A New Lithography Hotspot Detection Framework Based on AdaBoost Classifier and Simplified Feature Extraction

Tetsuaki Matsunawa[a], Jhih-Rong Gao[b], Bei Yu[b] and David Z. Pan[b]

[a] Center for Semiconductor Research & Development, Toshiba Corp., Kawasaki, Japan
[b] ECE Department, Univ. of Texas at Austin, Austin, TX, USA
tetsuaki.matsunawa@toshiba.co.jp, {jrgao, bei, dpan}@cerc.utexas.edu

ABSTRACT

Under the low-k1 lithography process, lithography hotspot detection and elimination in the physical verification phase have become much more important for reducing the process optimization cost and improving manufacturing yield. This paper proposes a highly accurate and low-false-alarm hotspot detection framework. To define an appropriate and simplified layout feature for classification model training, we propose a novel feature space evaluation index. Furthermore, by applying a robust classifier based on the probability distribution function of layout features, our framework can achieve very high accuracy and almost zero false alarm. The experimental results demonstrate the effectiveness of the proposed method in that our detector outperforms other works in the 2012 ICCAD contest in terms of both accuracy and false alarm.

Keywords: Design for Manufacturability, Lithography Hotspot Detection, Machine Learning, Real AdaBoost

1. INTRODUCTION

Shrinking device feature sizes is the holy grail for the semiconductor industry and it is being pursued through the use of shorter wavelength and several design for manufacturing (DFM) technologies, such as optical proximity correction (OPC) and resolution enhancement techniques (RETs). In view of the delay of next generation lithography technologies, such as extreme ultra-violet lithography (EUVL), optimization of design for manufacturing process has recently become much more important.[1] However, for 45nm node and below, *lithography hotspot*, which is lower fidelity pattern on a wafer, still exists even after application of these DFM techniques. In the physical design and verification phase, a technique to detect these lithography hotspots is essential for improving manufacturing yield.

So far, lithography simulation is the most widely used hotspot detection method.[2] Although this method is expected to achieve very high accuracy, it is also known to be extremely time-consuming.[3] Therefore, a hotspot detection method with reasonable runtime without accuracy loss is sorely needed in the physical verification phase to avoid increase of turn-around time (TAT) and manufacturing cost. To achieve a balance between runtime and accuracy, several hotspot detection methods without lithography simulation have been proposed. Popular candidates are pattern matching based methods[4–8] and machine learning based methods.[9–20] Pattern matching based methods define a pattern library consisting of known hotspots, followed by the whole layout scanning with this library. Although pattern matching based methods are very fast and accurate in detecting known hotspots, these methods have a fundamental problem in that unknown-hotspot detection is impossible.

On the other hand, machine learning based methods generate classification models based on given hotspots and non-hotspots (training data set) and can detect unknown hotspots. Machine learning based methods have been shown to be of benefit for the hotspot detection problem in terms of runtime and detection accuracy. However, the contradiction between high detection accuracy and low false alarm remains a critical issue that inhibits the practical use of machine learning based methods, for the following reasons:

(1) Layout feature selection: The optimal layout feature that represents geometrical attributes of hotspots and non-hotspots appropriately is uncertain. On the one hand, short runtime can be expected from a simple layout feature in low-dimensional feature space, but this simplicity may also lead to deterioration of detection accuracy because it is unable to describe hotspot-specific attributes exactly. On the other hand, highly accurate detection in high-dimensional space causes over-fitting (false alarm) and long runtime due to complexity of feature space.

Design-Process-Technology Co-optimization for Manufacturability IX, edited by John L. Sturtevant, Luigi Capodieci,
Proc. of SPIE Vol. 9427, 94270S · © 2015 SPIE · CCC code: 0277-786X/15/$18 · doi: 10.1117/12.2085790

(2) Classification model training: It is difficult to define an efficient classification algorithm for a complicated feature space with the relatively small number of training data sets. Non-hotspots usually significantly outnumber hotspots in a layout, and furthermore, the amount of real hotspots may be less than 50 in a later phase of process development. To improve detection accuracy with a small number of training data, a nonlinear classification algorithm is essential. However, an algorithm with strong nonlinearity that classifies patterns in high-dimensional space, such as SVM, increases the possibility of false alarms, resulting in difficulty of highly accurate and low-false alarm detection.

So far, a fragmentation-based feature[14] and a density-based feature,[13] have been proposed to represent layout patterns. In addition, several learning models, e.g., artificial neural network (ANN)[9,12] and support vector machine (SVM),[11] have been applied. Multi-level learning[13] based on multiple SVM kernels, hierarchical learning,[14] and a hybrid method combining pattern matching and machine learning have recently been proposed.[15–18,20] The basic idea of these state-of-the-art methods is the construction of multiple nonlinear classifiers in a restricted feature space through classifying training data before model calibration. Although these works have shown that higher accuracy and lower false alarm can be achieved compared to previous techniques, there is still much room for improvement in view of runtime increase and many remaining false alarms.

