Journal of
Micro/Nanolithography,
MEMS, and MOEMS

Nanolithography.SPIEDigitalLibrary.org

Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering

Bei Yu
Jhih-Rong Gao
Duo Ding
Xuan Zeng
David Z. Pan

SPIE.

Accurate lithography hotspot detection based on principal component analysis-support vector machine classifier with hierarchical data clustering

Bei Yu,[a,*] Jhih-Rong Gao,[a] Duo Ding,[b] Xuan Zeng,[c] and David Z. Pan[a]
[a]University of Texas at Austin, Department of Electrical and Computer Engineering, Austin, Texas 78712, United States
[b]Oracle Microelectronics, Austin, Texas 78727, United States
[c]Fudan University, State Key Laboratory of ASIC & Systems, Microelectronics Department, Shanghai 200433, China

Abstract. As technology nodes continue to shrink, layout patterns become more sensitive to lithography processes, resulting in lithography hotspots that need to be identified and eliminated during physical verification. We propose an accurate hotspot detection approach based on principal component analysis-support vector machine classifier. Several techniques, including hierarchical data clustering, data balancing, and multilevel training, are provided to enhance the performance of the proposed approach. Our approach is accurate and more efficient than conventional time-consuming lithography simulation and provides a high flexibility for adapting to new lithography processes and rules. [DOI: 10.1117/1.JMM.14.1.011003]

Keywords: lithography hotspot detection; machine learning; principal component analysis.

Paper 14102SSP received Jun. 28, 2014; revised manuscript received Aug. 29, 2014; accepted for publication Sep. 24, 2014; published online Nov. 4, 2014.

1 Introduction

With the continuous shrinking of technology nodes, layout patterns become more sensitive to lithography processes and degrade manufacturing yield. Lithography hotspots are forbidden topologies that need to be identified and eliminated during physical verification. Various designs for manufacturing (DFM) techniques have been proposed to avoid these hotspots.[2] In the meantime, there are resolution enhancement techniques, such as optical proximity correction, phase-shift mask, and off-axis illumination, to improve the printability of problematic topologies.[3,4] However, for a deep subwavelength process, preventing lithography hotspots is still challenging and requires accurate physical verification to identify these hotspots for improving yield.

In physical design and verification stages, the hotspot detection problem is to locate hotspots on a given layout with a fast turn-around time. Conventional lithography simulation[5,6] obtains pattern images using complicated lithography models. Although it is accurate, full-chip lithography simulation is computationally expensive and cannot provide quick feedback to guide the early physical design stages.

To overcome the runtime overhead from conventional lithography simulation recently, two hotspot detection methodologies have recently evolved: pattern matching-based methods[7–10] and machine learning-based methods.[1,11–17] In pattern matching-based approaches, a hotspot pattern is defined by its geometric characteristics, and the detection process involves matching the hotspot patterns with all layout patterns. This method relies on a set of predefined hotspot patterns, and patterns outside of the scope may all be viewed as nonhotspots. Defining too many hotspot patterns would lead to overestimation and overoptimization, while defining too few would limit the design space too aggressively. Although pattern matching-based methods are accurate and fast, how to properly define hotspot patterns is still the main issue. In machine learning-based approaches, a regression model is constructed according to a given training data which includes hotspot and nonhotspot patterns. The model is then used to identify hotspots on a given testing layout. Machine learning-based approaches enlarge the possible topologies for hotspots and can improve the detection rate. However, they also increase the false alarms, which mean some reported hotspots are not real hotspots.

Effective representation of layout data is essential for the hotspot detection problem and there have been several encoding methods proposed. Kahng et al. presented an early hotspot detection[7] that builds a graph for the full layout to reflect the pattern-related critical dimension (CD) variation. This method depends on a limited set of CD variation evaluation methods, and false alarms may be generated. Yu et al. proposed a DRC-based hotspot detection[10] by extracting critical topological features and modeling them as design rules. The method used to extract critical design rules is a crucial process for its performance because excessive rules would lead to numerous false alarms, while few rules would lead to missed real hotspots. A range pattern[8,9] is proposed to incorporate process-dependent specifications which then can be used to identify hotspots by performing a string matching. Recently, support vector nachine (SVM) has become a popular data learning model for hotspot detection. Drmanac et al.[11] utilize SVM to train patterns represented by the histogram extracted from pixel-based layout images. In Ref. 13, layout density-based metrics are extracted to train the SVM kernel. A hybrid pattern matching and machine learning-based approaches[14] are proposed to take advantage of both techniques.

*Address all correspondence to: Bei Yu, E-mail: bei@cerc.utexas.edu

The preliminary version of this work appeared in SPIE Intl. Symp. Advanced Lithography 2014.[1]

0091-3286/2015/$25.00

In this paper, we propose a high performance hotspot detection approach based on PCA-SVM classifier. Principal component analysis (PCA) is a technique for feature extraction and data reduction. To the best of our knowledge, it is the first time that a PCA-based method has been applied in hotspot detection or a manufacturing field. We will demonstrate the effectiveness of combining PCA with SVM: the combination can help to significantly improve detection accuracy. Besides, our approach integrates the advantages of pattern matching and data learning, where pattern matching techniques enable high accuracy and data learning algorithms provide high flexibility to adapt to new lithography processes and rules. The main contributions include:

- We propose a multilevel PCA-SVM-based data learning flow that can extract critical layout information through mathematical analysis.
- We present a two-stage hierarchical data clustering approach to partition the layout data, such that irrelevant data can be processed by different classifiers for both efficiency and accuracy improvements.
- We apply several data compression techniques to enhance the performance of PCA-SVM, including data sampling for hotspot/nonhotspot imbalances and dimension reduction for encoded layout data.
- The experimental results show that our approach can effectively optimize accuracy and false alarms, where more than 80% of hotspots on all given testing layouts can be successfully identified.

The rest of the paper is organized as follows. We will first provide some preliminaries and the problem formulation in Sec. 2. Our proposed approaches, including hotspot model calibration and full layout detection, will be explained in Secs. 3 and 4, respectively. Finally, we will show our experimental results and performance analysis in Sec. 5, followed by the conclusion in Sec. 6.

2 Preliminaries and Problem Formulation

2.1 *Layout Pattern Representation*

A layout pattern becomes a hotspot not only because of the shape itself, but also because of the combined impact of its neighboring patterns. One fundamental step for the hotspot detection problem is to represent layout patterns with a certain format that can well describe the layout environment. In this paper, we adopt the concept of the fragmentation-based context characterization[18] to encode the layout patterns. This characterization method provides important layout information that is sufficient to describe a hotspot/nonhotspot, including pattern shapes, the distance between patterns, corner information (convex or concave), and so on.

