Machine-Learning-Based Hotspot Detection Using Topological Classification and Critical Feature Extraction

Yen-Ting Yu, Geng-He Lin, Iris Hui-Ru Jiang, *Member, IEEE*, and Charles Chiang, *Senior Member, IEEE*

***Abstract*—Because of the widening sub-wavelength lithography gap in advanced fabrication technology, lithography hotspot detection has become an essential task in design for manufacturability. Unlike current state-of-the-art works, which unite pattern matching and machine-learning engines, we fully exploit the strengths of machine learning using novel techniques. By combing topological classification and critical feature extraction, our hotspot detection framework achieves very high accuracy. Furthermore, to speed-up the evaluation, we verify only possible layout clips instead of full-layout scanning. We utilize feedback learning and present redundant clip removal to reduce the false alarm. Experimental results show that the proposed framework is very accurate and demonstrates a rapid training convergence. Moreover, our framework outperforms the 2012 CAD contest at International Conference on Computer-Aided Design (ICCAD) winner on accuracy and false alarm.**

***Index Terms*—Design for manufacturability, fuzzy pattern matching, hotspot detection, lithography hotspot, machine learning, support vector machine (SVM).**

I. Introduction

In ADVANCED process technology, the ever-growing sub-wavelength lithography gap causes unwanted shape distortions of the printed layout patterns [1]. Although design rule checking and reticle/resolution enhancement techniques, such as optical proximity correction and sub-resolution assist features, can alleviate the printability problem, many regions on a layout may still be susceptible to lithography process. These problematic regions, so-called lithography hotspots, have to be detected and corrected before mask synthesis.

Hotspot detection, therefore, is an essential task in physical verification. In this paper, we investigate the hotspot detection problem: given a training data set of hotspot and nonhotspot patterns and a testing layout, identify hotspots in the testing layout such that the accuracy is maximized and the false alarm is minimized.

Hotspot detection has attracted increasing attention recently, see [2]–[15]. These works can be classified into four major categories: 1) lithography simulation; 2) pattern matching; 3) data clustering and machine learning; and 4) hybrid. The full lithography simulation provides the most accurate detection result. However, the simulation suffers from an extremely high computational complexity and long runtime [2]. Pattern matching is the fastest hotspot detection approach in literature. This approach uses a specific representation to encode the topology of a hotspot pattern and/or layout, e.g., a dual graph [3], strings [4], [5], modified transitive closure graph (MTCG) [6], and improved tangent space [7]. Then, some searching algorithm is applied to identify hotspots. Pattern matching is good at detecting precharacterized hotspot patterns but has a limited flexibility to recognize previously unseen ones. In contrast, data clustering and machine learning are good at detecting unknown hotspots but need special treatments to suppress the false alarm. Data clustering and machine-learning methods involve two phases: first, the training phase constructs a hotspot model, and second, the evaluation phase verifies a testing layout. Typically, a supervised learning model is adopted, e.g., artificial neural network or support vector machine (SVM). Recent works further develop hierarchical learning, multilevel learning, and fuzzy cluster growing [8], [9], [14], [15]. The hybrid approach unites both pattern matching and machine-learning engines (even with a lithography simulator) to enhance accuracy and reduce false alarm but may consume longer runtimes, see [10]–[12].

In fact, prior endeavors have not fully exploited the strengths of machine learning because of the following issues: an adequate level of fuzziness/tolerance should be added to identify potential hotspots undefined in the training set without increasing false alarm. To achieve high accuracy and low false alarm, noise removal from training patterns and a balanced population between hotspot and nonhotspot samples is desired.

