Research Article

Hotspot Detection with Machine Learning Based on Pixel-Based Feature Extraction

Zhifeng Lin,1 Zhenghua Gu,2 Zhipeng Huang,1 Xiqiong Bai,1 Lixuan Luo,1 and Geng Lin 3

1Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou, China
2State Key Laboratory of ASIC & System, Fudan University, Shanghai, China
3College of Mathematics and Data Science, Minjiang University, Fuzhou, China

Correspondence should be addressed to Geng Lin; lingeng413@163.com

Received 15 July 2021; Revised 27 April 2022; Accepted 15 July 2022; Published 23 August 2022

Academic Editor: Sikandar Ali

Copyright © 2022 Zhifeng Lin et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The complexity of physical verification increases rapidly with fast shrinking technology nodes. Considering only design rule checking (DRC) constraints or lithography models cannot capture the side physical effects in the fabrication process well. Thus, it is desirable to consider a more general physical verification problem with various types of hotspots. In this paper, we apply machine learning which is based on pixel-based feature extraction to deal with the generalized hotspot detection problem. First, a two-dimensional discrete Fourier transformation-based pixel extraction method is proposed to alleviate the shifting effect and produce stable hotspot features. Then, a pattern-based layout scanning approach is developed to enhance the program efficiency while preserving good detection accuracy. Finally, we design two false alarm reduction strategies to effectively reduce the number of detected nonhotspots and further improve the accuracy of hotspot position. Experimental results based on the industrial benchmarks show that our algorithm outperforms three competitive works in terms of accuracy, false alarm rate, efficiency, and time.

1. Introduction

With the continuous shrinking of process nodes and the increase of design complexity, how to manufacture a design correctly with minimal yield loss becomes a great challenge [1]. In a 28 nm node full chip layout, there could be billions of patterns and structures which need to be verified in the verification process, and consequently long processing time is often required [2]. Another issue is the emergence of secondary physical effects such as diffractions in lithography, which makes it much more difficult to handle problematic patterns to reduce defects. For these reasons, it is often hard to tell whether design or fabrication causes the yield loss, and the gap between design and manufacturing becomes wider and wider.

To build up the bridge between design and manufacturing, DRC has been proposed in the verification process [3, 4]. DRC prevents hotspots by setting the constraints such as the minimal pattern width rule and the minimal spacing rule. Based on these geometric rules, DRC can identify most problematic patterns on a layout. To improve the design manufacturability, DRC has been used in the physical design process such as placement and routing to prevent illegal patterns in early design stages [5, 6]. However, with the increasing complexity of lithography and the occurrence of previously ignored side effects, geometry-based DRC alone cannot clean up all the layout hotspots [7]. Thus, it is desirable to develop a new verification flow to deal with the emerging problem.

The hotspot detection problem addresses how to find potential defect patterns before the fabrication stage so that these patterns can be fixed earlier to prevent the time-consuming back-and-forth process between design and manufacturing [8]. As the design complexity keep increasing, the hotspot detection has become popular in modern circuit verification.
Physical simulation is one of the hotspot detection methods, in which hotspots are detected by examining the patterns simulated with physical models in the fabrication process, such as lithography and etching models [9]. In 2011, Zhang et al. [10] proposed an effective lithography model to address the self-aligned double patterning decomposition problem with overlay minimization and hotspot detection. The experimental results have validated the proposed method and decomposition results for NanGate open cell library. Generally, physical simulation based hotspot detection is the most accurate method if physical models are correct; however, it has the drawback of long computing time on these physical models [11, 12]. Another difficulty is the increasing complexity of modeling due to more and more side physical effects that cannot be ignored.

Pattern matching-based methods detect problematic patterns by matching the patterns with a previously established pattern library [7, 13–16]. The patterns in the library are simulated and then classified according to their manufacturability. Previous works [17–23] have presented some state-of-the-art pattern matching techniques. Pattern matching-based hotspot detection methods can detect the layout patterns in the hotspot library fast and accurately; however, these methods lack the capability to find undefined or unknown problematic patterns.

