Accurate Lithography Hotspot Detection based on PCA-SVM Classifier with Hierarchical Data Clustering

Jhih-Rong Gao, Bei Yu, and David Z. Pan

ECE Dept. Univ. of Texas at Austin, Austin, TX USA 78712
{jrgao, bei, dpan}@cerc.utexas.edu

ABSTRACT

As technology nodes continues shrinking, layout patterns become more sensitive to lithography processes, resulting in lithography hotspots that need to be identified and eliminated during physical verification. In this paper, we propose an accurate hotspot detection approach based on PCA (principle component analysis)-SVM (support vector machine) classifier. Several techniques, including hierarchical data clustering, data balancing, and multi-level training, are provided to enhance performance of the proposed approach. Our approach is accurate and more efficient than conventional time-consuming lithography simulation; in the meanwhile, provides high flexibility to adapt to new lithography processes and rules.

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 design for manufacturing (DFM) techniques[1,2] have been proposed to avoid these hotspots. In the meantime, there are resolution enhancement techniques (RET), such as optical proximity correction,[3] phase-shift mask, and off-axis illumination, to improve the printability of problematic topologies. However, for deep sub-wavelength process, preventing lithography hotspots is still challenging and it requires accurate physical verification to identify these hotspots for yield improving.

In physical design and verification stages, the hotspot detection problem is to locate hotspots on a given layout with fast turn-around-time. Conventional lithography simulation[4,5] obtains pattern images by complicated lithography models. Although it is accurate, full-chip lithography simulation is computational expensive, and thus cannot provide quick feedback to guide the early physical design stages.

Recently, pattern matching based[6,7] and machine learning based[8–13] hotspot detection have become popular candidates. 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 pre-defined hotspot patterns, and patterns outside of the scope of this set may all be viewed as non-hotspots. Defining too many hotspot patterns would lead to over-estimation and over-optimization; 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 non-hotspot patterns. The model is then used to identify hotspots on a given testing layout. Machine learning based approaches enlarge the possible topologies for hotspots, therefore can improve the detection rate. However, it also increase the false alarms, which means some reported hotspots are not real hotspots.

Effective representation of layout data is essential for hotspot detection problem and there have been several encoding methods proposed. Kahng et al. presented an early hotspot detection[6] that builds a graph for the full layout to reflect pattern-related CD variation. This method depends on a limited set of CD variation evaluation methods, and thus false alarms may be generated. Yu et al. proposed a DRC-based hotspot detection[7] by extracting critical topological features and modeling them as design rules. How to extract critical design rules is a crucial process for its performance because excessive rules would lead to numerous false alarms, while too few rules would lead to missed real hotspots. Range pattern[14,15] is proposed to incorporate process-dependent specifications, which then can be used to identify hotspots by performing a string matching. Recently, Support

Design-Process-Technology Co-optimization for Manufacturability VIII, edited by
John L. Sturtevant, Proc. of SPIE Vol. 9053, 90530E · © 2014 SPIE
CCC code: 0277-786X/14/$18 · doi: 10.1117/12.2045888

Vector Machine (SVM) has become a popular data learning model for hotspot detection. Drmanac et al.[8] utilize Support Vector Machine (SVM) to train patterns represented by the histogram extracted from pixel-based layout images. In Ref. 10, layout density-based metrics are extracted to train the SVM kernel. A hybrid pattern matching and machine learning based approach[11] is proposed to take the advantages of both techniques.

In this paper, we propose a high performance hotspot detection approach based on PCA-SVM classifier. Principle component analysis (PCA) is a technique for feature extraction and data reduction; combining PCA with SVM helps to improve the detection accuracy significantly. 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 multi-level 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 improvement.
- We apply several data compression techniques to enhance the performance of PCA-SVM, including data sampling for hotspot/non-hotspot imbalance, and dimension reduction for encoded layout data.
- The experimental results show that our approach effectively maximizes accuracy and minimizes false alarms at the same time, where more than 80% of hotspots on all given testing layouts can be identified successfully.

The rest of the paper is organized as follows. We will first give the problem formulation in Sec. 2. Our proposed hotspot model calibration will be explained in Sec. 3. Finally, we will show our experimental results and performance analysis in Sec. 4, followed by the conclusion in Sec. 5.

