VLSI Layout Hotspot Detection Based on Discriminative Feature Extraction

Hang Zhang, Haoyu Yang, Bei Yu and Evangeline F. Y. Young

Department of Computer Science and Engineering,
The Chinese University of Hong Kong, NT, Hong Kong
{hzhang,hyyang,byu,fyyoung}@cse.cuhk.edu.hk

***Abstract*—Feature extraction is a key stage in machine learning based VLSI layout hotspot detection flow. Conventional machine learning based methods apply various feature extraction techniques to approximate an original layout structure at nanometer level. However, some important layout pattern information is missed during the approximation process, resulting in performance degradation. In this paper, we present a comprehensive study on layout feature extraction and propose a new method that can preserve discriminative layout pattern information to improve the detection performance in terms of accuracy and extra. Experiments were conducted on an industrial benchmark and ICCAD benchmark suite to study the effectiveness of our proposed methods.**

I. Introduction

As the gap between modern integrated circuit feature size and optical lithography wavelength increases [1], it is essential to ensure the printability of the circuit layout design. Various printability-aware methods are developed, such as design rule check (DRC), optical proximity correction (OPC) and sub-resolution assist feature (SRAF), to reduce the performance degradation or the yield loss during circuit manufacturing. However, there exist some layout patterns that are sensitive to the lithography process and will produce undesirable printing. These layout patterns are called VLSI layout hotspots and need to be detected in the physical verification phase.

Recently, the hotspot detection problem has earned more and more attentions [2]. Pattern matching and machine learning based methods have been widely studied and applied in this problem [3]–[6]. Pattern matching is a computationally-efficient method for hotspot detection problem in the literature. However, this method needs a pre-defined training layout pattern set and unseen hotspots cannot be detected effectively. Machine learning is famous for its excellent generalization ability in detecting unseen layout patterns that have similar characteristics with the training data, and has been widely utilized. Nevertheless, it is imperative to design appropriate layout feature to capture discriminative characteristics between hotspot and non-hotspot layout patterns during feature extraction.

Various feature extraction approaches [4]–[8] have been explored to improve the performance of machine learning based methods. To represent layout patterns, the work [7] carefully designs a fragment based feature, which may be too complicated for some design cases. In addition, the work [6] uses some critical topology features such as the distance between two adjacent block tiles and the diagonal relations between two convex corners of block tiles. This feature is simple but may ignore some important information of the layout pattern. In order to simplify the layout feature while preserve the key information, [4], [5] apply density based feature extraction method. Although density based feature has achieved the highest accuracy in hotspot detection problem so far, there still exist many hotspots that cannot be detected.

In this paper, we propose a novel feature extraction method that has higher accuracy and lower number of extras. Our contributions are listed as follows:

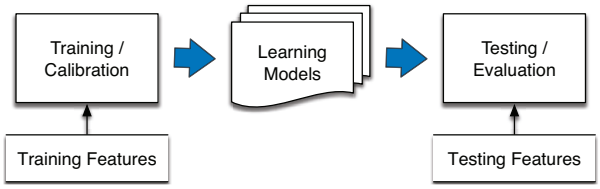


Fig. 1: Overall hotspot detection flow.

- We propose a layout feature called *local grid density differential (LGDD)*. Unlike conventional density based method, LGDD measures the density of a specific area in a grid.
- We further analyze the performance of LGDD when varying the spacing distance between adjacent grids.

The rest of the paper is organized as follows. Section II describes our hotspot detection framework and defines some useful terminologies to evaluate our methods. section III proposes local grid density differential (LGDD) method and introduces another parameter, stride. section IV presents a machine learning model used in our framework. Section V gives the experimental results, followed by conclusion in Section VI.

II. Preliminaries

This section describes our machine learning based hotspot detection flow, evaluation metrics, and problem formulation.

A. Hotspot Detection Flow

Our hotspot detection flow consists of two phases, "Training phase" and "Testing phase", as shown in Fig. 1. The objective of the training phase is to build a learning model. In this phase, a set of hotspot and non-hotspot layout patterns and their corresponding labels are given and a machine learning model is calibrated after layout feature extraction. The objective of the testing phase is to detect hotspots. In this phase, a set of testing layout patterns different from the training layout patterns are used as the input. Feature extraction is then performed on these layout patterns, after which the machine learning model will predict labels (identified as hotspot or non-hotspot) for these testing patterns.