Machine learning–based methods use machine learning models in the artificial intelligence domain. By giving the calibration data, a machine learning model is trained to find out the relationships among the training features and make decisions to the new testing data based on these relationships [24–30]. Recently, Agarwal et al. [31] presented a machine learning–based mechanism for detecting lithographic hotspots. Given a design layout, this method extracted frequency domain features to train a machine learning model and then classified a set of previously unseen patterns into hotspots and nonhotspots. Typically, machine learning–based hotspot detection methods can deal with never-seen-before patterns better compared with pattern matching-based approaches [32–35]; however, most of the machine learning–based methods suffer from low accuracy and high false alarm rate. In addition, their performance extremely depends on the calibration data and the learning model factors.

To improve machine learning–based approaches, the domain knowledge of the hotspot cause is needed to generate good calibration input vectors for the machine learning model. With properly selected training features and configuration, a machine learning model can approximate the simulation model with high detection accuracy and low false alarm rate [36].

For the generalized hotspot detection problem, in this paper, we use a two-stage algorithm flow which calibrates the machine learning model in the first stage and then predicts the hotspot positions in the second stage. The main contributions of this paper are summarized as follows:

(i) We present a two-dimensional discrete Fourier transformation–based pixel extraction method. Compared to the conventional pixel extraction approaches, our method is less sensitive to the shifting effect of a scan window.

(ii) We present a pattern-based layout scanning approach, which improves the program efficiency without loss of detection accuracy.

(iii) We present two false alarm reduction approaches to effectively reduce the number of detected nonhotspots and improve the accuracy of hotspot position.

(iv) Compared with three competitive works, experimental results based on the industrial benchmarks show the outperformance of our proposed algorithm in terms of accuracy, false alarm rate, efficiency, and time.

In the following sections, we introduce the problem description in Section 2. In Section 3, we present the two-stage algorithm flow. Section 4 presents the experimental results. Finally, we conclude this paper in Section 5.

2. Problem Description

The CAD Contest @ ICCAD is a research and development competition, focusing on advanced, real-world problems provided by industrial companies. In this paper, we aim to address the practical industry problem provided by the ICCAD’12 CAD Contest of Fuzzy Pattern Matching for Physical Verification [37]. To describe this problem clearly, we have the following definition.

Definition 1. Hotspots are the patterns or structures on a layout whose existence will produce yield loss on the wafer.

Definition 2. Hits are the patterns or structures on a layout which are correctly classified as hotspots.

Definition 3. The accuracy/hit rate of a hotspot detection result is the ratio of the hit number to the true hotspot number.

Definition 4. False alarms are the patterns or structures on a layout which are falsely classified as hotspots.

Definition 5. The false alarm rate of a hotspot detection result is the ratio of false alarm number to the true hotspot number.

Definition 6. The efficiency of a hotspot detection result is the ratio of the accuracy to the false alarm rate:

\[
\text{efficiency} = \frac{\text{accuracy}}{\text{false alarm rate}}
\]

Based on the above terminologies, the fuzzy pattern matching problem for physical verification is defined as follows:

(i) Problem: fuzzy Pattern Matching for Physical Verification.
(ii) **Instance**: a set of hotspot and nonhotspot patterns as the calibration data and a set of blind test layouts as the testing data are given.

(iii) **Question**: find the hotspot positions on the blind test layouts with a high accuracy and a low false alarm rate.

The given calibration data indicate the core area of each hotspot. Because of the intellectual property (IP) of the given layouts, each given hotspot/nonhotspot has a frame which contains limited patterns for calibration as shown in Figure 1. Because the given hotspots are extracted from a DRC-cleaned layout, the nonhotspots in the training data set outnumber the hotspots, which substantially reduces machine learning performance because of the over-tuning model for the nonhotspot patterns. Another issue is the diversity of the hotspot classes, which makes hotspot classification more difficult.