To achieve a good trade-off between accuracy and false alarm, it is necessary to utilize an uninvolved layout feature that is simple but sufficiently appropriate to represent geometrical attributes of hotspots. In addition, a smart nonlinear classification algorithm with lower complexity than SVM is required as well. In this paper, we propose a high-accuracy and low-false alarm hotspot detection framework. A robust classifier in conjunction with base classifiers using a probability distribution function of simplified layout features realizes high-accuracy detection without false alarm increase. Our key contributions are as follows:

- We develop a feature space evaluation index to quantify different layout extraction methods and to define a learning-algorithm-friendly layout feature.
- We propose a simple but highly effective layout feature based on the feature space index, which provides an efficient description of layout patterns in low-dimensional space.
- We develop a new Adaboost machine learning algorithm that works together with our simplified layout feature to boost detection accuracy without causing false alarm increase.
- We demonstrate high accuracy and low false alarm under industry-strength benchmarks.

The rest of the paper is organized as follows. In Section 2, we give the problem formulation and the overall hotspot detection flow. In Section 3 and Section 4, we present details of the proposed simplified features and classification models, respectively. Section 5 presents the experiment results, followed by the conclusion in Section 6.

2. PRELIMINARIES

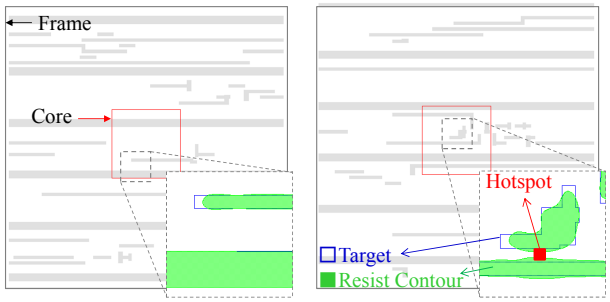
2.1 Problem Formulation

A hotspot is defined as a low-image-fidelity pattern on a wafer. As mentioned in the introduction, detecting all hotspots and eliminating them in the early phase of physical design and verification is becoming important. Fig. 1 (a) and (b) indicate examples of non-hotspot layout and hotspot layout, respectively. A "Frame" and a "Core" show the ambit associated with its center and the computational domain, respectively. The contour of a resist shape calculated by lithography simulation is also shown in each figure. We can see that, compared with non-hotspot layout, the hotspot layout shows a clear difference between the contour and the target shape. To evaluate the effectiveness of hotspot detection, we define several terms used throughout this paper.

Definition 1 (Accuracy). The detection accuracy rate is as follows:

$$Accuracy = \frac{Hit}{\#of\ hotspots} \tag{1}$$

where Hit is the number of correctly detected hotspots.



(a) Non-Hotspot. (b) Hotspot.

Figure 1. Examples of hotspot pattern.

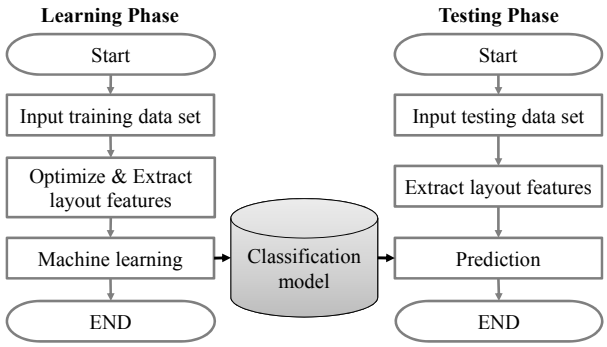


Figure 2. Overview of the proposed method.

Definition 2 (False alarm). The false alarm (extra) is as follows:

$$False\ Alarm = \#of\ falsely\ detected\ hotspots \tag{2}$$

Now we give the problem formulation of hotspot detection.

Problem 1 (Hotspot Detection). *Given layout data including hotspots and non-hotspots, a prediction model is calibrated to detect unknown hotspots from a verification layout. The goal of the hotspot detection is to maximize the accuracy and minimize the false alarm.*

In previous work it has been shown that optimization for either accuracy or false alarm is not difficult. However, it is really challenging to optimize accuracy and false alarm simultaneously.

