers of both total DRVs and M2 shorts of the designs optimized by our model are fewer than those optimized by the design-specific models. This is because the re-legalized designs after the model-guided placement refinement may generate new cell combinations unseen by the design-specific training models. If some important data prone to generating DRV are not produced by routed designs, design-specific models will be incompletely trained. In contrast, the library-based model is more general since the active learning-based training flow is able to query more representative and informative data from the given cell library set.

To show the generality of the proposed library-based model, we apply the design-specific model trained by *designA* to optimize *designB*, and vice versa. In our testcases, *designB* uses five different standard cell libraries and *designA* references to two of the five standard cell libraries. The fourth (rightmost) block of Table 3 shows the experimental results by using the model trained by *designB* to refine the placements of *designA*. Since the referenced cell libraries of *designA* are included in the referenced cell libraries of *designB*, the model trained by *designB* still greatly reduces the number of DRVs after placement refinement on *designA*, while the reduction rates are much lower than those of the other two models. The fourth block of Table 4 shows the experimental results by using the model trained by *designA* to refine the placements of *designB*. In contrast, the model trained by *designA* is not trained by some cell combinations which are included in the referenced cell libraries of *designB*. This results in a numerous number of placement spacing rules incorrectly generated by the incomplete *designA*-specific model, which causes all the placements of *designB* fail to be legalized.

5.3 Runtime Analysis

To train a design-specific model, several routed designs should be prepared in advance. In our experiments, we randomly generate 30 different routed designs to collect enough numbers of training data. We spend 28 hours and 68 hours on average for routing a *designA* and a *designB*, respectively. By running all designs in parallel, the least times for preparing 30 different routed *designA*s and routed *designB*s can be regarded as 28 hours and 68 hours. The training times of design-specific models for *designA* and *designB* are 216 seconds and 488 seconds, respectively. Therefore, the overall runtime is dominated by training data preparation in the design-specific training flow. As for the library-based training, the model is trained by the proposed active learning-based flow, which iteratively queries representative and informative data for incremental training until the model is good enough. In our experiments, the stopping criteria is set and the flow will be terminated if the true positive rate cannot be improved for further three training iterations. The model of the 13-th iteration is chosen since the true positive rates of the 14-th, 15-th, and 16-th iterations are not better than that of the 13-th iteration. The overall runtime of the library-based training flow with the 16 iterations is about 24 hours, which is faster than the design-specific training flow for both designs. Furthermore, the training for the library-based model is a one-time process and is applicable to all designs using the same reference library set, while design-specific models have to be trained for multiple times for designs using different reference libraries or different cells.

6 CONCLUSIONS

In this paper, We have proposed the first work of training a library-based pin accessibility prediction model in the cell library level. By given a set of cell libraries, the proposed active learning flow can automatically fake a batch of critical cell combinations via representativeness and informativeness querying strategies for each iteration. The experimental results show that the proposed library-based model outperforms design-specific models on both DRV reduction and the overall runtime cost including training data preparation and training procedure. The generality of the proposed library-based model is also proved by showing that the design-specific models cannot be directly applied to predict a design with different reference cell libraries.