Figure 1(a) shows the contour of three layout patterns and their corresponding Hanan grids. In geometry, the Hanan grid of a finite set of points in the plane is obtained by constructing vertical and horizontal lines through each point in this set. It is named after Maurice Hanan, who was the first person to introduce this graph.[19] Fragments are generated based on these grids as shown in (b). For each fragment f, an effective radius r is defined to cover the neighboring fragments which need to be considered in the context characterization of f. The radius r is process-dependent, which shows how neighboring patterns can affect the fragment of interest f. We then extract all fragments f_r covered by the circle with radius r as shown in Fig. 1(b) and their properties. A complete representation of f includes the geometric characteristic of each f_r, such as the length, corner, space, etc., which is stored as a vector for each fragment. In the following paper, we refer to the characterized vector of a fragment as fragment vectoer (FV).

The fragment generation in Ref. 18 is done by Calibre.[20] Since our hotspot detection flow is independent of Calibre, we generate fragments based on the Hanan grids.

2.2 *Problem Formulation*

There are two types of input layouts, training and testing layouts. In training layout, all hotspot locations are pointed out. An example of the input training layout is shown in Fig. 2, where two boxes indicate hotspots resulting from one-dimensional (1-D) and two-dimensional (2-D) patterns, respectively. Sometimes a 1-D pattern is called a uni-directional pattern, while a 2-D pattern is called a bi-directional pattern. For convenience, in the rest of this paper, we use the terms 1-D and 2-D patterns to refer to uni-directional and bi-directional patterns. The fragments within these hotspot locations are viewed as hotspot patterns, while the rest of the fragments are viewed as nonhotspot patterns. Given the

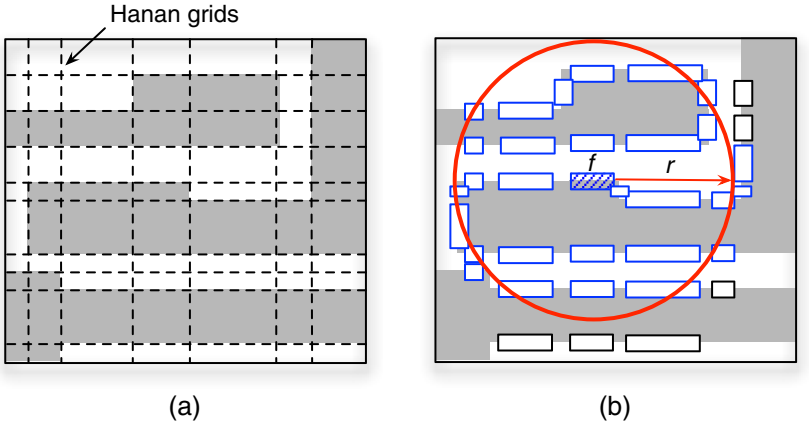

Fig. 1 Fragmentation-based hotspot signature extraction. (a) Layout patterns and the Hanan grids shown in dashed lines. (b) Fragmentation-based context characterization within the effective radius.

Fig. 2 Examples of hotspot patterns marked in red. (a) Hotspot resulted from one-dimensional (1-D) patterns only. (b) Hotspot resulted from complex 1-D and 2-D patterns.

training and testing layouts, the hotspot detection problem can be formulated as follows.

Problem 1 (hotspot detection). Given two sets of training layout clips, a set of hotspots and a set of nonhotspots, construct a system/model that can be used to identify unknown hotspots on a testing layout. The objective is to increase the number of true hotspots (**Hit**) and decrease the number of false hotspots (Extra).

Noted that when given a testing layout with unknown hotspots, the hotspot detection engine should report all possible hotspot locations. However, the reporting of excessive false hotspots would cause an overoptimization for the later hotspot fixing stage and should be minimized.

3 Hotspot Model Calibration

The hotspot detection is essentially composed of two steps: hotspot model calibration on training layouts, and hotspot detection on testing layouts. In this section, we will introduce our approaches to calibrate accurate hotspot classification models, which will be used in the hotspot detection process.

3.1 *Overall Data Calibration Flow*

Figure 3 shows our hotspot calibration flow. Given the training layout clips, we first decompose the layout patterns into small fragments based on Hanan grids, and collect a set of hotspot fragments and a set of nonhotspot fragments. We adopt the fragmentation-based pattern characterization method[18] to encode fragments, in which each fragment is represented by an FV. This characterization method provides layout information that is sufficient to describe a hotspot/nonhotspot environment, such as pattern shapes, the distance between patterns, corner information (convex or concave), and so on. First, we apply hierarchical data clustering to group similar fragments together based on their topological information (Sec. 3.2). Then the fragments in each cluster are sampled for data balancing (Sec. 3.3), and are then sent to our PCA-SVM-based learning process (Sec. 3.4). Finally, a set of hotspot classification models will be calculated for the use of the detection process.

3.2 *Hierarchical Layout Data Clustering*

The main objective of the hotspot calibration process is to build a model that can distinguish hotspots and nonhotspots. We observe that the accuracy of the calibrated model highly depends on the simplicity of the data. If the training data are very complicated, finding a general rule to classify them would be difficult and inaccurate. Figure 2 shows two hotspot examples highlighted in slashed rectangles; the hotspot in Fig. 2(a) simply results from 1-D patterns, while the one in Fig. 2(b) involves several 2-D patterns. Putting these two types of data in one classifier is already a challenge, not to mention there are many more types of hotspots. Training all data in a single classifier is not only time consuming, but also degrades the classification performance. Therefore, we propose a two-stage hierarchical layout data clustering approach to group the training data (fragments) according to their topological information.

The first-level clustering tries to cluster fragments by capturing the global view of the pattern environment; while the second-level clustering further clusters the fragments within each global cluster based on FV to reflect the local view. The first-level clustering is introduced in Sec. 3.2.1, and the second-level clustering is introduced in Sec. 3.2.2. By applying our clustering approach, the whole calibration problem can

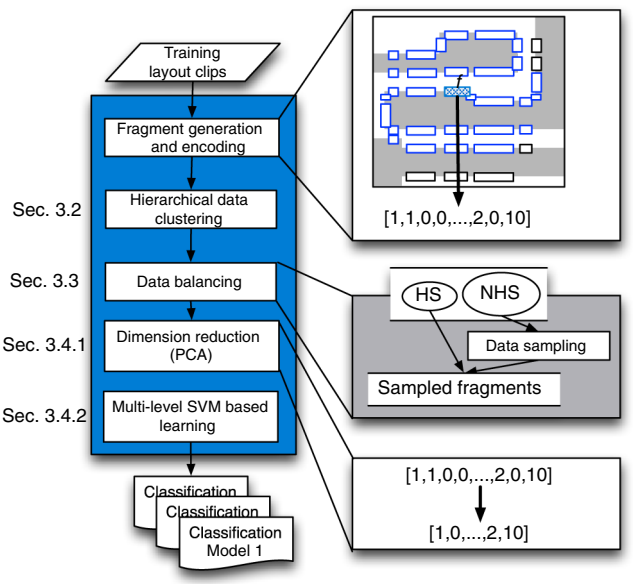

Fig. 3 Hotspot model calibration flow.

be divided into several independent subproblems. Then all the calibration results in different clusters can be merged into our final calibration result. We will show that this clustering approach helps to improve model accuracy and to reduce the overall runtime in our experimental results.