2. PROBLEM FORMULATION

The hotspot detection problem can be formulated as follows. Given two sets of verified layout clips, a set of hotspots and a set of non-hotspots, 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 ($Extra$).

A sample of the input layout clip is shown in Fig. 1. A $Frame$ corresponds to the ambit or context area associated to its center. A $Core$, if indicated, corresponds to the central location where a hotspot appeared; otherwise the clip is free of hotspots. When given a testing layout with unknown hotspots, the hotspot detection engine should report all possible hotspot locations. However, excessive false hotspots reported would cause over-optimization for the later hotspot fixing stage, thus should be minimized.

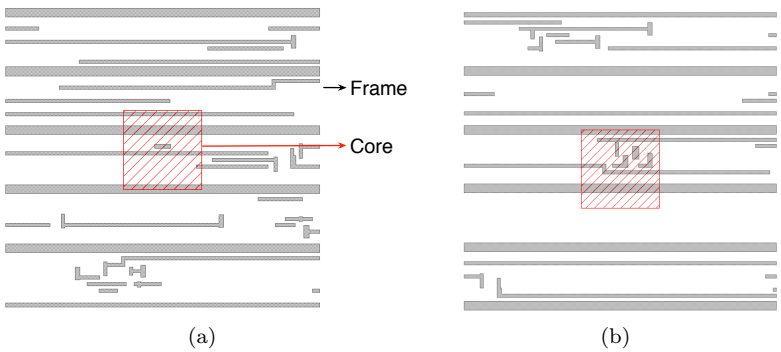
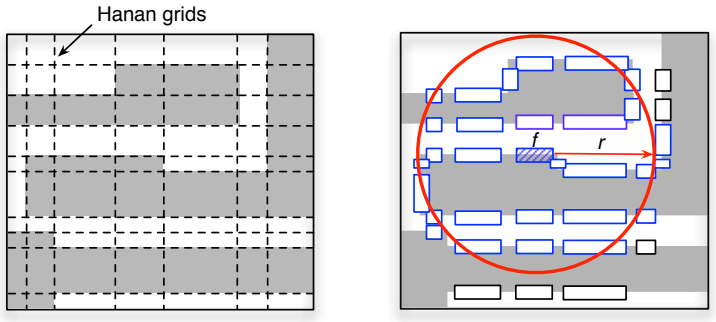


Figure 1. Examples of hotspot patterns marked in red. (a) Hotspot resulted from 1D patterns only. (b) Hotspot resulted from complex 1D and 2D patterns.



(a) (b)

Figure 2. 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.

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 certain format that can well describe the layout environment. In this paper, we adopt the concept of the fragmentation based context characterization[16] to encode the layout patterns. This characterization method provides important layout information that is sufficient to describe a hotspot/non-hotspot, including pattern shapes, the distance between patterns, corner information (convex or concave), and etc.

Fgiure 2 (a) shows the contour of three layout patterns and their corresponding Hanan grids. 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 relates to how neighboring patterns can affect the fragment of interest f. We then extract all fragments f_r covered by r as shown in blue in Fig. 2 (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 Vector* (FV).

The fragment generation in[16] is done by Calibre.[17] Since our hotspot detection flow is independent of Calibre, we generate fragments based on the Hanan grids in a *Frame*. All fragments inside a *Core* are viewed as hotspot patterns, and the rest of the fragments are viewed as non-hotspot patterns.

3. HOTSPOT MODEL CALIBRATION

The hotspot detection is essentially composed of two steps: hotspot model calibration with known patterns, 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

Fig. 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 non-hotspot fragments. We adopt the fragmentation based pattern characterization method[16] to encode fragments, in which each fragment is represented by a *Fragment Vector* (FV). This characterization method provides layout information that is sufficient to describe a hotspot/non-hotspot environment, such as pattern shapes, the distance between patterns, corner information (convex or concave), and etc. Next, we apply hierarchical data clustering to group similar fragments together based on their topological information (Sec. 3.2). Fragments in each cluster are sampled for data balancing (Sec. 3.3) and 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.