To properly address the above issues, in this paper, we propose a novel machine-learning-based hotspot detection framework with delicate techniques. In the training phase, our goal is to achieve a good training quality but maintain a high flexibility. First of all, we classify the known hotspot and nonhotspot patterns into clusters according to their

Manuscript received April 22, 2014; revised August 12, 2014 and October 22, 2014; accepted December 17, 2014. Date of publication January 6, 2015; date of current version February 17, 2015. This work was supported in part by Synopsys, and in part by Ministry of Science and Technology (MOST) of Taiwan under Grant MOST 103-2220-E-009-013. An earlier version was published at Association for Computing Machinery/the Electronic Design Automation Consortium/IEEE Design Automation Conference 2013 [13]. This paper was recommended by Associate Editor P. Gupta.

Y.-T. Yu and I. H.-R. Jiang are with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu 30010, Taiwan (e-mail: huiru.jiang@gmail.com).

G.-H. Lin was with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu 30010, Taiwan. He is now with Taiwan Semiconductor Manufacturing Company, Limited (TSMC), Ltd., Hsinchu 30078, Taiwan.

C. Chiang is with the Synopsys, Inc., Mountain View, CA 94043 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2014.2387858

topologies. Secondly, topological and nontopological critical features are extracted from each cluster. Thirdly, based on iterative multiple kernel learning, we construct a specific SVM kernel for each cluster. Because of topological classification, each kernel can concentrate on the critical features specific to its corresponding cluster, thus providing a flexibility to identify previously unseen patterns. Topological classification also facilitates hotspot and nonhotspot population balancing. We then utilize a feedback learning model to further suppress the false alarm. Compared with a single huge SVM kernel, our multiple kernel learning achieves high accuracy.

On the other hand, a testing layout contains tremendous sites that need to be evaluated. Therefore, in the evaluation phase, to avoid time-consuming full-layout scanning, we extract only possible layout clips based on the polygon density. After evaluation, we further filter the detected hotspots to reduce the false alarm without sacrificing the accuracy.

Experiments are conducted on six 32 and 28 nm industrial designs [16]. Experimental results show that our framework is very accurate. We also demonstrate a rapid training convergence, i.e., achieving high accuracy using a small amount of training data. Moreover, our framework outperforms the 2012 CAD contest at International Conference on Computer-Aided Design (ICCAD) winner on the overall performance, including accuracy and false alarm, with very competitive runtime.

The remainder of this paper is organized as follows. Section II describes the problem formulation. Section III details our hotspot detection framework. Section IV demonstrates the extension on multilayer hotspot patterns and double/multiple patterning. Section V lists experimental results. Finally, Section VI concludes this paper.

II. Problem Formulation

As mentioned in Section I, hotspot detection is an essential task in physical verification. We follow the problem formulation provided by 2012 CAD contest at ICCAD in fuzzy pattern matching for physical verification [16]. For clarity, we have the following definition.

A. Hotspot Detection Problem

Given a training data set of hotspot and nonhotspot patterns and a testing layout, identify hotspots in the testing layout. The primary objective is to maximize the accuracy, while the secondary objective is to minimize the false alarm.

Definition 1: A hotspot is a layout pattern that is susceptible to lithographic process and may induce the printability issue at the fabrication stage.

Definition 2: A hit is a correctly identified hotspot. Accuracy is the ratio of the number of total hits over the number of all actual hotspots.

Definition 3: An extra is a nonhotspot that is identified as a hotspot. The false alarm is the ratio of the number of total extras over the testing layout area.

Since one hotspot can kill a design, the primary objective is accuracy. The false alarm defined here represents the false positives (how many nonhotspots are reported as hotspots),

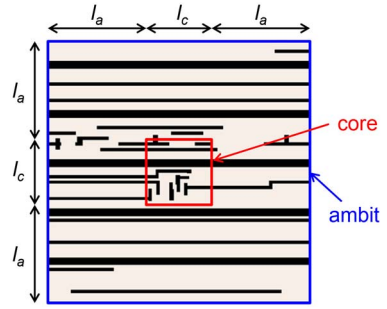

Fig. 1. Hotspot or nonhotspot pattern [16].