3.2.1 *Global pattern matching-based clustering*

In the first stage, we apply a pattern matching-based clustering technique to provide quick clustering results in a global view. Representative pattern types are predefined in our pattern matching engine. These pattern types are obtained by observing the common pattern combinations in the testing layout clips. We define an impact region based on the lithography process. For a fragment f, if there are only 1-D patterns in its impact region, f is clustered as a 1-D pattern. In general, 1-D and 2-D patterns are separated, and some special 1-D/2-D shapes are defined. Specifically, we define a 1-D-type pattern that includes one long feature as shown in Fig. 4(a), and another 1-D-type pattern that includes parallel 1-D features as shown in (b). On the other hand, if there is a 2-D pattern inside the impact region of a fragment f, it is clustered as a 2-D pattern. Figure 4(c) shows the pattern defined by an L-shaped feature and a long feature, while (d) shows a mountain-shaped pattern. In our implementation, we define six types of patterns: two types for 1-D patterns, and four types for 2-D patterns. The definition of the types is based on our observation that similar pattern types may share the same rules for detecting hotspot patterns.

Pattern matching is performed for each fragment to determine which pattern type a fragment belongs to. If no specific pattern type is found, this fragment is assigned to a default cluster. The pattern matching-based clustering only requires scanning the fragments within the impact region, thus it can be efficiently implemented.

3.2.2 *Local k-means clustering*

Once the pattern matching-based clustering is done, we further apply local clustering by k-means clustering[21] for each cluster obtained in the first stage. Given a set of N data points in d-dimensional space $\mathbf{R}^d$, we use k-means clustering to partition the points into k disjoint subsets S. The objective is to minimize the sum of the mean-squared distance within each cluster:

$$\min : \sum_{i=1}^{k} \sum_{n \in S_i} \left\| x_n - \mu_i \right\|^2, \tag{1}$$

where x_n is a vector representing the n-th data point, and μ is the mean of points in S_i. By mapping FVs with d elements to d-dimensional points, we can directly apply k-means clustering to partition the fragments inside each global cluster based on the differencees of their geometrical properties. Since Eq. (1) minimizes the sum of mean-squared distance, each dimension of a data point should be at the same scale. In our implementation, we normalize each element of FV before applying k-means clustering. Figure 5 illustrates one example of such a clustering. The initial patterns are further divided into five different subclusters through the k-means method. For each cluster, one typical layout is illustrated in Fig. 5.

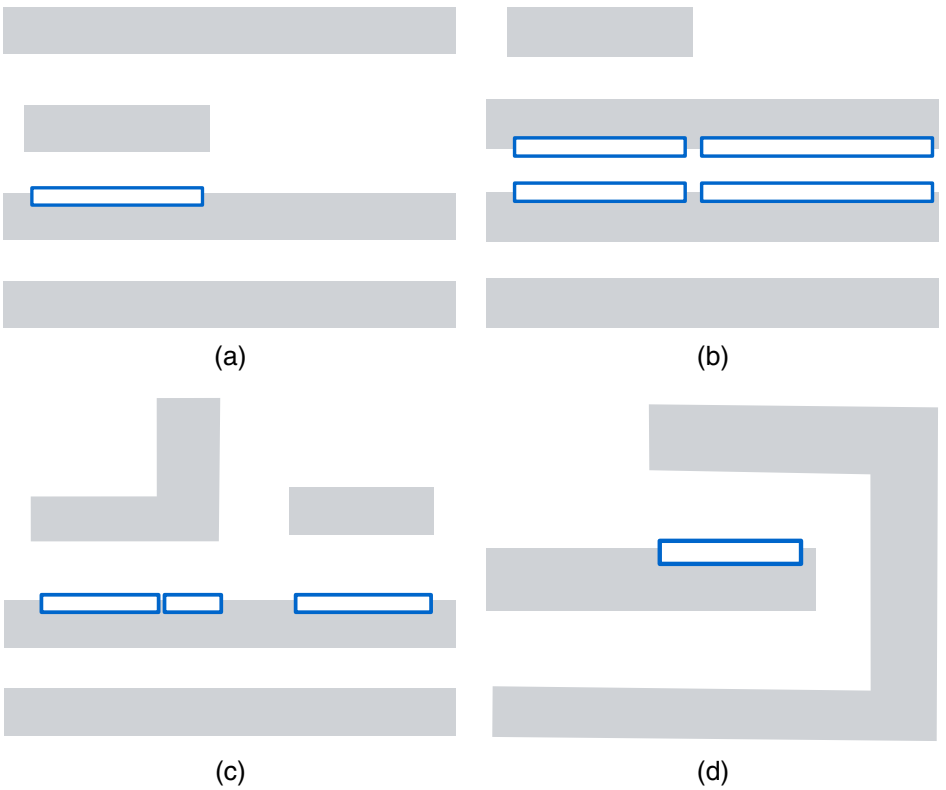

Fig. 4 Pattern types. (a) 1-D pattern including one long feature; (b) 1-D pattern including two parallel long features; (c) 2-D pattern including one long feature; (d) 2-D mountain-shaped pattern.

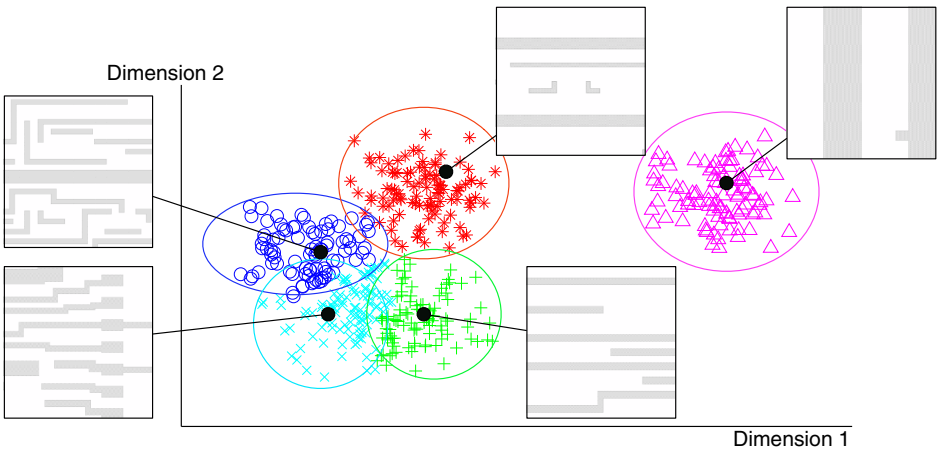

Fig. 5 *K*-means clustering with $k = 5$.