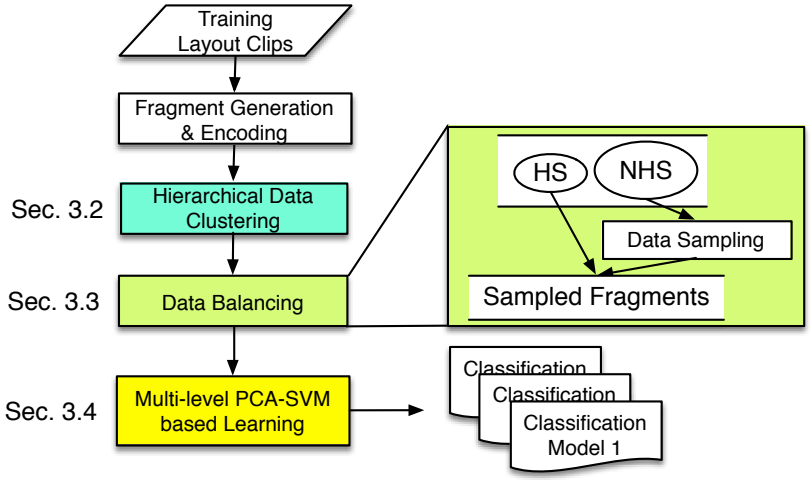


Figure 3. Hotspot model calibration flow.

3.2 Hierarchical layout data clustering

The main objective of the hotspot calibration process is to build a model that can distinguish hotspots and non-hotspots. We observe that the accuracy of the calibrated model highly depends on the simplicity of the data. If the training data is very complicated, finding a general rule to classify them would be difficult and inaccurate. Fig. 1 shows two hotspot examples highlighted in red slashed rectangles; the hotspot in (a) simply results from 1D patterns, while the one in (b) involves several 2D patterns. Putting these two types of data in one classifier is already a challenge, not to mention there are much more types of hotspots. Training all data in a single classifier not only is 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 try to cluster fragments by capturing the global view of the pattern environment; and the second-level clustering further cluster the fragments within each global cluster based on FV to reflect the local view. By applying our clustering approach, the whole calibration process can be done in a divide-and-conquer manner. We will show that this clustering approach helps to improve the model accuracy and reduce the overall runtime in our experimental results.

3.3 Non-hotspot data balancing

There are numerous various-shaped patterns in a layout, and the problem is that non-hotspot patterns greatly outnumbers hotspot patterns.[18] For example, for a layout with hundreds of millions patterns per mm^2, the amount of hotspots may be less than 100. The imbalance between hotspot and non-hotspot data is called *imbalanced populations*, which critically affect the success of SVM learning.[19] In addition, since we decompose the layout patterns as $fragments$ and represent them by FVs, the size of the training data increases rapidly. 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 non-hotpot data.

In simple random sampling, every element in the given data set has an equal chance to be chosen. However, this method is vulnerable to sampling error because the random selection may not reflect the real data distribution. In systematic sampling, on the other hand, elements of the given data set are first sorted in a certain order, and each element at a regular interval is selected. 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 a FV with $d = 250$ to well describe the property of a fragment. The sorting process for all fragments would be time-consuming. Besides, FVs are usually not evenly distributed in the d-dimensional space, which can result in over- or under-represented of the data.

Our data reduction approach utilize k-means clustering to group together data with similar geographical information. By doing this, 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.

3.4 Multi-level PCA-SVM based classification

3.4.1 Dimension reduction with PCA

PCA[20] 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 represent the data as a new set of uncorrelated variables, called principle components. The number of principle components s is less than or equal to the number of the original variables. In Computer Vision field, the combination of PCA and SVM[21,22] has been proven to improve the performance of pattern recognition. We apply PCA in front of our SVM process, which has the advantages of reducing the data size and increasing the hotspot classification accuracy.