B. Evaluation Metrics and Problem Formulation

In order to evaluate the performance of our proposed methods, we define several useful terms as follows:

Definition 1 (Accuracy)**.** *The rate of correctly predicted hotspots among the set of actual hotspots.*

Definition 2 (Extra)**.** *The number of falsely detected hotspots.*

Based on the above two terminology definitions, we give the problem formulation of hotspot detection as follows:

Problem 1 (Hotspot Detection)**.** *Given a set of training layout patterns including hotspots and non-hotspots, a machine learning model is trained to identify hotspots in the testing layout. The objective is to maximize the accuracy and minimize the number of extras.*

III. Layout Feature Extraction

As mentioned in the introduction, layout feature plays an important role in the hotspot detection problem, where layout pattern information is represented by a set of feature vectors. Although several layout feature extraction methods [4]–[8] have been proposed, how to represent layout pattern accurately remains a problem. In this section, we propose a novel layout feature extraction method called local grid density differential (LGDD) to improve the performance of conventional feature extraction methods. Besides, we analyze the impact of different spacing distances on the detection performance.

A. Conventional Density Based Feature

Circuit layout patterns are well known to be very complicated and are very difficult to be handled because it can only be represented in a high-dimensional space. For example, assuming that the 1.2 μm^2 area in the core of an ICCAD benchmark is set on a $1nm$ grid, we have to use a 1200×1200 vector to represent the pattern. However, these high-dimensional vectors pose a threat to the training phase, simply because it is not easy to get enough instances for model training. One way to resolve this problem is to apply efficient feature representation or extraction methods for these layout patterns. As mentioned in Section I, density-based feature is one of the most efficient methods and has much potential to be improved. We will discuss the conventional density based feature and derive its improved version.

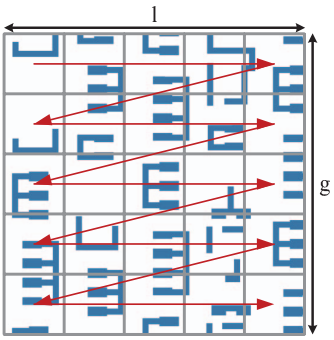


Fig. 2: Density based layout feature representation. Feature vector is represented as: $X = \{a_{11}, a_{12}, ..., a_{54}, a_{55}\}$.

The conventional density based layout feature representation is shown in Fig. 2. Feature vector X is consisted of several real values represented as a_{ij} denoting the density value of the grid in the i^{th} row and the j^{th} column. This feature has two important parameters which are the size of the layout pattern l and the number of grids g in a row or column. The total length of a density based feature vector is then g^2. For example, in Fig 2, $g = 5$ and the length of the feature vector is $g^2 = 25$.

Density based feature has simple representation of the layout pattern, namely the feature is in low-dimensional space and may be able to avoid the over fitting issue. Although it has achieved good performance in the hotspot detection problem [4], it still ignores some important information during the feature extraction. In the following two subsections, we will present a novel extension of conventional density based feature extraction methods from two perspectives: 1) how to extract discriminative information from one grid; 2) how to sample density grids.

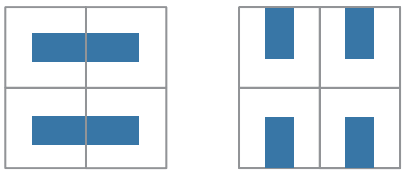


Fig. 3: Example of information loss in conventional density based method.

B. Local Grid Density Differential

Conventional density based method extracts geometric information of the layout pattern by locally averaging the density value of a grid. Although it achieves reasonable good performance, it may lose some important information during the feature extraction. As shown in Fig. 3, if we extract features from Fig. 3(a) and Fig. 3(b), their representation vectors will be the same. Therefore, it is imperative to extract more discriminative information from each grid. In this paper, we propose a method named local grid density differential (LGDD) to extract discriminative feature representation from the layout.

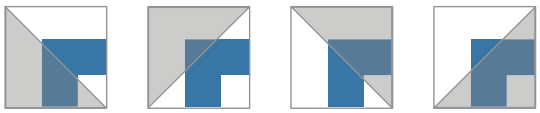


Fig. 4: Examples of specific areas for LGDD.