2.2 Overview of Hotspot Detection Flow

Our hotspot detection method consists of two phases, "Learning phase" and "Testing phase", as shown in Fig. 2. In the learning phase, a training layout is given and a classification model is calibrated after optimization and extraction of a layout feature. The details of the layout feature and classification model making are described in Section 3 and 4. In the testing phase, a verification layout that is not the same as the training layout and includes unknown hotspots is used as the input. After the feature extraction from the verification layout, labels, which consist of -1 for non-hotspots and $+1$ for hotspots, are predicted by using the classification model trained in the learning phase.

As shown in Fig. 2, our proposed flow is simple and complicated processing such as layout classification before model learning in hierarchical or multi-level learning[13,14] does not need to be performed. Also, optimization with complicated nonlinear classification kernels is not required. The following are two reasons why high-accuracy detection and low false alarm can be expected with our simple flow. (1) Simplified Layout Feature: Simple but highly effective layout feature with high linear separability and low-dimensional space is defined by using our feature space index. (2) Boosting-Based Classification Model: Utilization of a weakly nonlinear learning algorithm is provided by using simplified layout feature.

3. SIMPLIFIED LAYOUT FEATURE

In hotspot detection, feature extraction is a vitally important phase, in which the initial geometrical information is translated into a set of layout features. Although several layout features have been proposed, the question of how to define the optimal parameters of layout features remains open. In this section, we present a feature space index to resolve this issue.

3.1 Feature Space Index

In order to define an appropriate layout feature, we propose a feature space index capable of optimizing the parameters of layout features. The basic idea is to evaluate the performance of layout features by measuring the distances between hotspots and non-hotspots in low-dimensional feature space. Based on the index, we can select an appropriate layout feature with a small amount of data while suppressing complexity of feature space. The definition of the index is as follows. The feature vectors X are projected to form new low-dimensional feature vectors P.

$$P = XU_k \tag{3}$$

In this paper, eigenvectors U are determined by using principal component analysis (PCA) to avoid redundant layout attributes and reduce dimensions.[20] In the PCA calculation, U are calculated by solving the following eigenvalue problem:

$$R_x U = U\Lambda \tag{4}$$

where R_x is the variance-covariance matrix of feature vectors X, Λ is the diagonal matrix of the eigenvalues in $(\lambda_1, ..., \lambda_M)$, and k in equation (3) is defined by the value of contribution rate η_k, which is 1.

$$\eta_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{M} \lambda_i} \tag{5}$$

In the projected feature space P, feature space index H is given by the following equation.

$$H = \left| 1 - \frac{1}{\alpha + \exp(-Z)} \right| \tag{6}$$

where α, discussed in detail below, is a hyper-parameter in consideration of generalization capability and Z is average distance of hotspot features to non-hotspot features.

$$Z = \frac{1}{N} \sum_{i=1}^{N} d_i \tag{7}$$

$$d_i = \frac{\sqrt{(x_i - \mu)^{\mathrm{T}} V^{-1} (x_i - \mu)} - d_{NHS_{min}}}{d_{NHS_{max}} - d_{NHS_{min}}} \tag{8}$$

where N is the total number of real hotspots in the training set, d is the Mahalanobis distance[21] normalized by non-hotspot features, μ is the center of mass of non-hotspot features, V is the variance covariance matrix of non-hotspot features, $d_{NHS_{max}}$ is the maximum Mahalanobis distance of non-hotspot features, and $d_{NHS_{min}}$ is the minimum Mahalanobis distance of non-hotspot features.

With the above definitions, the linear separability and predicting performance can be estimated by H and Z in a given layout feature space. For a layout extraction feature, if its resulting Z is in the range of $1 < Z < 10$ and its resulting H is near or toward zero, it indicates that it is an appropriate layout feature. Specifically, $Z < 1$ means that it is difficult to separate hotspots and non-hotspots linearly as most hotspots are located in the non-hotspot feature space. In contrast, $Z > 1$ shows linear-separation-friendly features because of the distance of hotspots from non-hotspots. However, too large Z indicates deterioration of prediction accuracy. When the hotspot features are too far away from the non-hotspot features, the allowable range of decision boundary is broadened, and as a result, generalization capability for the testing layout cannot be ensured. The appropriate upper limit of Z depends on training data, but according to our preliminary experiments, it is up to 10.

In our framework, generalization capability is evaluated using H . In equation (6), $\alpha = 1$ is equivalent to sigmoid function, but this means the bigger Z the better. To prevent decreases in prediction accuracy, an appropriate criterion is defined by adjusting α. This parameter depends on technology node and automatic determination of the optimal value of α is a subject for future work. In this paper, we set α value as 0.9 by the prescribed preliminary experiments.