3.3 *Nonhotspot Data Balancing*

There are numerous various-shaped patterns in a layout, and the problem is that nonhotspot patterns greatly outnumber hotspot patterns.[22] For example, for a layout with hundreds of millions of patterns per square millimeter, the number of hotspots may be less than 100. The imbalance between hotspot and nonhotspot data is called imbalanced populations, which critically affect the success of SVM learning.[21] In addition, since we decompose the layout patterns as fragments and represent them with FVs, the size of the training data rapidly increases. It is important to shrink the data size to speed up the later data learning process. To enhance both accuracy and efficiency, we propose a data sampling technique to reduce the number of nonhotpot data.

There are two common sampling techniques: simple random sampling and systematic sampling. In simple random sampling, every element in the given dataset has an equal chance of being chosen. In systematic sampling, on the other hand, elements of the given dataset are first sorted in a certain order, and each element at a regular interval is selected. However, both of them suffer from some limitations and cannot be applied in our data sampling here. On one hand, simple random sampling is vulnerable to sampling error because the random selection may not reflect the real data distribution. On the other hand, the difficulty of sampling FVs with systematic sampling is that the dimension d of each FV may be very high. In our experience, we may need an FV with $d = 250$ to well describe the property of a fragment. The sorting process for all fragments would be time consuming. FVs are usually not evenly distributed in the d-dimensional space, which can result in over- or under-representation of the data.

Our sampling process utilizes k-means clustering to group together data with similar geographical information. We can choose the center of each cluster as the sampled data of the corresponding cluster, where the center is the mean of the data within a cluster. Using a larger number of clusters can minimize the data difference within a cluster and reduce the sampling error, while using a smaller number of clusters makes the training process faster with an average view of the data. By carefully choosing the size of the cluster, we can keep the main characteristics of each cluster without losing the sampling coverage.

3.4 *Multilevel Principal Component Analysis-Support Vector Machine-Based Classification*

3.4.1 *Dimension reduction with principal component analysis*

PCA[23] is a statistical technique that analyzes a set of data composed of possibly inter-correlated variables. The goal is to extract the important information of the original data and to represent the data as a new set of uncorrelated variables, called principal components. The number of principal components s is less than or equal to the number of the original variables. In the computer vision field, the combination of PCA and SVM[24,25] has been proven to improve the performance of pattern recognition. However, the PCA method is still new to manufacturing community. Therefore, in this work, we introduce the ideas of PCA. We apply PCA in front of our SVM process, which has the advantages of reducing the data size and increasing the hotspot classification accuracy. In other words, through the PCA process, the dimension of each fragment vector can be reduced. For example, the length of FV without PCA is 250, while the maximum length of FV with PCA is 80.

The PCA problem is defined as follows. Given a dataset $\mathbf{x} \in \mathbf{R}^d$, transform $\mathbf{x}$ into a new dataset $\mathbf{y} \in \mathbf{R}^s$:

$$\begin{aligned} y_{i,1} &= A_{11}x_{i,1} + A_{12}x_{i,2} + \ldots + A_{1\,s}x_{i,s} \\ y_{i,2} &= A_{21}x_{i,1} + A_{22}x_{i,2} + \ldots + A_{2\,s}x_{i,s} \\ &\cdots \\ y_{i,s} &= A_{s1}x_{i,1} + A_{s2}x_{i,2} + \ldots + A_{ss}x_{i,s} \end{aligned} \quad , \tag{2}$$

$$\forall\, x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})^T \in \mathbf{x}, \quad i = 1, \ldots, n$$

such that each $y_i \in \mathbf{y}$ explains as much as possible of the variance in the original dataset and that elements in $\mathbf{y}$ are uncorrelated. The correlation matrix A is a $d \times d$ matrix, which defines the new coordinate system. Each i'th column $A_i = (A_{i1}, A_{i2}, \ldots A_{is})$ is the i-th eigenvector of the data covariance matrix C.

$$C = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T. \tag{3}$$

PCA starts from calculating the covariance matrix C and then solves the eigenvector problem:

$$\mathrm{CA}_i = \lambda_i A_i, \quad i = 1, \ldots, n, \tag{4}$$

to obtain eigenvalues λ and their corresponding eigenvectors. The eigenvector with the largest eigenvalue captures the most variation among the training vectors $\mathbf{x}$, while the eigenvector with the smallest eigenvalue has the least variation.

Geometrically, PCA enables us to calculate a projection of the data to a subspace formed by eigenvectors corresponding to the most dominant eigenvalues. By sorting the eigenvalues in descending order, we can choose the first s principal components to represent the original data. This allows us to reduce our high-dimensional FV into a much shorter and more unique vector.

3.4.2 *Support vector machine with polynomial kernel*

SVM is a machine learning method for classification and learning tasks. In SVM, data vectors are mapped into a higher dimensional space using a kernel function, and an optimal linear discrimination function in the space or an optimal hyperplane that fits the training data is built. The objective is to maximize the margin between the separating hyper plane and the nearest data vectors from both classes. We adopt a two-class C-type SVM.[26,27] Given training vectors $x_i \in \mathbf{R}^d$, $i = 1, \ldots, n$ and an indicator vector $z \in \{1, -1\}$, the classical problem formulation is noted as follows.

$$\begin{array}{ll} \min: & \frac{1}{2}\alpha^T Q \alpha - e^T \alpha \\ \text{s.t.} & z^T \alpha = 0 \\ Q_{ij} = z_i z_j K(x_i, x_j) & i, j = 1, \ldots, n \\ 0 \le \alpha_i \le C, & i = 1, \ldots, n \end{array}, \tag{5}$$

where e is the vector of all ones, Q is an $n \times n$ positive semidefinite matrix, and the parameter C controls the trade-off between allowing training errors and forcing rigid separating margins. For each element $Q_{ij} \in Q$, $Q_{ij} = z_i z_j K(x_i, x_j)$. The kernel function K maps the data into a different space so a hyperplane can be used to do the separation. We use a polynomial kernel function in our implementation, which achieves the best results in our experiments. The objective function is a classical form for a C-type SVM, where α is variable and matrix Q and vector e are constants.

After solving the optimization problem in Eq. (5), all $\alpha_i \in \alpha$ can be calculated. Then, given a new testing vector t, the decision function is sgn $\left[\sum_i z_i \alpha_i K(x_i, t)\right]$, where the sgn function is defined as follows:

$$\mathrm{sgn}(x) = \begin{cases} -1, & \text{if } x \le 0 \\ 1, & \text{if } x > 0 \end{cases}.$$

In other words, for the testing vector t, if "$\sum_i z_i \alpha_i K(x_i, t) > 0$" it is not a hotspot. Otherwise, the vector t is detected as hotspot.