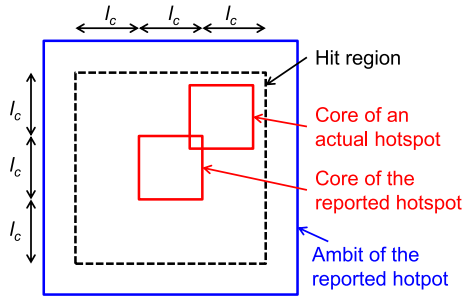

Fig. 2. Hit.

while "1.0—accuracy" reflects the false negatives (how many hotspots are missed). Based on the problem formulation, we shall maximize the number of hits first and then maximize the hit/extra rate.

As shown in Fig. 1, a hotspot or nonhotspot pattern in the training data set is a layout clip defined by a core and its ambit, where the core is the central part of this clip providing its significant characteristics, while the ambit is the peripheral part of this clip providing supplementary information. The training data set, provided by foundry (or lithography simulation), is highly imbalanced, i.e., the nonhotspot patterns greatly outnumber the hotspot patterns.

As shown in Fig. 2, a reported hotspot is considered as a hit if the core of the reported hotspot overlaps with the core of an actual hotspot. For a hit, the clip of the reported hotspot fully covers the core of an actual hotspot, and two clips overlap at least a certain amount of area.

III. Our Approach

In this section, we detail our hotspot detection framework.

A. Overview

Fig. 3 shows the overview of our machine-learning-based hotspot detection framework. To fully exploit the strengths of machine learning, we introduce adequate fuzziness (Section III-D3), rebalance hotspot and nonhotspot population (Section III-D3), reduce false alarm by feedback learning and redundant clip removal (Sections III-D4 and III-F), and develop an efficient evaluation scheme (Sections III-E and III-F).

In the training phase, our goal is to achieve a good training quality but maintain a high flexibility. First of all, we classify the known hotspot and nonhotspot patterns into clusters according to the topologies in their core regions.

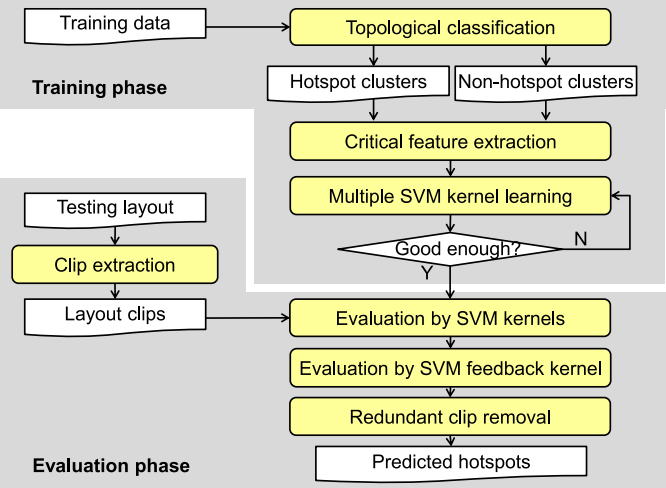

Fig. 3. Our hotspot detection framework.

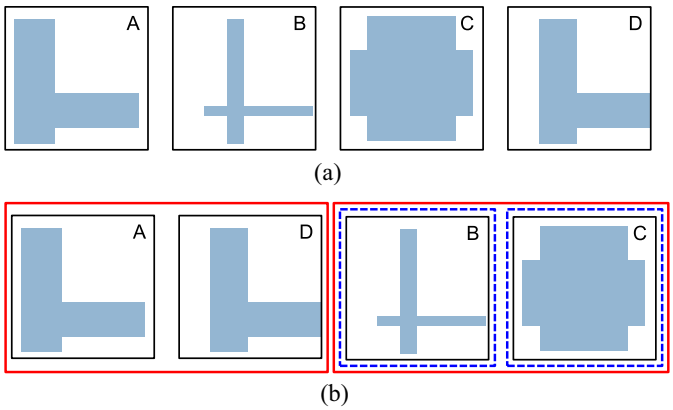

Fig. 4. Topological classification. (a) Core regions of four patterns A, B, C, and D. (b) Generated clusters: $\{A, D\}$, $\{B\}$, and $\{C\}$.