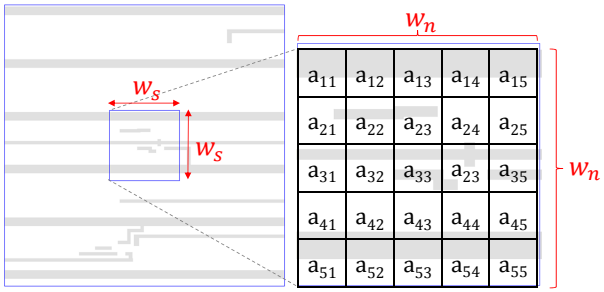


Figure 3. Density-based layout feature. Feature Vector is represented as: $X = \{a_{11}, a_{12}, ..., a_{54}, a_{55}\}$.

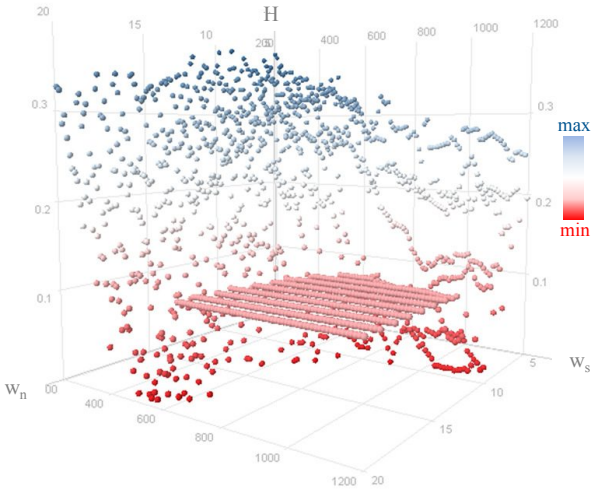


Figure 4. Solution space of density-based parameters.

3.2 Feature Selection and Optimization

We utilize a simplified density-based feature as an appropriate feature. "Simplified" means the feature with high linear separability in low-dimensional space and its parameters are optimized by using equation (6) and (7). Fig. 3 indicates the basic concept of the density-based feature. Feature vectors X show arrangement of area values of layout patterns in a given grid. Parameters of a feature w consist of the total size of the encoding area w_s and the number of grids w_n. Based on the discussion in the previous section, these parameters can be optimized by the following formula:

$$\text{minimize } H(w) \tag{9}$$

$$\text{subject to } w_s \in \{1, 2, ..., \mathcal{S}\} \tag{10}$$

$$w_n \in \{1, 2, ..., \mathcal{N}\} \tag{11}$$

Because equation (6) and (7) are nonlinear and parameters w are integer values, the above formulation is a nonlinear integer programing problem (NIPP). Fig. 4 indicates an example of the solution space of density-based parameters for training data set of Case 1 in Table 1 (See section 5). Since both integer linear programming and non-linear programming are well-known hard problems, it is not surprising that solving the combined version, NIPP, is challenging. However, as illustrated in Fig. 4, due to the limited solution space, we can find the optimal w using brute-force search method. The w_s is to be restricted within $\mathcal{S} = 1200$nm depending on core area of the layout data, and because the small grid size leads to increase in data, computational time and complexity of feature space, w_n is preferably $\mathcal{N} = 20$ or less. Fig. 4 shows a bottom (low H) region representing parameters expected to ensure high prediction accuracy with good linear separability.

The flat region indicates parameters restricted by α because the large w_n features are able to represent layout attributes in detail but it can also lead to a big difference in Z due to its complexity.

Although the density-based layout feature has been used successfully for the hotspot detection problem,[13,19] it is unclear how the parameters of the feature are defined. In this framework, our simplified feature can define the optimal parameters and represent larger area of the layout with lower-dimensional space compared to the conventional density-based method.[10] We will further discuss the advantages of the proposed layout feature compared to the conventional features in Section 5.

4. HOTSPOT DETECTION

As mentioned in the introduction, strong nonlinear classification models such as ANN and SVM are inappropriate for the hotspot detection problem in terms of false alarm reduction. Moreover, multi-nonlinear kernel models are unfit for low-false alarm detection and cause TAT increase. We therefore employ a decision tree classifier[22]

to prevent increasing of false alarms. A decision tree is simple to interpret and has advantages both of fast runtime and good performance even if its assumptions are somewhat violated by the true model. Furthermore, we enhance accuracy based on the probability distribution function of layout features corresponding with the complexity of parity structure in layout feature space.