3.4.3 *Multilevel training for false alarm minimization*

We obtain several clusters from the clustering step explained in Sec. 3.2 and individually train a kernel for each cluster. The fact that hotspot data are far less than nonhotspot data significantly affects the performance of the SVM. We find that although our trained models can successfully identify true hotspots, numerous false alarms (Extra) are also reported. In order to reduce the number of false alarms, we adopt a multilevel self validation kernel structure. Conceptually, we verify our trained model using known data and collect false alarm information. These false alarms are fed into the training process in the next level, where the SVM model can focus on eliminating those false alarms.

Our multilevel kernel training flow is shown in Fig. 6. In first level, all data within a cluster are sent to the SVM kernel training process, where a classification model will be calculated. The classification model is tested with the same data used for training, thus we can verify the performance of the model by the number of Hit and Extra. If the number of Extra exceeds a certain threshold, we train another SVM kernel in the next level, where the input data will be only Hit and Extra data. Equation (6) is used to determine if the training process is continued, where th is a user-define threshold value for the false alarm rate.

$$\frac{\#\text{Extra}}{\#\text{Total Nonhotspots}} > \text{th}. \tag{6}$$

The larger th is, the greater the number of Hit; however, the number of false alarms also goes up. In our implementation, th is set as 5%.

It is worth mentioning that the data in each cluster are independent. Therefore, we can perform the kernel training and the later detection process in parallel. By taking advantage of multicore machines, our approach can be more efficient, which is a practical feature for modern complex layouts.

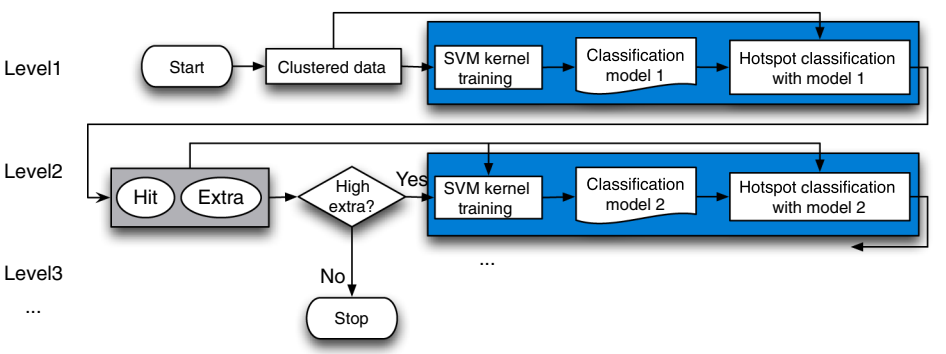

Fig. 6 SVM-based data learning flow.

4 Full Layout Hotspot Detection

Once the hotspot classification models are obtained in the training process, we can use these models to identify unknown hotspots on a given testing layout.

4.1 *Layout Scanning*

Given a full layout, we need to first generate fragments and encode each fragment by FV according to the geometric information in its nearby area. Constructing fragments and FV for the whole layout is time consuming and impractical since hotspots are only formed in small regions. Therefore, we propose a layout scanning technique to perform our hotspot checking process in a more efficient way. The basic idea is that the whole layout is decomposed into a set of grids, and the hotspot detection for each grid is independent. Therefore, the parallel techniques can be adopted here solve these independent hotspot detection problems.

First, the layout is decomposed into grids. The grid size is process-dependent and must be larger than the potential hotspot diameter to give sufficient information for FV. By default, we set the grid size to be the same as the frame size of the training clips. For each grid, we extract fragments and construct FV according to patterns inside the grid. Because the layout data outside of the grid are ignored at this time, we may miss some important information for fragments on the grid boundary. In order not to underestimate hotspots, we slightly enlarge the grid area whenever a grid is processed, as shown in Fig. 7(a). In this way, we create an overlapped checking area between adjacent grids, which helps to increase hotspot identification accuracy. The boxes in Fig. 7(b) show two adjacent checking areas by enlarging their corresponding grids; the slashed area will be checked twice (one for the red box and one for the blue box) to ensure the result is not biased by the grid boundary.

4.2 *Hotspot Identification Steps*

In the hotspot detection process, we are required to identify hotspots on a given testing layout using the prebuilt classification models. The hotspot identification flow is shown in Fig. 8. We first partition the layout into smaller grids and scan the layout on the grid base. The same fragment generation and data compression techniques are applied as the training flow. Each fragment to be tested will then be assigned into a specific cluster. Therefore, we can feed the fragments and FVs into their corresponding classification models obtained in the previous training process. According to our multilevel hotspot detection structure, a potential hotspot fragment must be identified by all classification models before they are reported. This helps to significantly reduce the number of false alarms. Note that the flow here is similar to that in the training process (see Fig. 3). However, one major difference is that here the classification models are

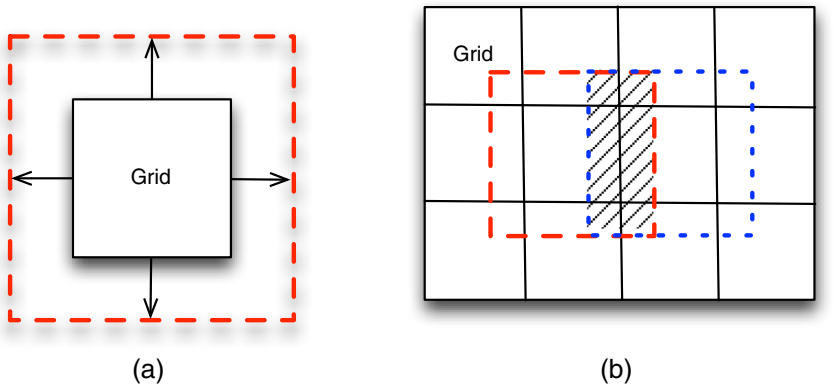

Fig. 7 Layout scanning. (a) Enlarged checking area. (b) Slashes show the overlapped area between adjacent checking areas.

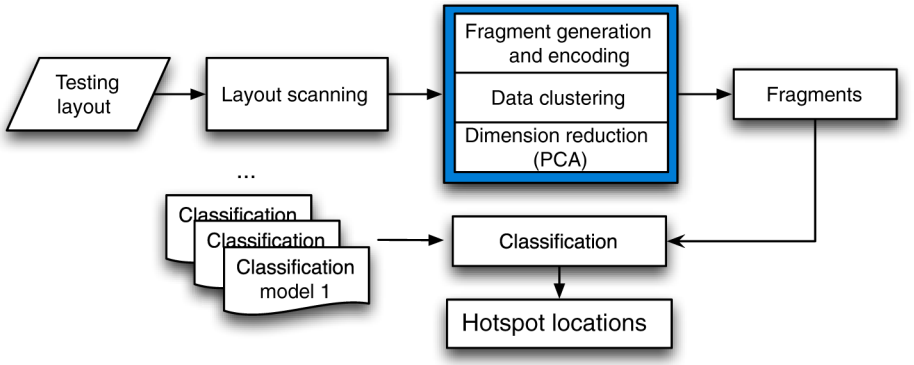

Fig. 8 Full layout hotspot detection flow.