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

$$\begin{aligned} &y_{i,1} = A_{11}x_{i,1} + A_{12}x_{i,2} + ... + A_{1s}x_{i,s} \\ &y_{i,2} = A_{21}x_{i,1} + A_{22}x_{i,2} + ... + A_{2s}x_{i,s} \\ &... \\ &y_{i,s} = A_{s1}x_{i,1} + A_{s2}x_{i,2} + ... + A_{ss}x_{i,s} \\ \\ &\forall x_i = (x_{i,1}, x_{i,2}, ..., x_{i,d})^T \in \mathbf{x}, \qquad\qquad i = 1, ..., n \end{aligned} \tag{1}$$

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

$$C = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T \tag{2}$$

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

$$CA_i = \lambda_i A_i, \quad i = 1, ..., n \tag{3}$$

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 principle 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 SVM 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 C-type SVM.[23,24] Given training vectors $x_i \in \mathbf{R}^d$, $i = 1, ..., n$, and an indicator vector $z \in \{1, -1\}$ for 2-class SVM, the problem formulation is briefed as follows.

$$\begin{aligned} \min_{\alpha}: \quad & \tfrac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ subject\ to \quad & z^T\alpha = 0 \\ & 0 \leqslant \alpha_i \leqslant C, \qquad i = 1, ..., n \end{aligned} \tag{4}$$

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

3.4.3 Multi-level training for false alarm minimization

We obtain several clusters from the clustering step explained in Sec. 3.2 and train a kernel for each cluster individually. The fact that hotspot data is far less than non-hotspot data significantly affect the performance of 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 multi-level 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 multi-level kernel training flow is shown in Fig. 4. In Level 1, all data within a cluster is 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, and 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. Eq. (5) is used to determine if the training process is continued, where α is a user-define parameter for the false alarm rate.

$$\frac{\#Extra}{\#Total\ Non\text{-}hotspots} > \alpha \tag{5}$$

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

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

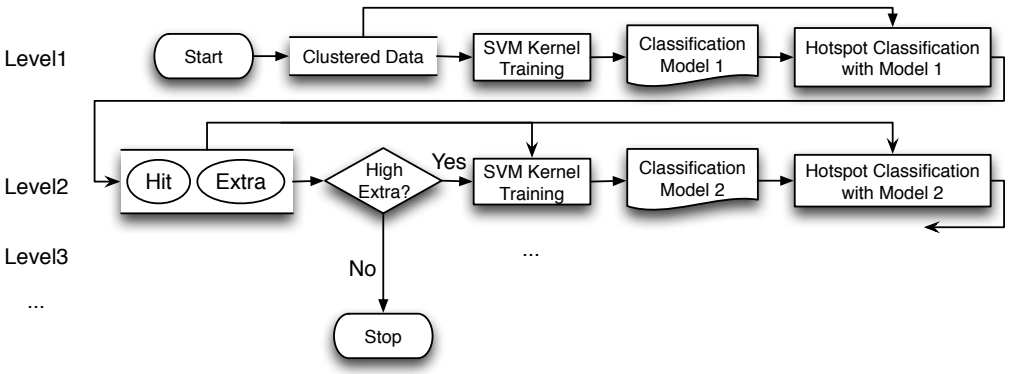


Figure 4. SVM based data learning flow.

4. 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[25] 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 non-hotspot centers within a cluster is 500; and the number of principle components for FV is 80.

We test our approach on the industrial benchmarks released in.[18] Table 1 shows the statistics of five benchmarks, including 32nm and 28nm designs. The training layouts are the input of the hotspot calibration process,

Table 1. ICCAD12 Benchmark statistics.

Tech	Training Layouts			Testing Layouts		
	Name	#HS	#NHS	Name	#HS	Area (mm^2)
32nm	MX_benchmark1	99	340	Array_benchmark1	226	12516
28nm	MX_benchmark2	174	5285	Array_benchmark2	498	106954
28nm	MX_benchmark3	909	4643	Array_benchmark3	1808	122565
28nm	MX_benchmark4	95	4452	Array_benchmark4	177	82010
28nm	MX_benchmark5	26	2716	Array_benchmark5	41	49583

Table 2. Result comparison with Ref. 12.