Different from conventional density based feature which locally averages the density value of a gird, we locally average the density value of a specific area in a grid. Since the polygon in a layout is consisted of several rectangles, intuitively, we set the shape of the sampling area as shown in Fig. 4. More specifically, we set the sampling area as a triangle locating at different corners of a grid. The goal of having different sampling area is to extract more discriminative information for layout pattern representation. As we can see from Fig. 4, there are two triangles in a grid, for each case a shadowed one and a blank one. When we extract features from these grids, we only average the density values from the shadowed ones. We then concatenate all the grid values to form a feature vector in the same way as conventional density based methods.

From Fig. 3 we can see that density based methods may extract the same feature vectors from different patterns (Fig. 3(a) and Fig. 3(b)). Although the LGDD method may have the same problem, we can resolve this problem by concatenating values from different sampling areas (e.g. 4 kinds of sampling area in Fig. 4). However, this may also result in $4\times$ longer feature vector dimension and cause potential problem of over fitting. Fortunately, Adaboost [9] with Decision Tree [10] classifier can perform feature selection efficiently and possibly avoid over fitting. we can see from the experimental results that the performance of LGDD outperforms conventional density based methods.

C. Stride between Density Grids

In this subsection, we will introduce a new parameter for conventional density based methods called "*stride*". The stride (denoted as

s) used in our method is the spacing between adjacent grids (vertical and horizontal) where density value of the grid will be extracted (see Fig. 5). Conventional density based method can be seen as a special case in our method, where $s = w$ (w is the width of a single grid, as shown in Fig. 5).

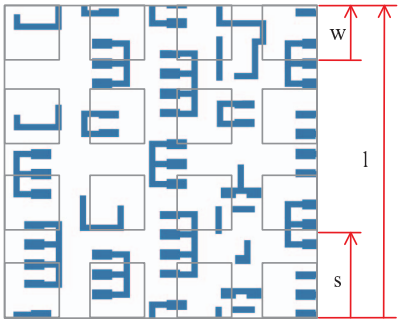


Fig. 5: Concept of strides between adjacent grids.

Different from conventional density based method where the stride between adjacent grids is the same as their width, our method can extract pattern information from more grids by having a smaller s.

IV. Machine Learning Model

In this section, we will describe the machine learning model used in solving hotspot detection problem. We focus on one of the most efficient machine learning methods, which is Adaboost [9] with Decision Tree [10]. Before describing the machine learning model, we give some generic variables that will be used in the model. We use $(\mathbf{x}, y)$ to represent a layout pattern, where x is the extracted feature vector of the layout pattern and y is the label. Given a set of training vectors $\mathbf{X} = (\mathbf{x_1}, ..., \mathbf{x_n})$ and the corresponding labels $\mathbf{Y} = (y_1, ..., y_n)$, we can construct our classifiers.

A. Adaboost

Adaboost is an ensemble learning algorithm that can be used in conjunction with many other weak learning algorithms (weak learner) to boost their performance. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing (i.e., their error rate is smaller than 0.5 for binary classification), the final model can be proven to converge to a strong learner. The algorithm flow of Adaboost is shown in Algorithm 1.

Algorithm 1 Adaboost classifier

Require: $\mathbf{X} = (\mathbf{x_1}, ..., \mathbf{x_n})$, $\mathbf{Y} = (y_1, ..., y_n)$, T.
1: **for** $i \leftarrow 1$ to n **do**:
2: $\quad D_1(i) = \frac{1}{n}$;
3: **end for**
4: **for** $t \leftarrow 1$ to T **do**:
5: $\quad h_t \leftarrow$ base classifier with small error ϵ_t;
6: $\quad \epsilon_t \leftarrow P(h_t(\mathbf{x_i}) \neq y_i) = \sum_{i=1}^{n} D_t(i) I(h_t(\mathbf{x_i}) \neq y_i)$;
7: $\quad \alpha_t \leftarrow \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$;
8: $\quad Z_t \leftarrow 2[\epsilon_t(1-\epsilon_t)]^{\frac{1}{2}}$;
9: $\quad$ **for** $i \leftarrow 1$ to n **do**:
10: $\quad\quad D_{t+1}(i) \leftarrow \frac{D_t(i) exp(-\alpha_t y_t h_t(\mathbf{x_i}))}{Z_t}$;
11: $\quad$ **end for**
12: **end for**
13: $f \leftarrow \text{sign}(\sum_{t=1}^{T} \alpha_t h_t)$;
14: **return** f

In Algorithm 1, T is the number of weak classifiers, D_t is the weights of the samples in the t^{th} iteration, n is the number of training samples, e_t is the error rate of the t^{th} iteration, α_t is the weight of the t^{th} weak classifer, h_t is the t^{th} classifier. The final classifier can be cascaded as: $f(\mathbf{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}))$.