Table 1 ICCAD12 benchmark statistics.

	Training layouts			Testing layouts		
Technique	Name	#HS	#NHS	Name	#HS	Area (mm^2)
32 nm	MX_benchmark1	99	340	Array_benchmark1	226	12,516
28 nm	MX_benchmark2	174	5285	Array_benchmark2	498	106,954
28 nm	MX_benchmark3	909	4643	Array_benchmark3	1808	122,565
28 nm	MX_benchmark4	95	4452	Array_benchmark4	177	82,010
28 nm	MX_benchmark5	26	2716	Array_benchmark5	41	49,583

input, while in the training process the classification models are calculated as output.

5 Experimental Results

The proposed algorithms are implemented in C++ and tested on the machine with eight 3.0 GHz CPUs and 32 GB memory. The OpenMP[28] library is used for our parallel implementation. We apply the same setting of parameters in our approach for all benchmarks. The number of local clusters is set as 10; the maximum number of sampled nonhotspot centers within a cluster is 500; and the number of principal components for FV is 80.

We test our approach on the industrial benchmarks released in Ref. 22. Table 1 shows the statistics of five benchmarks, including 32 and 28 nm designs. The training layouts are the input of the hotspot calibration process where hotspot and nonhotspot clips are given, while the testing layouts need to be verified by our hotspot detection flow to report the locations of identified hotspots. The number of total hotspot and nonhotspot clips is shown by #HS and #NHS, respectively. According to the definition in Ref. 22, a reported hotspot is a Hit if it overlaps a real hotspot in the testing layout, otherwise it is an Extra. Here, we define two important criteria to evaluate the performance of hotspot identification as shown in Eqs. (7) and (8). Both terms should be maximized.

$$\text{Accuracy} = \frac{\#\text{Hit}}{\#\text{HS}}, \tag{7}$$

$$\text{H/E Ratio} = \frac{\#\text{Hit}}{\#\text{Extra}}. \tag{8}$$

Table 2 shows our results compared with Ref. 15. Note that although Ref. 16 also utilizes ICCAD12 benchmarks, their

Table 2 Result comparison with Ref. 15.

Testing layout	Methods	Accuracy (%)	H/E ratio	CPU (s)
Array_benchmark1	Yu et al.[15]	94.69	0.143	38.1 s[a]
	Ours	80.97	0.253	63 s[b]
Array_benchmark2	Yu et al.[15]	98.20	0.041	3 min 54 s[a]
	Ours	81.12	0.041	34 min 57 s[a]
Array_benchmark3	Yu et al.[15]	91.88	0.123	14 min 58 s[b]
	Ours	90.93	0.098	29 min 42 s[a]
Array_benchmark4	Yu et al.[15]	85.94	0.045	5 min 56 s[b]
	Ours	87.01	0.057	13 min 8 s[a]
Array_benchmark5	Yu et al.[15]	92.86	0.032	20 s[b]
	Ours	80.49	0.049	8 min 26 s[a]
Overall improvement		−9.0	27.17%	

[a]8 Intel Xeon 3.0 GHz CPUs with 32 GB memory.
[b]2 Intel Xeon 2.3 GHz CPUs with 64 GB memory.

Table 3 Runtime breakdown.

	Benchmarks				
CPU time	B1 (s)	B2	B3	B4	B5
Training	55	29 min 4 s	23 min 34 s	11 min 14 s	7 min 21 s
Detection	8	5 min 53 s	6 min 8 s	1 min 54 s	1 min 5 s

approach does not process a full layout but only layout clips. Because of this fundamental difference, we cannot provide a fair comparison with Ref. 16. H/E ratio includes the information of both Hit and Extra. Since the hotspot detection problem requires both Hit maximization and Extra minimization, H/E ratio can more generally represent the overall performance. Hotspot detection should simultaneously optimize both the accuracy and H/E ratio. H/E ratio is an important metric, because each false alarm may require additional effort to manually modify the layout. From Table 2, we can see that our approach (Ours) steadily identifies more than 80% of the hotspots on all benchmarks and maintains a good H/E ratio at the same time. On average, our approach improves the H/E ratio by 27.17% compared with Ref. 15. The CPU time in Table 2 is the overall runtime including training and detection processes. We can see that our runtime is longer than Ref. 15. The reason for the longer runtime is twofold: (1) our training and detection processes are multilevel approaches. Compared with a conventional single level approach, a longer runtime is expected. (2) The potential hotspots are encoded into FVs. Therefore, for one region there may be many FVs, which increases the complexity of the training process. We believe applying another encoding method without fragments can effectively reduce the runtime.

Table 3 shows the training time and detection time of our approach. It can be seen that the runtime spent on prediction is relatively low. In a real application, the training process requires a one-time effort to build the classification model. Then the obtained models can be repeatedly used for layouts with the same process parameters. It is worthwhile to obtain an accurate model with an affordable runtime effort considering that the model determines the detection performance and is built only once.

Figure 9 lists six hotspot pattern examples that can be successfully detected by our approach. Figure 10 illustrates three false alarms. In other words, these three patterns are reported, but they are not real hotspots. Besides, Fig. 11 depicts some hotspot examples that cannot be detected by our current approach.

5.1 *Performance Analysis of Nonhotspot Data Balancing*

In Sec. 3.3, we introduce our data sampling technique for nonhotspots to alleviate the imbalance between hotspot and nonhotspot data. We adjust different sampling rates as Eq. (9) and see how the sampled data affect the results.

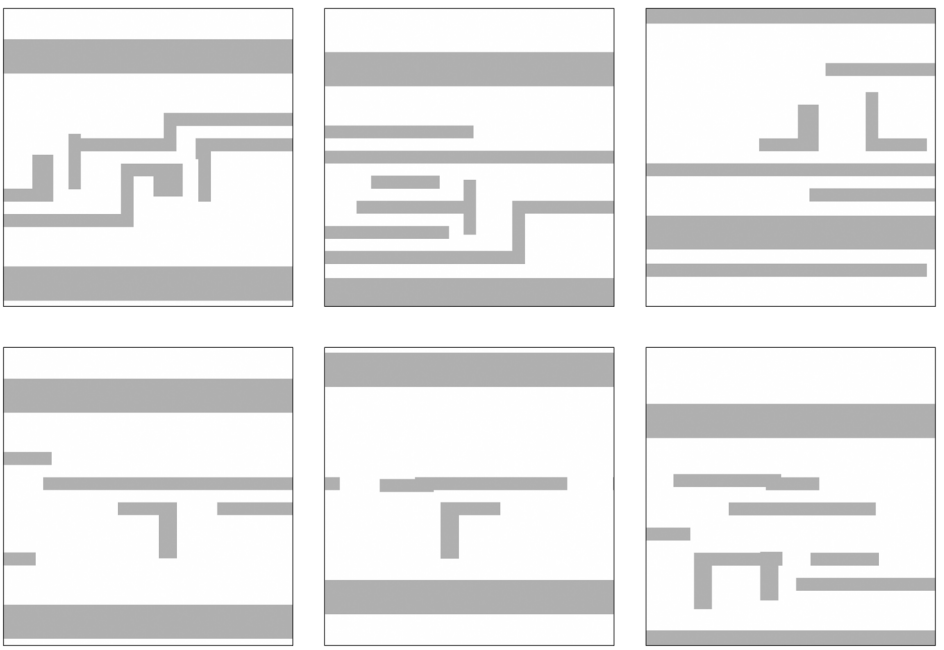

Fig. 9 Pattern examples of detected hotspots.