Testing Layout	Methods	Accuracy	H/E ratio	CPU (s)
Array_benchmark1	[12]	94.69%	0.143	$38.1s^*$
	Ours	80.97%	0.253	$63s^+$
Array_benchmark2	[12]	98.20%	0.041	$3m54s^*$
	Ours	81.12%	0.041	$34m57s^+$
Array_benchmark3	[12]	91.88%	0.123	$14m58s^*$
	Ours	90.93%	0.098	$29m42s^+$
Array_benchmark4	[12]	85.94%	0.045	$5m56s^*$
	Ours	87.01%	0.057	$13m8s^+$
Array_benchmark5	[12]	92.86%	0.032	$20s^*$
	Ours	80.49%	0.049	$8m26s^+$
Overall Impr.		-9.0%	27.17%	

* 2 Intel Xeon 2.3 GHz CPUs with 64 GB memory.
$^+$ 8 Intel Xeon 3.0 GHz CPUs with 32 GB memory.

where hotspot and non-hotspot 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 clips and non-hotspot clips are shown by $\#HS$ and $\#NHS$, respectively. According to the definition in,[18] 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 Eq. (6) and (7). Both terms should be maximized.

$$Accuracy = \frac{\#Hit}{\#HS} \tag{6}$$

$$H/E\ Ratio = \frac{\#Hit}{\#Extra} \tag{7}$$

Table 2 shows our results compared with Ref. 12. Note that although Ref. 13 also utilizes ICCAD12 benchmarks, their approach does not process a full layout but layout clips. Because of this fundamental difference, we cannot provide a fair comparison with Ref. 13. From Table 2, we can see that our approach ($Ours$) steadily identifies more than 80% hotspots on all benchmarks, and maintain good H/E ratio at the same time. 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. On average, our approach improves H/E ratio by 27.17% compared with Ref. 12.

The CPU time in Table 2 is the overall runtime including training and detection process. 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 real application, the training process takes 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 affordable runtime effort considering the model determines the detection performance and is built only once.

Table 3. Runtime breakdown.

CPU time	Benchmarks				
	B1	B2	B3	B4	B5
Training	55s	29m4s	23m34s	11m14s	7m21s
Detection	8s	5m53s	6m8s	1m54s	1m5s

4.1 Performance analysis of non-hotspot data balancing

In Sec. 3.3, we introduce our data sampling technique for non-hotspots to alleviate the imbalance between hotspot and non-hotspot data. We adjust different sampling rates as Eq. (8) and see how the sampled data affects the results.

$$Sampling\ Rate = \frac{\#\text{Sampled fragments for the training process}}{\#\text{Total fragments}} \tag{8}$$

Fig. 5 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 the trend of lower accuracy and higher H/E ratio. This is because our training process needs to ensure the detection accuracy by the multi-level SVM kernels. When the number of data is large, our training process would generate stricter detection models to prevent false alarms. The trend of increasing H/E ratio and decreasing accuracy reflects the effect of the stricter models. The sampling rate needs to be decided properly 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, and then higher H/E ratio is considered. As a result, the 1.56% compression rate in Fig. 5 is selected as our final parameter.

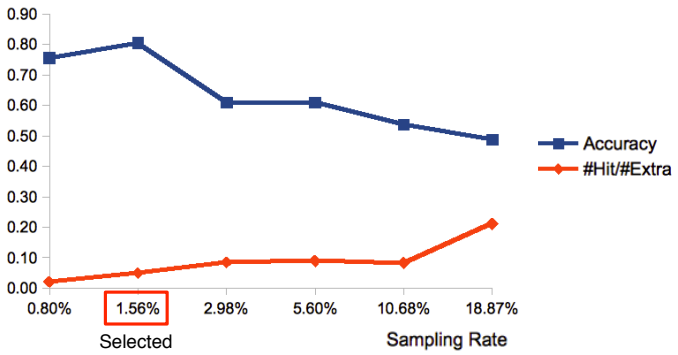


Figure 5. Results comparison with different sampling rates for Array benchmark5.

4.2 Performance analysis of PCA-based SVM

In order to understand the impact on SVM results by applying PCA, we implement two versions of our approach, one uses the presented PCA-SVM (w/ PCA), and the other uses typical SVM (w/o PCA). The maximum 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 on the accuracy is little, showing that reducing the vector dimension does not lose critical information. On the other hand, the H/E ratio are 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 runtime, while maintaining the accuracy at the same time.

5. CONCLUSIONS

Lithography hotspots have a great impact on the manufacturing yield. Identifying the forbidden pattern topologies in the physical verification or early physical design stage has become a critical problem. We present a high

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

Benchmark	w/o PCA			w/ PCA		
	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

performance hotspot detection approach based on PCA-SVM classifier. Several techniques, including hierarchical data clustering, data balancing, and multi-level training, are provided to enhance performance of the proposed approach. Pattern matching techniques enable high accuracy and data learning algorithms provide high flexibility to adapt to new lithography processes and rules. Our data clustering and data compression techniques help to improve the accuracy and 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 hotspots on all given testing layouts can be identified successfully.