B. Decision Tree

In this paper, we use Decision Tree as our weak classifier [10], which is a non-parametric supervised learning method in classification. Here we give the mathematical formulation of the Decision Tree classifier. Let the data at node m be represented by Q. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m, partition the data into $Q_{left}(\theta)$ and $Q_{right}(\theta)$ subsets

$$Q_{left}(\theta) = (\mathbf{X}, \mathbf{Y}) | \mathbf{X^j} \leq t_m, \tag{1}$$

$$Q_{right}(\theta) = Q \setminus Q_{left}(\theta). \tag{2}$$

The impurity at m is computed using an impurity function H defined as below. The impurity is formed as below:

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta)), \tag{3}$$

where n_{left} is the number of data samples in the left partition, n_{right} is the number of data samples in the right partition and N_m is the number of all samples in the node m. Then we select the parameters that minimizes the impurity:

$$\theta^* = \text{argmin}_\theta \, G(Q, \theta). \tag{4}$$

Recurse for subsets $Q_{left}(\theta^*)$ and $Q_{right}(\theta^*)$ until the maximum allowable depth is reached, or $N_m < N_{min}$ or $N_m = 1$ (N_{min} is the minimum number of samples in a node). The target is a classification outcome taking on values $0, 1, ..., K-1$, (K is the number of classes) for node m, representing a region in the feature space with N_m observations. Let

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in X} I(y_i = k) \tag{5}$$

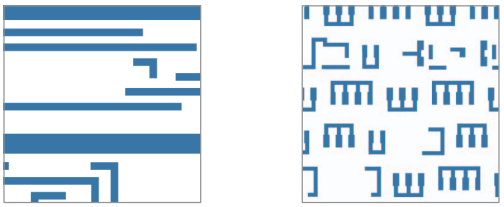
be the proportion of class k observations in node m, Gini index is used to construct the impurity function H:

$$H(\mathbf{X_m}) = \sum_k p_{mk}(1 - p_{mk}), \tag{6}$$

where $\mathbf{X_m}$ stands for the data samples at node m.

V. Experiments and Analysis

The proposed methods are implemented in Python on a computer with four-core 3.7GHz CPU and 16GB memory. The ICCAD 2012 contest benchmark suite is applied to verify our methods. This benchmark suite contains five test cases, and for simplicity, the statistics of this benchmark can be found in [4]. Besides the ICCAD benchmark, we also apply an industrial benchmark, where 885 hotspot patterns and 602 non-hotspot patterns are available and the resolution of each pattern is $2000nm \times 2000nm$. Unlike ICCAD benchmark where the training and testing sets are pre-defined, we randomly choose half of the patterns as our training set and another half as our testing set in the industrial benchmark. It should be noted that the layout pattern in the industrial benchmark is much more complicated than that in the ICCAD benchmark, as shown in Fig. 6. We first conduct experiments on the impact of the LGDD feature and the stride parameter on the performance of the industrial benchmark. Then we will further analyze the results of our methods on both ICCAD and industrial benchmarks.

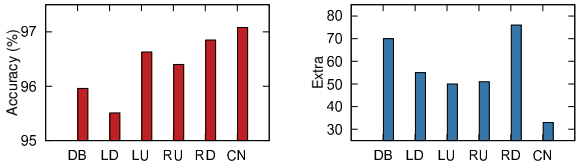


(a) (b)

Fig. 6: Layout pattern examples. (a) ICCAD benchmark. (b) Industrial benchmark.

A. Effect of LPDD

We compare our LGDD feature with conventional density based feature by showing the performance of using both single sampling area and four concatenated sampling areas. We denote the sampling areas in Fig. 4(a), Fig. 4(b), Fig. 4(c) and Fig. 4(d) as **"LD"**, **"LU"**, **"RU"** and **"RD"**, respectively. In addition, **"DB"** and **"CN"** denote the conventional density based method and the concatenated LGDD.