We briefly describe the formulation to build a robust classifier as follows. Given the training data $x_n : n = 1, ..., N$, label of the training data $t_n \in \{-1, +1\}$ and number of base classifiers $m = 1, ..., M$, first the sample weight D is initialized.

$$D_1(n) = 1/N \tag{12}$$

Then the base decision tree classifier y_m is fitted using D_m to define candidate split $\theta = (j, s_k)$ of a feature j and threshold s at node k.

$$\theta = \text{argmin}_\theta(G, \theta) \tag{13}$$

$$G = \frac{n_l}{N_k} F(P(Q_{left})) + \frac{n_r}{N_k} F(P(Q_{right})) \tag{14}$$

$$Q_{left}(\theta) = (x, t)|x_j \leq s_k \tag{15}$$

$$Q_{right}(\theta) = Q \setminus Q_{left}(\theta) \tag{16}$$

$$P = \frac{1}{N_k} \sum_{i \in x^k} D_m(t_i = l) \tag{17}$$

$$F = \sum_l P(1 - P) \tag{18}$$

where G is Gini index, Q is new tree, n_l, n_r are the numbers of x in new trees, N_k is the total number of x at node k, P is proportion of class $l = \pm 1$ and F is impurity function. Next, the probability distribution function W is defined and the output of the base classifier is set as follows:

$$W_l^j = \sum_{n: j \in J \wedge t_n = l} D_m(n) \tag{19}$$

$$y_m(x) = \frac{1}{2} \ln \left(\frac{W_{+1}^j + \epsilon}{W_{-1}^j + \epsilon} \right) \tag{20}$$

$$D_{m+1}(n) = D_m(n) \exp\left(-t_n y_m(x_n)\right) \tag{21}$$

where ϵ is a small positive constant. The processing above is repeated as many times as the number of M, and finally the robust classifier Y is given by the following equation.

$$Y(x) = \text{sign} \left[\sum_{m=1}^{M} y_m(x) \right] \tag{22}$$

This kind of classifier is known as "Real Adaboost", which was proposed by Friedman et al.[23] We show the details for calibration of our detection model in Algorithm 1, where "DECISIONTREE" is decision tree function to define θ in equation (13). The output of the base classifier is calculated by using the sample weight based on θ in (19) and (20).

In our framework, layout features are simplified by our feature space index but still contain a little complexity of parity structure, discussed in detail in Section 5.1, that includes several disjoint clusters in layout feature space. Thus, SVM cannot resolve the trade-off relation between accuracy and false alarm because it is difficult to select an appropriate kernel for this kind of layout feature. Although ANN might be able to learn this problem, it is very difficult to optimize its parameters while minimizing the false alarm. In contrast, a simple decision tree achieves a certain degree of accuracy without false alarm increase because this is a weakly nonlinear algorithm.[22] It should be noted that a weakly nonlinear algorithm is indicated to be a classifier which is only slightly correlated with the true classification model. However, our classifier is able to achieve a performance superior to that of other classification algorithms by adding base classifiers.

Algorithm 1 Real Adaboost with Decision Tree

Require: x, t, m
1: Initialize the sample weights D_1
2: **for all** base classifier **do**
3: Tree depth = 0
4: DECISIONTREE(x, t, D_m)
5: Calculate the probability distribution function W
6: Set the output of base classifier y_m
7: Update the sample weights D_m
8: **end for**
9: **return** Final classifier Y

Generally, Adaboost is sensitive to the data including lots of noise and outliers. However, it can be superior with respect to the over-fitting issue compared to other classification algorithms. In the hotspot detection problem, layout features contain few or no factors corresponding to noise or outliers because the layout patterns are restricted by design rules. Although the minuscule differences among the patterns, such as small jogs, may be regarded as a noisy factor, Real Adaboost is able to construct a robust classifier considering such small differences based on the probability distribution function of feature space.

5. EXPERIMENTAL RESULTS

The proposed methodologies are implemented in Python and accelerated by Cython on a Linux machine with eight 3.4GHz CPUs and 32GB memory. The industrial benchmark suite released by[24] is applied. This benchmark suite includes five test cases, and the statistics of each case are listed in Table 1. It should be noted that Case 1 and 3 have a sufficient number of hotspots (HS) compared to the non-hotspots (NHS). In Case 2, 4 and 5, however, model learning is expected to be very difficult because they have small numbers of hotspots, which are less than 5% compared to their total numbers of non-hotspots. These cases can be seen as disadvantageous for machine learning, and the performance of the model is greatly influenced by feature selection and the classification algorithm. We conducted two experiments related to feature space evaluation and comparison for hotspot detection accuracy, respectively.

Table 1. ICCAD 2012 Benchmark Data.