Acknowledgment

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] B. Yu, J.-R. Gao, D. Ding, Y. Ban, J.-S. Yang, K. Yuan, M. Cho, and D. Z. Pan, "Dealing with ic manufacturability in extreme scaling," in *Proc. Int. Conf. on Computer Aided Design*, 2012.

[2] D. Z. Pan, B. Yu, and J.-R. Gao, "Design for manufacturing with emerging nanolithography," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* , pp. 1453–1472, 2013.

[3] P. Yu, S. X. Shi, and D. Z. Pan, "Process variation aware OPC with variational lithography modeling," in *Proc. Design Automation Conf.*, 2006.

[4] 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," in *Proc. of SPIE*, **5256**, 2003.

[5] E. Roseboom, M. Rossman, F.-C. Chang, and P. Hurat, "Automated full-chip hotspot detection and removal flow for interconnect layers of cell-based designs," in *Proc. of SPIE*, **6521**, 2007.

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

[7] Y.-T. Yu, Y.-C. Chan, S. Sinha, I. H.-R. Jiang, and C. Chiang, "Accurate process-hotspot detection using critical design rule extraction," in *Proc. Design Automation Conf.*, pp. 1167–1172, 2012.

[8] D. G. Drmanac, F. Liu, and L.-C. Wang, "Predicting variability in nanoscale lithography processes," in *Proc. Design Automation Conf.*, pp. 545–550, 2009.

[9] D. Ding, X. Wu, J. Ghosh, and D. Z. Pan, "Machine learning based lithographic hotspot detection with critical-feature extraction and classification," in *International Conference for Integrated Circuit Design Technology*, 2009.

[10] J.-Y. Wuu, F. G. Pikus, A. Torres, and M. Marek-Sadowska, "Rapid layout pattern classification," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 781–786, 2011.

[11] D. Ding, B. Yu, J. Ghosh, and D. Z. Pan, "EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 263–270, 2012.

[12] Y.-T. Yu, G.-H. Lin, I. H.-R. Jiang, and C. Chiang, "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *Proc. Design Automation Conf.*, pp. 67:1–67:6, 2013.

[13] S.-Y. Lin, J.-Y. Chen, J.-C. Li, W. yu Wen, and S.-C. Chang, "A novel fuzzy matching model for lithography hotspot detection," in *Proc. Design Automation Conf.*, pp. 68:1–68:6, 2013.

[14] H.Yao, S. Sinha, C. Chiang, X. Hong, and Y. Cai, "Efficient process-hotspot detection using range pattern matching," in *Proc. Int. Conf. on Computer Aided Design*, pp. 625–632, 2006.

[15] J. Xu, S. Sinha, and C. C. Chiang, "Accurate detection for process-hotspots with vias and incomplete specification," in *Proc. Int. Conf. on Computer Aided Design*, pp. 839–846, 2007.

[16] D. Ding, A. J. Torres, F. G. Pikus, and D. Z. Pan, "High performance lithographic hotspot detection using hierarchically refined machine learning," in *Proc. Asia and South Pacific Design Automation Conf.*, pp. 775–780, 2011.

[17] "Mentor Graphics Calibre [http://www.mentor.com/]."

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

[19] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm : Analysis and implementation," in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2002.

[20] I. T. Jollife, "Principal component analysis," in *Springer-Verlag*, 1986.

[21] O. Dniz, M. Castrilln, and M. Hernndez, "Face recognition using independent component analysis and support vector machines," in *Pattern recognition letters*, 2003.

[22] E. Gumus, N. Kilic, A. Sertbas, and O. N. Ucan, "Evaluation of face recognition techniques using pca, wavelets and svm," in *Journal of Expert Systems with Applications*, 2010.

[23] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. Computational Learning Theory*, 1992.

[24] C. Cortes and V. Vapnik, "Support-vector networks.," in *Journal of Machine Learning*, 1995.

[25] "OpenMP [http://www.openmp.org/]."