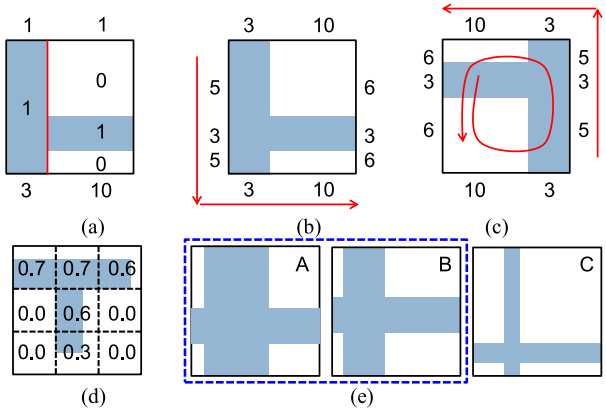

Fig. 5. Topological classification. (a)–(c) String-based classification. (d) and (e) Density-based classification.

Secondly, topological and nontopological critical features of core regions are extracted from each cluster. Thirdly, based on iterative multiple kernel learning, we construct a specific SVM kernel for each cluster. Finally, after the self-evaluation process, we collect the miss-predicted nonhotspot clips to train an additional kernel—feedback kernel—using the features in the ambit regions. With topological classification, each kernel can concentrate on the critical features specific to its corresponding cluster as well as provide a flexibility to identify previously unseen patterns. Topological classification also facilitates hotspot and nonhotspot population balancing. Compared with a single huge SVM kernel, our multiple kernel learning achieves high accuracy. On the other hand, in the evaluation phase, to avoid time-consuming full-layout scanning, we extract only possible layout clips based on the polygon density. During evaluation, a feedback kernel evaluation process is applied after normal evaluation process to reduce miss-predicted nonhotspot clips. After evaluation, we further filter the detected hotspots to reduce redundant hotspot clips. Our redundant clip removal can greatly reduce the false alarm without sacrificing the accuracy.

B. Topological Classification

We observe that some training patterns have similar shapes and some are quite different. Hence, to facilitate the subsequent machine-learning kernel training, hotspot/nonhotspot patterns in the training data set are classified into clusters based on the topologies of their core regions. After topology classification, the patterns within one cluster have very similar geometrical characteristics (critical topological features) in their core regions.

We devise two-level topological classification: string-based and density-based classification. For example, consider the core regions of four patterns A, B, C, D listed in Fig. 4(a). String-based classification first splits these core patterns into two clusters $\{A, D\}$ and $\{B, C\}$ based on topology (B and C are both crosses), and density-based classification further divides $\{B, C\}$ into clusters $\{B\}$ and $\{C\}$ based on polygon distribution. Topological classification makes each machine-learning kernel concentrate on the critical features specific to its corresponding cluster as well as facilitates hotspot and nonhotspot population balancing (see Section III-D3).

1) String-Based Classification: By extending the string representation used by [4], we propose four directional strings to capture the topology of the core region of one pattern. To generate the string for the downward direction, the core pattern is first vertically sliced along polygon edges, e.g., two slices are generated for Fig. 5(a). For each slice, the boundary is labeled as "1," a polygon block is labeled as "1," and a space block is labeled as "0." Thus, each slice corresponds to a binary sequence, and then this sequence is converted to a decimal number (or a radix number that is most suitable for the current case). The downward slicing of Fig. 5(a) generates a string, <3, 10> (= $<11_2, 1010_2>$), recorded at the bottom side. Similarly, we have the other three strings recorded at right, top, left sides [see Fig. 5(b)]. Any two strings recorded at adjacent sides can fully capture the topology of a core pattern.