Name	Tech	Training data			Testing data	
		#HS	#NHS	HS rate	#HS	Area (mm^2)
Case 1	32nm	99	340	29.12%	226	12516
Case 2	28nm	174	5285	3.29%	498	106954
Case 3	28nm	909	4643	19.58%	1808	122565
Case 4	28nm	95	4452	2.13%	177	82010
Case 5	28nm	26	2716	0.96%	41	49583

5.1 Feature Space Evaluation

In order to define an appropriate layout feature in consideration of the complexity of feature space, we compare our simplified density-based feature with fragmentation-based feature[17] and conventional density-based feature[13] by using the method described in Section 3.1. Motif data constitutes the core area in the training data of the Case 1 layout. Fig. 5 shows the histograms of d_i values for each layout feature. The parameters in our feature are set as the size of encoding area $w_s = 1200$ nm and the number of grids $w_n = 10$.We use following parameters for the conventional density-based method: $w_s = 420$ nm and $w_n = 12$.Fig. 5(a) and (b) indicate that in the fragmentation-based and conventional density-based features, linear separation is hard as most d_i values are smaller than 1. In contrast, Fig. 5(c) represents that, with our simplified feature, linear separation is less difficult than with the other two methods because all d_i are larger than 1.

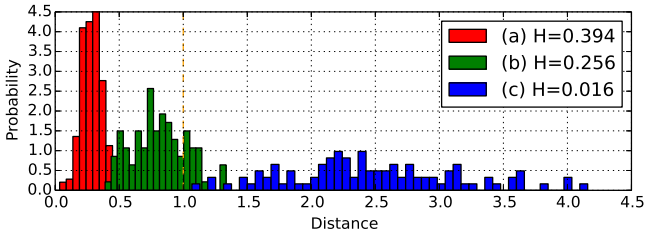
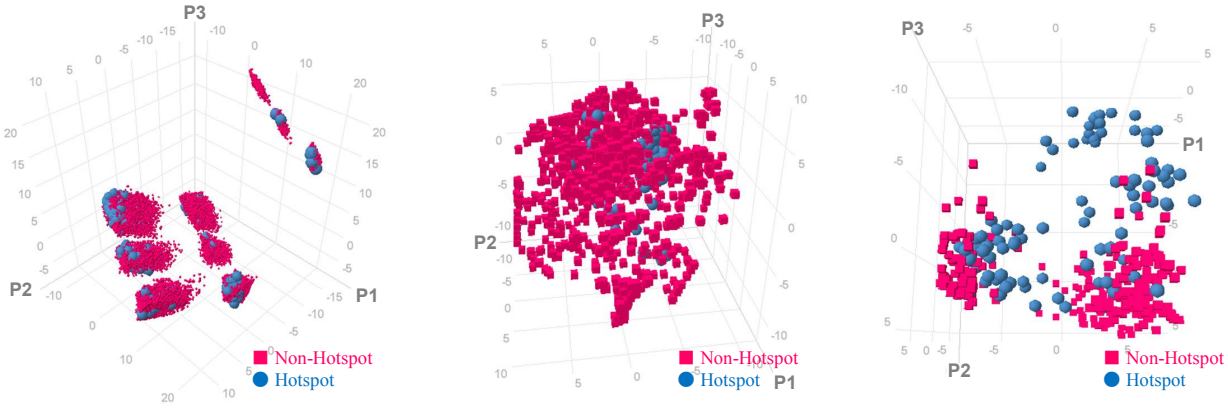


Figure 5. Histograms of distances (d_i) between hotspots and non-hotspots, (a) Fragmentation-based, (b) Conventional density-based, (c) Our simplified density-based.



(a) Fragmentation-based.

(b) Conventional density-based.

(c) Our simplified density-based.

Figure 6. 3D visualized images of feature space.

To check the complexity of feature space, reference data to which principal component analysis was applied is shown in Fig. 6. The three axes in Fig. 6 correspond to the orthogonal basis of the three higher eigenvalues. Comparing these figures, layout features in fragmentation-based space are separated into nine clusters, and hotspots are mixed with non-hotspots in each cluster. Thus, a multi nonlinear classification kernel technique such as those used in[13] or[14] is required for this kind of complicated feature space. However, a multi kernel method may cause over-fitting.Moreover, runtime will also increase because the data clustering process after the feature extraction phase is added in the testing phase. For the same reason, the conventional density-based feature is too complicated for a single classification kernel because most hotspot features are covered with non-hotspot features. In contrast, Fig. 6(c) indicates that our simplified feature can define the decision boundary easily due to its unsophisticated feature space.

5.2 Effectiveness of Hotspot Detection

In order to confirm the hotspot detection accuracy and false alarm, we compared our results with[18] as the highest level of accuracy and[19] as the lowest level of false alarm. Since the benchmark data consists of layout clips, we limited the verification region to the core area (layout clips only) in order to avoid uncertainty effect from outside the frame area. In parameter setting for Real Adaboost classifier, we adjust them by increasing the number of base classifiers and tree depth within the following ranges until the prediction accuracy in the testing data is stabilized: #classifiers are 10 to 300 and tree depth is 1 to 4. As shown in Table 2, with the exception of the Case 4, our approach achieves more than 95% average accuracy and 0 false alarm. Though the runtime of our method is comparable even with Python implementation, further acceleration, e.g., 10 to 100 times, is possible by implementation in a statically typed language such as C++.