To be practical in the industry physical verification process, according to the contest metrics, the performance of hotspot detection must have over 80% accuracy with at most 100 false alarms per mm$^2$, and the runtime must be less than 1 hour per mm$^2$. However, the contest results are far behind the requirements, which means this problem is not easy and worth researching.

### 3. The Algorithm Flow

We propose a two-stage algorithm flow to detect hotspots as shown in Figure 2. In the first stage, the machine learning model is calibrated by pixel-based features, and then we predict the hotspot positions based on pattern-based layout scanning followed by false alarm reduction in the second stage.

In the following subsections, we discuss four important factors in our algorithm flow: (1) pixel-based feature extraction, (2) machine learning model, (3) pattern-based layout scanning, and (4) false alarm reduction.

#### 3.1. Pixel-Based Feature Extraction

Before building the machine learning models, we need to construct our relative hotspots features. In this subsection, we first introduce (1) pixel extraction, which is the basic pixel processing method, and (2) edge-based pixel extraction, which is an extension of (1). Furthermore, we propose our feature extraction method in (3) two-dimensional discrete Fourier transformation-based pixel extraction.

##### 3.1.1. Pixel Extraction

The pixel extraction method uses the pixel-image representation in the image processing domain to represent layouts [38]. In our paper, we adopt the well-known portable bitmap format (PBM), which represents the patterns as binary matrices. Figure 3(b) shows an example of the pixel-image of the original pattern in Figure 3(a). The grids covered by the patterns are denoted as 1. Figure 4(b) shows the extracted input feature vectors from the original pixel extraction method. It should be noted that, to feed the pattern feature to our machine learning model, we flatten the two-dimension pixel grids. For example, the feature vector index 17 in Figure 4(b) corresponds to the pixel at row 3 and column 5. Due to the shifting effect, the two vectors are staggered and quite different.

In order to alleviate the shifting effect in the pixel extraction, a more robust feature extraction method is required. Therefore, we propose a two-dimensional discrete Fourier (2D DFT) algorithm in this paper. The 2D DFT is defined as follows:

$$
\text{The 2D DFT is defined as follows:}
$$
where \( f(x, y) \) represents the element at the \( x \)-th row and \( y \)-th column of the matrix \( T \). Note that \( T \) is the \( M \times N \) matrix of a portable bitmap format. \( F(u, v) \) represents the element at the \( u \)-th row and \( v \)-th column of the matrix \( T' \), where \( T' \) is the \( M \times N \) complex-valued matrix that represents the result of the 2D Fourier transformed from \( T \).

Since in the frequency domain shifting the original function only affects the phase of the complex value, we choose the absolute value of the 2D DFT matrix as our feature to alleviate the shifting effect. Figure 4(c) shows that after 2D DFT, features of shifted frames \( A \) and \( B \) have very high similarities. Specifically, for other different patterns, their absolute values of the 2D DFT matrix are quite different. Thus, the DFT can be applied to differentiate patterns.

3.2. Machine Learning Model. Machine learning models are with great impacts on the solution qualities of classification tasks. Among machine learning techniques, the Support Vector Machine can provide great performance and small overfitting due to its optimal margin characteristic and thus is widely used for hotspot detection [26, 28, 29]. Given \( n \) feature-label pair \( (\mathbf{x}_i, y_i), \quad i = 1 \ldots n, \quad \mathbf{x}_i \in \mathbb{R}^d, \) and \( y \in \{-1, +1\} \) where \( d \) is the feature space dimension. The object function of the soft margin SVM is
\[
\min_{\omega, b, \xi} \frac{1}{2} \omega^T \omega + C \sum_{i=1}^{n} \xi_i.
\]

subject to
\[
y_i (\omega^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.
\]

where \(C\) is the penalty constant for the violations \(\xi_i\), and \(\omega\) and \(b\) are the parameters to form the separating hyperplane \(y = \omega^T x + b\).

Because of the large scaling of the dataset and very high dimension of our feature space, we choose linear kernel \(K(x_1, x_2) = x_1^T x_2\) to reduce the training process time to an acceptable time duration.