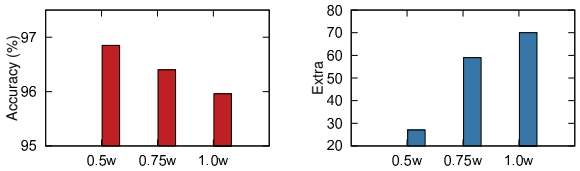
(a) (b)

Fig. 7: Performance comparison between conventional density based feature and LGDD. (a) Accuracy. (b) Extra.

We can see from Fig. 7 that our method using concatenated LGDD CN outperforms all the other methods. Compared to conventional density feature, our method not only improves the detection accuracy by 1.12%, but also reduces 37 extras. In addition, other features like LU and RU also performs better than conventional density based feature in terms of both accuracy and extra.

B. Effect of Stride

By setting the stride smaller, we can get a longer feature vector. With a longer feature vector, we are more likely to obtain the discriminative feature after the feature selection procedure in the Decision Tree classifier. In this experiment, we fixed the sampling area of LGDD with DB and vary the stride over $1w$, $0.75w$ and $0.5w$. The results are described in Fig. 8, which shows a clear downward trend in performance of accuracy and a clear upward trend in number of extras with increasing strides as we expected.



(a) (b)

Fig. 8: Performance comparison among different strides (1.0w refers to conventional density based method). (a) Accuracy. (b) Extra.

C. Final classification result

We have shown the performance of applying LGDD and stride to conventional density based feature separately. By adjusting these parameters simultaneously, we can further improve the performance of both accuracy and extra in both ICCAD benchmark and the industrial benchmark. As we can see from Table. I, comparing with conventional density based method, our method can on average increase detection accuracy from 94.63% to 95.38% and reduce the extra number from 12.3 to 5.6.

TABLE I: Comparison with conventional density based method

	Density Based		Our Proposed	
	Extra#	Accuracy	Extra#	Accuracy
ICCAD-1	**0**	99.50%	2	**100.00%**
ICCAD-2	**0**	97.18%	**0**	**98.80%**
ICCAD-3	**0**	97.50%	1	**97.78%**
ICCAD-4	4	82.49%	5	**83.05%**
ICCAD-5	**0**	**95.12%**	**0**	**95.12%**
Industry	70	95.96%	**26**	**97.53%**
Average	12.3	94.63%	**5.6**	**95.38%**

VI. Conclusion

In this work, we have conducted extensive experiments using improved density based feature in solving hotspot detection problem. Local grid density differential (LGDD) approach was proposed to extract feature vectors that contain more discriminative information. We then further analysis the impact of different strides on the performance of hotspot detection. Equipped with LGDD and flexible parameter, stride, we are able to further improve the performance of conventional density based method.

References

[1] D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 32, no. 10, pp. 1453–1472, 2013.

[2] A. J. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2012, pp. 349–350.

[3] D. Ding, B. Yu, J. Ghosh, and D. Z. Pan, "EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 2012, pp. 263–270.

[4] T. Matsunawa, J.-R. Gao, B. Yu, and D. Z. Pan, "A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction," in *Proceedings of SPIE*, vol. 9427, 2015.

[5] W.-Y. Wen, J.-C. Li, S.-Y. Lin, J.-Y. Chen, and S.-C. Chang, "A fuzzy-matching model with grid reduction for lithography hotspot detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 33, no. 11, pp. 1671–1680, 2014.

[6] 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," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 34, no. 3, pp. 460–470, 2015.

[7] B. Yu, J.-R. Gao, D. Ding, X. Zeng, and D. Z. Pan, "Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering," *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)*, vol. 14, no. 1, p. 011003, 2015.

[8] T. Matsunawa, S. Maeda, H. Ichikawa, S. Nojima, S. Tanaka, S. Mimotogi, H. Nosato, H. Sakanashi, M. Murakawa, and E. Takahashi, "Generator of predictive verification pattern using vision system based on higher-order local autocorrelation," in *Proceedings of SPIE*, vol. 8326, 2012.

[9] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *Annals of Statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[10] T. Hastie, R. Tibshirani, J. Friedman, and J. Franklin, "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer*, vol. 27, no. 2, pp. 83–85, 2005.