We verify whether two core patterns have the same topology as follows. The four directional strings are generated for these two core patterns. Any two strings of adjacent sides of one core pattern are selected. Two composite strings are generated by concatenating the strings of the other core pattern counterclockwise and clockwise. (The string of the beginning side should be added at the end.) String matching is performed to

verify if they have the same topology. We have the following theorem.

Theorem 1: Considering eight possible orientations,[1] two core patterns have the same topology if and only if any two strings at adjacent sides of one core pattern exist in the counterclockwise or clockwise composite string of the other core pattern.

For example, we select two strings at adjacent sides of the core pattern in Fig. 5(b), say <5, 3, 5, 3, 10> from the left and bottom sides. We have the composite strings of the core pattern in Fig. 5(c) as follows:

Counterclockwise: <6, 3, 6, 10, 3, 5, 3, 5, 3, 10, 6, 3, 6>, and Clockwise: <10, 3, 5, 3, 5, 3, 10, 6, 3, 6, 10, 3>.

<5, 3, 5, 3, 10> exists in the counterclockwise composite string. Hence, two core patterns given in Fig. 5(b) and (c) have the same topology. Theorem 1 guarantees that the clusters generated by string-based classification are unique.

2) Density-Based Classification: After string-based classification, the patterns within one cluster have the same topology in their core regions. Even so, in some cases, two core patterns with the same topology may still have very different geometrical characteristics. For example, one could be a hotspot, while the other is a nonhotspot under discrete process forbidden rules.

A core pattern p_i is first pixelated, and the polygon density of each pixel d_k is calculated [9] [see Fig. 5(d)]. The distance $\tilde{n}(p_i, p_j)$ between two core patterns p_i and p_j is defined by the summation of the pixel density difference over all pixels based on the same orientation

$$\rho(p_i, p_j) = \min_{\tau \epsilon D_8} \sum_k \left| d_k(p_i) - d_k\left(\tau \ \ (p_j)\right) \right| \qquad (1)$$

where τ is the orientation and D_8 represents the set of eight possible orientations. Based on the distance metric, the cluster radius used by density-based classification is defined as follows:

$$R = \max\left(R_0, \ \max_{i,j} \rho(p_i, p_j) \ / K\right) \qquad (2)$$

where R_0 is the user-defined radius threshold and K is a user-defined expected cluster count. We adopt an efficient yet effective clustering method: for an investigated pattern, we check whether this pattern is covered by some existing cluster. [A pattern is covered by a cluster if the distance between this pattern and the centroid (representative) of the cluster is less than or equal to the radius value.] If so, the pattern is added into the covering cluster. Otherwise, this pattern becomes the centroid of a new cluster. This flow is repeated for all patterns. In addition, the clustering process can be further improved by recalculating the centroid once a pattern is added to some cluster. As shown in Fig. 5(e), two clusters, $\{A, B\}$ and $\{C\}$, are generated by density-based classification.

C. Critical Feature Extraction

We extract topological (geometry-related) and nontopological (lithography-process-related) critical features of each core pattern.

[1]Eight orientations include combinations of four rotations (0°, 90°, 180°, and 270°) and two mirrors (horizontal and vertical mirrors).

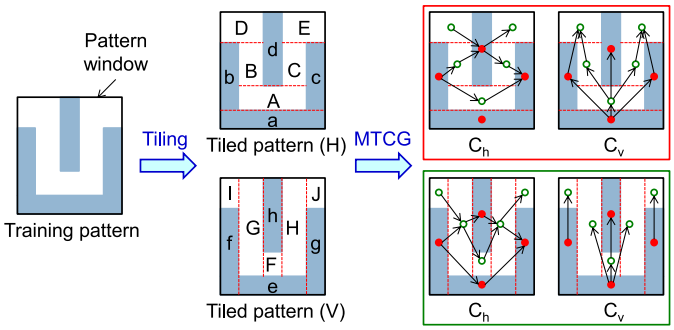

Fig. 6. From tiled pattern to MTCG [6].