3.3. Polygon-Based Layout Scanning. To detect all hotspot patterns, we have to inspect the full chip layout carefully. Raster scanning is a well-known approach to scan through a full layout and inspect local features [39]. Let \(w\) denotes the width of a scan window. Figure 5 shows that the raster
scanning starts from the upper left corner to the lower right corner of the chip with a step size of $w/2$ and there are 50% overlap between two scan windows. However, 50% overlap between the scan windows may still lose accuracy. Although shrinking the step size can solve the unmatched problem, the computational effort is increased significantly. Furthermore, if the layout contains lots of white space, raster scanning wastes time scanning low-density area. To overcome this drawback, we propose a polygon-based layout scanning approach. Our polygon-based scanning approach consists of three stages: (1) pattern checking, (2) pattern decomposition, and (3) rectangle scanning.

3.3.1. Pattern Checking. For each pattern in the layout, we first check the pattern boundary. Let $W$ and $H$ be the width and the height of a pattern, respectively. If $W \leq w/2$ and $H \leq w/2$, we directly extract the feature based on the window centered at the center of this pattern. Generally, $w$ is a user-specified parameter, and we empirically set the window length as 16 units of the circuit board to trade-off the runtime and solution quality. Figure 6(a) shows that the width and height of a pattern are both smaller than $w/2$, and the center of the scan window is set exactly at the center of the polygon.

3.3.2. Pattern Decomposition. For those patterns whose width or length are larger than $w/2$ and the shapes are not rectangular, we partition the patterns into rectangles in this stage. The pattern partition problem can be formulated as follows: given a pattern $p$, decomposition it into a set of rectangles $R = \{r_1, r_2, \ldots, r_{|R|}\}$. In this paper, we implement the effective partition algorithm [40] to achieve desired solution. The partitioner presented in [40] is an iterative algorithm. Each pass through the algorithm alters or reduces an array of points describing an increasingly simplified pattern and generates one rectangle, which is added to a list of rectangles describing the pattern. This algorithm continues to be iterated until the array of corner points is empty. Figure 6(b) shows an example that the height of the polygon is larger than $w/2$, and then the polygon is decomposed into two rectangles.

3.3.3. Rectangle Scanning. After the pattern decomposition stage, all patterns are rectangular. For each pattern, if both the width and the height of the pattern are smaller than $w/2$, we can directly extract features from the pattern by the polygon checking method. If either the width or the height of the polygon is larger than $w/2$, we use raster scanning to inspect the polygon. Figure 6(c) shows that after the polygon decomposition, the upper pattern can be directly handled by the polygon checking method. Figure 6(d) shows that after processing the upper pattern, since the height of the lower pattern is larger than $w/2$, the scan window starts from the upper side of the pattern and moves to the lower side.

3.4. False Alarm Reduction. Since the machine learning–based method can induce a large amount of false alarms and multiple detections on the same hotspot, to reduce the number of false alarms and make hotspot positions more accurate, we propose two false alarm reduction approaches: (1) prediction analyzing and (2) prediction clustering.

3.4.1. Prediction Analyzing. Considering that the scanning window may not cover all the features of a real hotspot, there exists additional room for improving the false alarms. In this part, we further analyze the neighboring area of each predicted hotspot. Let $P = \{p_i: 1 \leq i \leq |P|\}$ denote the set of the hotspot obtained by the previous stage and $(x_i, y_i)$ be the 2D-coordinate of the center point of hotspot $p_i$. Then, four new reference points $p_{ili}$, $p_{iri}$, $p_{ilb}$, and $p_{irb}$ are diagnosed by the machine learning model for each $p_i$. Figure 7 shows the relationship between $p_i$ and the four reference points. The coordinates of $p_{ili}$, $p_{iri}$, $p_{ilb}$, and $p_{irb}$ are

\[
\begin{align*}
(x_{ili}, y_{ili}) &= (x_i - k, y_i + k), \\
(x_{iri}, y_{iri}) &= (x_i + k, y_i + k), \\
(x_{ilb}, y_{ilb}) &= (x_i - k, y_i - k), \\
(x_{irb}, y_{irb}) &= (x_i + k, y_i - k),
\end{align*}
\]

where $k$ is a user-specified number deciding the size of concerned area. In our experiment, we set $k$ to be $w/2$.