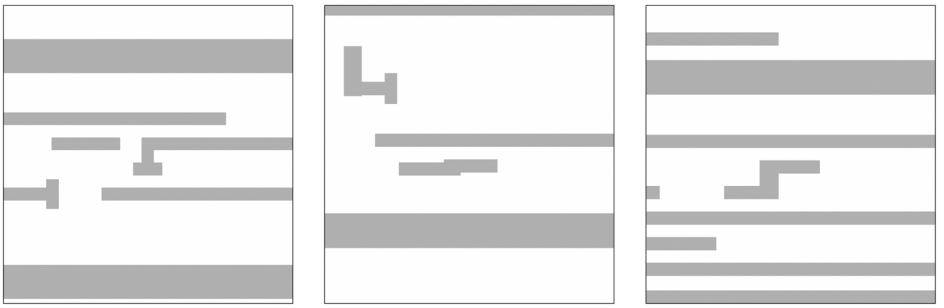

Fig. 10 Pattern examples of false alarms.

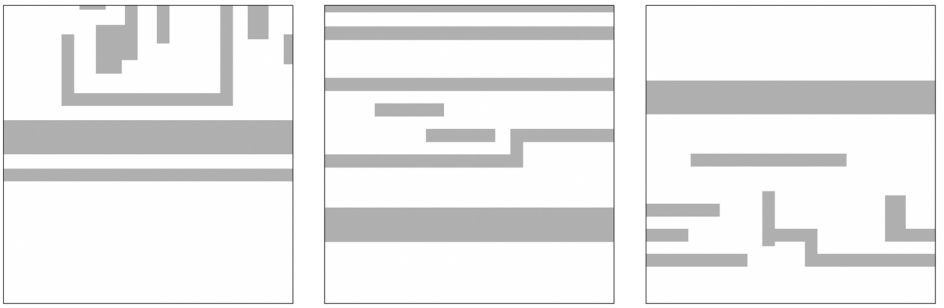

Fig. 11 Pattern examples of nondetected hotspots.

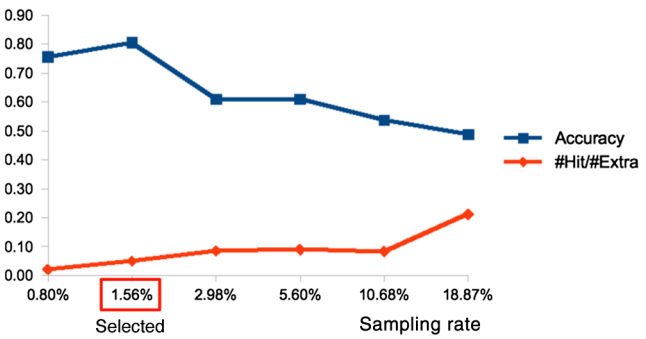

Fig. 12 Results comparison with different sampling rates for Array_benchmark5.

$$\text{Sampling Rate} = \frac{\#\text{Sample dfragments for the training process}}{\#\text{Total fragments}}. \tag{9}$$

Figure 12 shows the results of Array_benchmark5 with different sampling rates, where the x-axis is the sampling rate and the y-axis represents the values of the accuracy and the H/E ratio. One can observe that when the sampling rate gets higher, the results have a trend of lower accuracy and higher H/E ratio. The reason for such trend is twofold: (1) the sampling technique is working for nonhotspots, while hotspot patterns are all kept for the training process. As the sampling rate increases, more nonhotspot patterns are involved into our classification models. Therefore, more testing patterns tend to be treated as nonhotspots. (2) Our

Table 4 Comparison of results with PCA and without PCA applied.

	Without PCA			With PCA		
Benchmark	Accuracy (%)	H/E ratio	CPU (s)	Accuracy (%)	H/E ratio	CPU (s)
B1	80.09	0.217	69	80.97	0.253	63
B2	81.73	0.041	2005	81.12	0.041	2097
B3	85.90	0.074	1934	85.40	0.092	1782
B4	87.57	0.023	814	87.01	0.057	788
B5	82.93	0.036	496	80.49	0.049	506
Average	1	1	1	0.99	1.45	0.97

training process needs to ensure detection accuracy with multilevel SVM kernels. When the number of data is large, our training process would generate stricter detection models to prevent false alarms. In other words, too high a sampling rate may cause an "overfitting" problem. The trend of increasing H/E ratio and decreasing accuracy reflects the effect of the stricter models. The sampling rate needs to be properly chosen to maintain a good trade-off between the accuracy and the H/E ratio. In our implementation, we set 80% accuracy as our main optimization objective, then a higher H/E ratio is considered. As a result, the 1.56% sampling rate in Fig. 12 is selected as our final parameter.

5.2 *Performance Analysis of Principal Component Analysis-Based Support Vector Machine*

In order to understand the impact on SVM results by applying PCA, we implement two versions of our approach, one using the presented PCA-SVM (with PCA), and the other using a typical SVM (without PCA). The length of FV without PCA is 250, while the maximum length of FV with PCA is 80.

Table 4 shows the comparison of the five benchmarks in terms of accuracy, H/E ratio, and CPU Time. We can see that the difference in the accuracy is low, showing that reducing the vector dimension does not lose critical information. On the other hand, the H/E ratio is significantly improved in most cases, showing that eliminating less-relevant information using PCA helps to reduce false alarms. The results show the effectiveness of PCA-SVM on reducing the false alarms and the runtime, while maintaining the accuracy at the same time.

6 Conclusions

Lithography hotspots have a great impact on manufacturing yield. Identifying these forbidden pattern topologies in the physical verification or early physical design stage has become a critical problem. In this paper, we present a high performance hotspot detection approach based on PCA-SVM classifier. Several techniques, including hierarchical data clustering, data balancing, and multilevel training, are provided to enhance the performance of the proposed approach. Pattern matching techniques enable high accuracy and data learning algorithms and provide high flexibility to adapt to new lithography processes and rules. Our data clustering and data compression techniques help to improve the accuracy and to reduce the false alarms. The experimental results show that our approach effectively maximizes accuracy and minimizes false alarms at the same time, where more than 80% of the hotspots on all given testing layouts can be successfully identified.