To evaluate the impact of our proposed feature space index, we compare our simplified density-based feature

Table 2. Comparison with [18] and [19].

Layout	Methods	Accuracy	False Alarm	Runtime
Case 1	[18]	94.69%	1,493	38.1s
	[19]	62.30%	17	14.4s
	ours	100.00%	0	7.0s
Case 2	[18]	98.20%	11,834	3m54s
	[19]	43.30%	75	3m1s
	ours	98.60%	0	5m51s
Case 3	[18]	91.88%	13,850	14m58s
	[19]	42.50%	7	4m3s
	ours	97.20%	0	4m57s
Case 4	[18]	85.94%	3,664	5m56s
	[19]	52.60%	41	1m37s
	ours	87.01%	1	2m50s
Case 5	[18]	92.86%	1,205	20s
	[19]	53.70%	4	44.6s
	ours	92.68%	0	1m9s

Table 3. Comparison on different w_n.

Layout	Methods	Accuracy	False Alarm	Runtime
Case 1	Low-w_n	92.00%	3	4.8s
	High-w_n	100.00%	4	10.4s
	ours	100.00%	0	7.0s
Case 2	Low-w_n	83.10%	2	3m47s
	High-w_n	95.80%	0	9m1s
	ours	98.60%	0	5m51s
Case 3	Low-w_n	94.70%	43	3m44s
	High-w_n	97.60%	5	6m51s
	ours	97.20%	0	4m57s
Case 4	Low-w_n	72.88%	7	2m6s
	High-w_n	79.10%	2	3m46s
	ours	87.01%	1	2m50s
Case 5	Low-w_n	65.85%	2	1m5s
	High-w_n	90.24%	0	1m28s
	ours	92.68%	0	1m9s

Table 4. Comparison of different algorithms.

Layout	Methods	Accuracy	False Alarm	Runtime
Case 1	LR	92.50%	32	10.2s
	SVM	95.10%	14	10.21s
	ours	100.00%	0	7.0s
Case 2	LR	47.20%	320	5m49s
	SVM	77.50%	142	4m55s
	ours	98.60%	0	5m51s
Case 3	LR	54.70%	1,001	4m31s
	SVM	90.80%	258	4m26s
	ours	97.20%	0	4m57s
Case 4	LR	63.84%	149	3m55s
	SVM	66.67%	74	2m42s
	ours	87.01%	1	2m50s
Case 5	LR	63.41%	82	1m9s
	SVM	65.85%	21	1m6s
	ours	92.68%	0	1m9s

with the other two parameters. In Table 3, Low-w_n and High-w_n indicate $w_n = 5$ and $w_n = 20$, and H values of Low-w_n, High-w_n and ours are 0.196, 0.111 and 0.016, respectively. All w_s are set to 1200 nm. Table 3 shows that the feature parameters are optimized by our proposed feature space index to maximize the accuracy and minimize the false alarm.

We further compare our proposed Real Adaboost method with two other classification algorithms: Logistic Regression (LR) and SVM. We use the parameters for SVM defined by the grid search method: $C = 1000$, $\gamma = 100$. Table 4 shows the comparison results and indicates that Real Adaboost has the best performance for both accuracy and false alarm. It should be noted that LR is able to achieve reasonable accuracy even though it is a simple linear classification model. This result shows that the hotspot detection problem is simplified by larger-area representation based on our feature space index. This result also shows a little complexity of layout features due to the large number of false alarms in LR. However, the result in SVM indicates that it is difficult to resolve the trade-off relation between accuracy and false alarm with a strong nonlinear classification algorithm.

6. CONCLUSIONS

This paper proposes an accuracy-boosted hotspot detection method for layout optimization. By applying our feature space evaluation index for layout representation, a density-based layout feature is defined, which realizes extraction of layout attributes with low-dimensional space. The robust Real Adaboost classifier is able to detect hotspots accurately with extremely low false alarm. The experimental results show that our method can achieve

over 95% hotspot detection accuracy with almost zero false alarm and outperform the best published results for the ICCAD 2012 benchmarks. We believe this simple but effective methodology is promising to dramatically reduce the manufacturing and process optimization cost.

REFERENCES

[1] Pan, D. Z., Yu, B., and Gao, J.-R., "Design for manufacturing with emerging nanolithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* **32**(10), 1453–1472 (2013).