The decision of $k$ affects the analysis performance. If $k$ is larger than half of the window length, $p_i$ cannot be fully analyzed because the four reference points are too far from $p_i$. On the other hand, if $k$ is too small, $p_{ili}$, $p_{iri}$, $p_{ilb}$, and $p_{irb}$ are too close to $p_i$, which makes the analysis less significant. If less than three of $p_{ili}$, $p_{iri}$, $p_{ilb}$, and $p_{irb}$ are diagnosed as hotspot, empirically, $p_i$ has low possibility to be a hotspot and can be removed from $P$. After processing all of the points in $P$, the analyzed set $P' = \{p_i: 1 \leq i \leq |P'|\}$ can be obtained as
where $R_i = \{p_{i_r}, p_{i_l}, p_{i_b}, p_{i_b}\} \cap H$. 

and $H$ is the set of points which are diagnosed as hotspots.

By reclassifying the hotspots with low possibility as nonhotspots, our prediction analysis can effectively reduce the number of false alarms.

3.4.2. Prediction Clustering. In this stage, we further reduce false alarms and improve the accuracy of hotspot position by considering the nearby point $p_i \in P'$ together. To do this, we divide the layout region into uniform nonoverlapping bin grids. And if the width and the height of bins are small enough, for each bin, we can merge hotspots in the bin into one by calculating the cluster center of hotspot patterns without reducing hit rate. Let the set $P_b = \{p_{bi}: 1 \leq i \leq |P_b|\}$ represent hotspot patterns in a bin $b$ and $x_{bi}$ and $y_{bi}$ be the 2D-coordinate of point $p_{bi}$. The coordinate of cluster center $p^*_b$ is

$$
(x^*_b, y^*_b) = \left( \frac{1}{|P_b|} \sum_{i=1}^{P_b} x_{bi}, \frac{1}{|P_b|} \sum_{i=1}^{P_b} y_{bi} \right),
$$

Finally, the set $\cup_b P^*_b$ is the clustering result of our proposing method. Figure 8(a) shows the hotspots and grid before taking the cluster center point. Figure 8(b) shows that after obtaining the cluster center, the original hotspots are removed.

4. Experimental Results

In this section, we first introduce our experimental setup and benchmarks. Then we show the experimental results of our proposed methods: (1) pixel-based feature extraction, (2) pattern-based layout scanning, and (3) false alarm reduction. Finally we compare our results with the top three winners of the ICCAD’12 CAD Contest.

4.1. Experimental Setup and Benchmarks. We implemented our methods in the C++ programming language and conducted our experiments on a Linux machine with 24 Intel 2.00 GHz CPUs and 72 GB memory. We used LIBSVM [41] for the machine learning SVM engine.

All the experiments were based on the benchmark suite of the ICCAD’12 CAD Contest of Fuzzy Pattern Matching for Physical Verification [37]. Due to the IP issue, the contest organizer cannot release the original blind test layouts used in contest evaluation, and thus only layouts of clipped and arranged version were released. Note that we adopted the same parameters for all the tested cases in our implementation.
Table 1 gives the benchmark statistics, where column “Technology” lists the technology of each benchmark, “#HST” lists the number of hotspots used for training, “#HSB” lists the number of hotspots used in blind test, and “Area” lists the area of each blind test.

All of the reported results are evaluated by three metrics: (1) accuracy, (2) false alarm rate, and (3) efficiency. These metrics are defined in Section 2. There is a trade-off between the accuracy and the false alarm rate; a higher accuracy typically incurs a higher false alarm rate. Thus the efficiency is considered as an important factor for evaluation. The three metrics are all considered as the contest metrics.

4.2. Pixel-Based Feature Extraction. We compared the three pixel-based feature extraction methods which are presented in Section 3.1: (1) pixel extraction, (2) edge-based pixel extraction, and (3) two-dimensional discrete Fourier transformation–based pixel extraction.