Acknowledgments

The authors would like to thank Dr. Tetsuaki Matsunawa at Toshiba (currently a visiting scholar at UT Austin) for helpful discussion. This work is supported in part by NSF, SRC, NSFC, and Toshiba.

References

1. J.-R. Gao, B. Yu, and D. Z. Pan, "Accurate lithography hotspot detection based on PCA-SVM classifier with hierarchical data clustering," *Proc. SPIE* **9053**, 90530E (2014).
2. D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* **32**(10), 1453–1472 (2013).
3. P. Yu, S. X. Shi, and D. Z. Pan, "Process variation aware OPC with variational lithography modeling," in *Proc. IEEE/ACM Design Automation Conference*, pp. 785–790, ACM, San Francisco, CA (2006).
4. J.-R. Gao et al., "MOSAIC: mask optimizing solution with process window aware inverse correction," in *Proc. IEEE/ACM Design Automation Conference*, pp. 1–6, ACM, San Francisco, CA (2014).
5. J. Kim and M. Fan, "Hotspot detection on post-OPC layout using full chip simulation based verification tool: a case study with aerial image simulation," *Proc. SPIE* **5256**, 919–926 (2003).
6. E. Roseboom et al., "Automated full-chip hotspot detection and removal flow for interconnect layers of cell-based designs," *Proc. SPIE* **6521**, 65210C (2007).
7. A. B. Kahng, C.-H. Park, and X. Xu, "Fast dual graph based hotspot detection," *Proc. SPIE* **6349**, 63490H (2006).
8. H. Yao et al., "Efficient process-hotspot detection using range pattern matching," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 625–632, ACM, San Jose, CA (2006).
9. J. Xu, S. Sinha, and C. C. Chiang, "Accurate detection for process-hotspots with vias and incomplete specification," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 839–846, ACM, San Jose, CA (2007).
10. Y.-T. Yu et al., "Accurate process-hotspot detection using critical design rule extraction," in *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp. 1167–1172, ACM, San Francisco, CA (2012).
11. D. G. Drmanac, F. Liu, and L.-C. Wang, "Predicting variability in nanoscale lithography processes," in *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp. 545–550, ACM, San Francisco, CA (2009).
12. D. Ding et al., "Machine learning based lithographic hotspot detection with critical-feature extraction and classification," in *Proc. IEEE Int. Conf. on IC Design and Technology (ICICDT)*, pp. 219–222, IEEE, Austin, TX (2009).
13. J.-Y. Wuu et al., "Rapid layout pattern classification," in *Proc. IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 781–786, ACM, Yokohama, Japan (2011).
14. D. Ding et al., "EPIC: efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation," in *Proc. IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 263–270, IEEE, Sydney, NSW (2012).
15. Y.-T. Yu et al., "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp. 671–676, ACM, Austin, TX (2013).
16. S.-Y. Lin et al., "A novel fuzzy matching model for lithography hotspot detection," in *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp. 681–686, ACM, Austin, TX (2013).
17. H. Nosato et al., "Hotspot prevention and detection method using an image-recognition technique based on higher-order local autocorrelation," *J. Micro/Nanolithogr. MEMS MOEMS* **13**(1), 011007 (2014).
18. D. Ding et al., "High performance lithographic hotspot detection using hierarchically refined machine learning," in *Proc. IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, pp. 775–780, ACM, Yokohama, Japan (2011).
19. M. Hanan, "On steiner's problem with rectilinear distance," *SIAM J. Appl. Math.* **14**(2), 255–265 (1966).
20. M. Graphics, "Calibre verification user's manual," http://www.mentor.com (2008).
21. T. Kanungo et al., "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Trans. Pattern Anal. Mach.* **24**(7), 881–892 (2002).
22. J. A. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)* (2012).
23. I. T. Jollife, *Principal Component Analysis*, Wiley Online Library, Hoboken, New Jersey (2005).
24. O. Dniz, M. Castrilln, and M. Hernndez, "Face recognition using independent component analysis and support vector machines," *Pattern Recognit. Lett.* **24**(13), 2153–2157 (2003).
25. E. Gumus et al., "Evaluation of face recognition techniques using, PCA, wavelets and SVM," *J. Expert Syst. Appl.* **37**(9), 6404–6408 (2010).
26. B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Conf. on Learning Theory*, pp. 144–152, ACM (1992).
27. C. Cortes and V. Vapnik, "Support-vector networks," *J. Mach. Learn.* **20**(3), 273–297 (1995).
28. R. Chandra, "Parallel programming in OpenMP," Morgan Kaufmann, http://www.openmp.org/ (2001).

Bei Yu is a postdoctoral scholar in the Department of Electrical and Computer Engineering, University of Texas at Austin, where he

received his PhD degree in 2014. His research interests include design for manufacturability and optimization algorithms with applications in CAD. He received an SPIE education scholarship in 2013, IBM PhD scholarship in 2012, Best Paper Awards at ICCAD'13 and ASPDAC'12, and three other Best Paper Award nominations.

Jhih-Rong Gao is on the senior technical staff at Cadence Design Systems, Inc., Austin. She received her PhD degree from the Department of Electrical and Computer Engineering, University of Texas at Austin in 2014. Her current research interests include physical design and design for manufacturability. She was a research and development engineer with Synopsys, Inc., Taiwan, from 2007 to 2009. She was awarded the BACUS Photomask Scholarship from SPIE in 2013.

Duo Ding is a senior hardware engineer at Oracle Microelectronics, Austin, Texas. He received PhD degree from the Department of Electrical and Computer Engineering, University of Texas at Austin, in 2011. He has received the 2013 ACM SIGDA Best PhD Dissertation Award in Electronic Design Automation, and two Best Paper Awards in ICICDT 2009 and ASPDAC 2012.

Xuan Zeng is a professor in the Microelectronics Department, and serves as the director of State Key Laboratory of ASIC and Systems, Fudan University. She received the First-Class Award of Electronic Information Science and Technology from the Chinese Institute of Electronics in 2005, the Second-Class Award of Science and Technology Advancement and the Cross-Century Outstanding Scholar Award in 2006 and 2002, respectively. She received the award of "IT Top 10" in Shanghai in 2003.

David Z. Pan is a professor in the Department of Electrical and Computer Engineering, University of Texas at Austin. His research interests include nanometer IC design for manufacturability/reliability, new frontiers of physical design, and CAD for emerging technologies. He has published over 200 technical papers, and is the holder of 8 US patents. He has received many awards, including SRC Technical Excellence Award, 11 Best Paper Awards, among others. He is an IEEE fellow.