REFERENCES

[1] W. T. J. Chan, P.-H. Ho, A. B. Kahng, and P. Saxena, "Routability Optimization for Industrial Designs at Sub-14nm Process Nodes Using Machine Learning," In *Proc. International Symposium on Physical Design (ISPD),* 2017.
[2] Q. Zhou, X. Wang, Z. Qi, Z. Chen, Q. Zhou, and Y. Cai. "An accurate detailed routing routability prediction model in placement." In *Proc. Asia Symposium on Quality Electronic Design (ASQED),* 2015.
[3] A. F. Tabrizi, N. K. Darav, S. Xu, L. Rakai, I. Bustany, A. Kennings and, L. Behjat. "A Machine Learning Framework to Identify Detailed Routing Short Violations from a Placed Netlist." *Proc. Design Automation Conference (DAC),* 2018.
[4] Z. Qi, Y. Cai and Q. Zhou. "Accurate Prediction of Detailed Routing Congestion using Supervised Data Learning." In *Proc. International Conference on Computer Design (ICCD),* 2014.
同一作
[5] T. C. Yu, S. Y. Fang, H. S. Chiu, K. S. Hu, P. H. Y. Tai, C. C. F. Shen, and H. Sheng. "Pin Accessibility Prediction and Optimization with Deep Learning-based Pin Pattern Recognition." *Proc. Design Automation Conference (DAC),* 2019.
[6] A. F. Tabrizi, N. K. Darav, L. Rakai, A. Kennings and L. Behjat. "Detailed Routing Violation Prediction During Placement Using Machine Learning." *Proc. VLSI Design, Automation and Test (VLSI-DAT),* 2017.
[7] W. T. J. Chan, Y. Du, A. B. Kahng, S. Nath and K. Samadi. "BEOL stack-aware routability prediction from placement using data mining techniques" In *Proc. IEEE International Conference on Computer Design (ICCD),* 2016.
[8] Z. Xie, Y. H. Huang, G. C. Fang, H. Ren, S. Y. Fang, Y. Chen and J. Hu. "RouteNet: Routability Prediction for Mixed-Size Designs Using Convolutional Neural Network." In *Proc. International Conference on Computer-Aided Design (ICCAD),* 2018.
[9] X. Xu, B. Yu, J. R. Gao, C. L. Hsu and D. Z. Pan. "PARR: Pin Access Planning and Regular Routing for Self-Aligned Double Patterning." *ACM Transactions on Design Automation of Electronic Systems (TODAES),* vol. 21, no. 3, article 42, 2016.
[10] X. Xu, Y. Lin, V. Livramento and D. Z. Pan. "Concurrent Pin Access Optimization for Unidirectional Routing." *Proc. Design Automation Conference (DAC),* 2017.
[11] J. Seo, J. Jung, S. Kim and Y. Shin. "Pin Accessibility-Driven Cell Layout Redesign and Placement Optimization." *Proc. Design Automation Conference (DAC),* 2017.
[12] W. Ye, B. Yu, D. Z. Pan, Y. C. Ban and L. Liebmann. "Standard Cell Layout Regularity and Pin Access Optimization Considering Middle-of-Line." *Proc. Great Lakes Symposium on VLSI (GLSVLSI),* 2015.
[13] Y. Ding, C. Chu and W. K. Mak. "Pin Accessibility-Driven Detailed Placement Refinement." In *Proc. International Symposium on Physical Design (ISPD),* 2017.
[14] M. M. Ozdal. "Detailed-Routing Algorithms for Dense Pin Clusters in Integrated Circuits." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD),* vol. 28, no. 3, pp. 340–349, 2009.
[15] X. Xu, B. Cline, G. Yeric, B. Yu and D. Z. Pan. "Self-Aligned Double Patterning Aware Pin Access and Standard Cell Layout Co-Optimization." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD),* vol. 34, no. 5, pp. 699–712, 2015.
[16] D. Vasisht, A. Damianou, M. Varma, and A. Kapoor. "Active learning for sparse Bayesian multilabel classification." In *Proc. ACM SIGKDD.* New York, NY, USA, 2014, pp. 472–481.
[17] P. Melville and R. J. Mooney. "Diverse ensembles for active learning." In *Proc. 21th Int. Conf. Mach. Learn.* Banff, AB, Canada, 2004, p. 74.
[18] I. Guyon, G. Cawley, V. Lemaire, and G. Dror. "Results of the active learning challenge." *J. Mach. Learn. Res.* vol. 16, pp. 19–45, 2011.
[19] M. Wang and X.-S. Hua. "Active learning in multimedia annotation and retrieval: A survey." *ACM Trans. Intell. Syst. Technol.* vol. 2, no. 2, pp. 1–21, 2011.
[20] Y. Chen and A. Krause. "Near-optimal batch mode active learning and adaptive submodular optimization." *J. Mach. Learn. Res.* vol. 28, no. 1, pp. 160–168, 2013.
[21] S. Tong and D. Koller. "Support vector machine active learning with applications to text classification." *J. Mach. Learn. Res.* vol. 2, pp. 45–66, 2001.
[22] J. Zhu and M. Ma. "Uncertainty-based active learning with instability estimation for text classification." *ACM Trans. Speech Lang. Process.* vol. 8, no. 4, pp. 1–21, 2012.
[23] M. Park and J. W. Pillow. "Bayesian active learning with localized priors for fast receptive field characterization." In *Proc. NIPS.* Nevada City, CA, USA, 2012, pp. 2357–2365.
[24] W. Luo, A. Schwing, and R. Urtasun. "Latent structured active learning." In *Proc. NIPS.* Lake Tahoe, CA, USA, 2013, pp. 728–736.
[25] N. V. Cuong, W. S. Lee, N. Ye, K. M. A. Chai, and H. L. Chieu. "Active learning for probabilistic hypotheses using the maximum Gibbs error criterion." In *Proc. NIPS.* Lake Tahoe, CA, USA, 2013, pp. 1457–1465.
[26] B. Du, Z. Wang, L. Zhang, L. Zhang, W. Liu, J. Shen, and D. Tao. "Exploring representativeness and informativeness for active learning." *IEEE Trans. Cybern.* 47(1), 14–26, 2017.
[27] X. Li, D. Kuang, and C. X. Ling. "Active learning for hierarchical text classification." In *Advances in Knowledge Discovery and Data Mining.* Berlin, Germany: Springer, 2012, pp. 14–25.
[28] I. Dino, B. Albert, Z. Indre, and P. Bernhard. "Clustering based active learning for evolving data streams." In *Discovery Science* Berlin, Germany: Springer, 2013, pp. 79–93.
[29] R. Chattopadhyay et al. "Batch mode active sampling based on marginal probability distribution matching." In *Proc. ACM SIGKDD.* China, 2012, pp. 741–749.
[30] Y. Fu, B. Li, X. Zhu, and C. Zhang. "Active learning without knowing individual instance labels." *IEEE Trans. Knowl. Data Eng.* vol. 26, no. 4, pp. 808–822, Apr. 2014.
[31] T. Reitmaier, A. Calma, and B. Sick. "Transductive active learningâĂŤA new semi-supervised learning approach based on iteratively refined generative models to capture structure in data." *Inf. Sci.* vol. 293, pp. 275–298, Feb. 2015.
[32] Synopsys ICC2 user guide. https://www.synopsys.com
[33] Tensorflow C++ Reference. https://www.tensorflow.org/api_docs/cc/