Table 2 shows the pixel-based feature extraction comparison results, where columns “Accuracy,” “False Alarm Rate,” and “Efficiency” list the three metrics of each feature extraction methods. Column “Pixel” indicates the pixel extraction, “Edge” indicates the edge-based pixel extraction, and “Fourier” indicates the two-dimensional discrete Fourier transformation–based pixel extraction.

Based on the results, our two-dimensional discrete Fourier transformation–based pixel extraction method can achieve 20% and 12% improvement on average accuracy compared to the others. However, the false alarm rate also increases and overall efficiency is lower than the others. Thus, we proposed the false alarm reduction approach in order to reduce the false alarm rate, whose performance is evaluated in Section 4.4. We conclude this subsection that our two-dimensional discrete Fourier transformation–based pixel extraction method can achieve higher accuracy since the method can deal with shifted frames.

4.3. Pattern-Based Layout Scanning. We compared our proposed pattern-based layout scanning approach with the well-known raster scanning approach presented in Section 3.3. Table 3 shows the comparison results, which lists the three metrics of each scanning approach. Columns “Raster” and “Pattern” indicate the raster scanning approach and the pattern-based layout scanning approach, respectively.

The experimental results show that our pattern-based layout scanning approach can achieve 19% improvement on accuracy and 33% on efficiency, compared with the raster scanning approach.

4.4. False Alarm Reduction. We evaluated the performance of our false alarm reduction approaches which are presented in Section 3.4. Table 4 shows the false alarm reduction results. Columns “Original” and “FAR” indicate the program without and with false alarm reduction respectively.

Based on the results, we conclude that our false alarm reduction approach can achieve 68% improvement on efficiency with less than a 1% accuracy overhead.

4.5. Overall Results. We compared our approach with the top three winners of ICCAD’12 CAD Contest. Note that the results are reported from final submission binaries of all teams which were tested by the released clipped benchmarks. Table 5 summarizes the experimental results, which lists the three metrics of each team, and column “Time” indicates the runtime of each benchmark by each team. Columns “1st,” “2nd,” “3rd,” and “Ours” indicate the 1st place team, the 2nd place team, the 3rd place team, and our approach, respectively.

Overall, compared with the 1st and 3rd teams, our approach averagely improves the efficiency by 22% and 18%, respectively. As for the 2nd team, although their efficiency is
their accuracy is the lowest among the four teams and even has two cases with accuracy lower than 50%. And the runtime of the 2nd team is the longest among the four teams. The 1st place team, which focused more on the accuracy metric, achieved the highest accuracy but suffered from the lowest efficiency. We conclude that our approach can achieve overall good average performance and tradeoffs, compared to the three top winners.

5. Conclusions

In this paper, we have applied machine learning which is based on pixel-based feature extraction to deal with the generalized hotspot detection problem. Our hotspot detection algorithm consists of a two-dimensional discrete Fourier transformation–based pixel extraction method, a pattern-based layout scanning approach, and two false alarm reduction approaches. The Fourier transformation–based feature extraction method is proposed to alleviate the shifting effect and produce stable hotspot features. The pattern-based layout scanning approach is presented to enhance the program efficiency while preserving good detection accuracy. Finally, the two false alarm reduction approaches are applied to effectively reduce the number of detected nonhotspots and further improve the accuracy of hotspot position. Experimental results based on the industrial benchmarks have shown that
our work is effective for the addressed problem, which can be optimized for faster detection coverage. Future work lies in addressing the pattern matching problem on other different test cases such as the ICCAD 2016 CAD Contest benchmark suite [42]. Besides, incorporating the hotspot detection process into the design flow to enhance the circuit performance is also an important topic needing further investigation.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This research was funded by the National Key Research and Development Project, under Grant 2018YFB2202704, the Fujian Science Fund for Distinguished Young Scholars, under Grant 2019J06010, and the Natural Science Foundation of Fujian Province of China, under Grant 2020J01843.

References