[2] Inoue, S., Kotani, T., Nojima, S., Tanaka, S., Hashimoto, K., and Mori, I., "Total hot spot management from design rule definition to silicon fabrication," in *Electronic Design Processes Workshop EDP*, (2003).

[3] Simmons, M. C., Kang, J.-H., Kim, Y., Park, J. I., weon Paek, S., and Kim, K.-s., "A state-of-the-art hotspot recognition system for full chip verification with lithographic simulation," in *Proc. of SPIE*, **7974** (2011).

[4] Kahng, A. B., Park, C.-H., and Xu, X., "Fast dual graph based hotspot detection," in *Proc. of SPIE*, **6925** (2006).

[5] Kang, J.-H., Choi, J.-Y., Shim, Y.-A., Lee, H.-S., Su, B., Chan, W., Zhang, P., Wu, J., and Kim, K.-Y., "Combination of rule and pattern based lithography unfriendly pattern detection in opc flow," in *Proc. of SPIE*, **71221** (2008).

[6] Yu, Y.-T., Chan, Y.-C., Sinha, S., Jiang, I. H.-R., and Chiang, C., "Accurate process-hotspot detection using critical design rule extraction," in *IEEE/ACM Design Automation Conference (DAC)*, 1167–1172 (2012).

[7] H.Yao, Sinha, S., Chiang, C., Hong, X., and Cai, Y., "Efficient process-hotspot detection using range pattern matching," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 625–632 (2006).

[8] Xu, J., Sinha, S., and Chiang, C. C., "Accurate detection for process-hotspots with vias and incomplete specification," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 839–846 (2007).

[9] Nagase, N., Suzuki, K., Takahashi, K., Minemura, M., Yamauchi, S., and Okada, T., "Study of hot spot detection using neural network judgment," in *Proc. of SPIE*, **6607** (2007).

[10] Wuu, J.-Y., Pikus, F. G., Torres, A., and Marek-Sadowska, M., "Detecting context sensitive hot spots in standard cell libraries," in *Proc. of SPIE*, **7275** (2009).

[11] Drmanac, D. G., Liu, F., and Wang, L.-C., "Predicting variability in nanoscale lithography processes," in *IEEE/ACM Design Automation Conference (DAC)*, 545–550 (2009).

[12] Ding, D., Wu, X., Ghosh, J., and Pan, D. Z., "Machine learning based lithographic hotspot detection with critical-feature extraction and classification," in *IEEE International Conference on IC Design and Technology (ICICDT)*, 219–222 (2009).

[13] Wuu, J.-Y., Pikus, F. G., Torres, A., and Marek-Sadowska, M., "Rapid layout pattern classification," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 781–786 (2011).

[14] Ding, D., Torres, A. J., Pikus, F. G., and Pan, D. Z., "High performance lithographic hotspot detection using hierarchically refined machine learning," in *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, 775–780 (2011).

[15] Wuu, J.-Y., Pikus, F. G., and Marek-Sadowska, M., "Efficient approach to early detection of lithographic hotspots using machine learning systems and pattern matching," in *Proc. of SPIE*, **7974** (2011).

[16] Mostafa, S., Torres, J. A., Rezk, P., and Madkour, K., "Multi-selection method for physical design verification applications," in *Proc. of SPIE*, **7974** (2011).

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

[18] Yu, Y.-T., Lin, G.-H., Jiang, I. H.-R., and Chiang, C., "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *IEEE/ACM Design Automation Conference (DAC)*, 671–676 (2013).

[19] Lin, S.-Y., Chen, J.-Y., Li, J.-C., Wen, W.-y., and Chang, S.-C., "A novel fuzzy matching model for lithography hotspot detection," in *IEEE/ACM Design Automation Conference (DAC)*, 681–686 (2013).

[20] Gao, J.-R., Yu, B., and Pan, D. Z., "Accurate lithography hotspot detection based on pca-svm classifier with hierarchical data clustering," in *Proc. of SPIE*, 90530E–90530E (2014).

[21] Mahalanobis, P. C., "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)* **2**, 49–55 (1936).

[22] Hastie, T., Tibshirani, R., Friedman, J., and Franklin, J., "The elements of statistical learning: data mining, inference and prediction," *The Mathematical Intelligencer* **27**(2), 83–85 (2005).

[23] Friedman, J., Hastie, T., and Tibshirani, R., "Special invited paper. additive logistic regression: A statistical view of boosting," *Annals of statistics* , 337–374 (2000).

[24] "ICCAD contest 2012." `http://cadcontest.cs.nctu.edu.tw/CAD-contest-at-ICCAD2012/problems/p3/p3.html`.