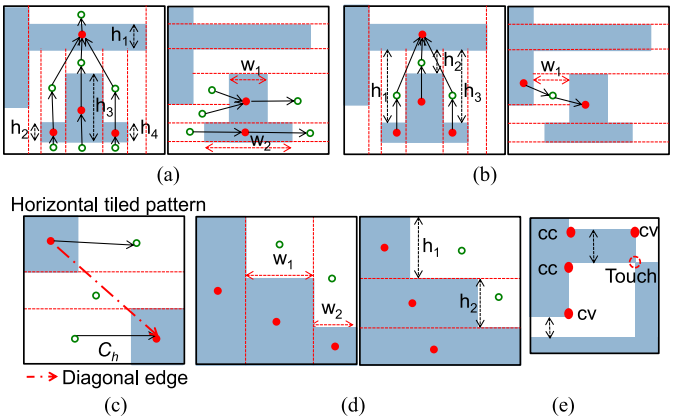

Fig. 7. Critical features. (a) Internal feature. (b) External feature. (c) Diagonal feature. (d) Segment feature. (e) Nontopological feature.

We adopt MTCG proposed by [6] to extract the topological critical features. As shown in Fig. 6, we firstly tile the core region of a given training pattern horizontally and vertically to capture the topological information among polygons. Next, we convert the horizontally (vertically) tiled core pattern to a horizontally (vertically) tiled MTCG. In an MTCG, each vertex represents a block tile (dots) or a space tile (circles), while each edge represents some topological relation among tiles.

MTCGs can be constructed by the sweep-line algorithm. In the vertical constraint graph C_v, a directed edge is added between any two adjacent tiles if their projections on x-axis overlap. Similarly, in the horizontal constraint graph C_h, a directed edge is added between any two adjacent tiles if their projections on y-axis overlap. Moreover, only in horizontally tiled horizontal constraint graph C_h, a diagonal directed edge is added between any two adjacent block tiles (space tiles) if their projections on the y-axis do not overlap, where adjacency means there are no other block tiles (space tiles) in between the corner region formed by these two block tiles (space tiles).

All critical features of a given core pattern will be extracted only from horizontally tiled C_h graph and vertically tiled C_v graph. The rest of the graphs are used for checking window boundaries. As shown in Fig. 7(a)–(d), four types of topological features are extracted.

1) Internal Feature—The Width and Height of a Block Tile: As shown in Fig. 7(a), we find the dimension of each block tile with at most one edge touching the window

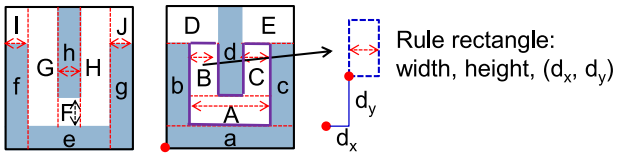

Fig. 8. Critical feature extraction.

boundaries. Given an MTCG, we extract all block vertices whose incoming and outgoing edges are connected to space vertices.

2) *External Feature—The Distance Between Two Adjacent Block Tiles:* As shown in Fig. 7(b), we find the dimensions of all space tiles that are located in between block tiles and with at most one edge touching the window boundaries. Given an MTCG, we extract any space vertex which lies in between exactly two block vertices.
3) *Diagonal Feature—The Diagonal Relations (or Distance) Between Two Convex Corners of Block Tiles (Space Tiles):* As shown in Fig. 7(c), a diagonal relation among adjacent block tiles (space tiles) can be directed extracted if there is a diagonal edge.
4) *Segment Feature—The Space Tile With Two or Three Edges Touching the Window Boundary or Space Tiles:* As shown in Fig. 7(d), we extract the dimensions of space boundary tiles.

Fig. 8 shows the result of pattern "mountain" after topological feature extraction. Each extracted feature was marked by a double-headed arrow. The internal feature was extracted from tile "*h*." The external features were extracted from tiles "*A*," "*B*," "*C*," and "*F*." The segment features were extracted from the downward side of tiles "*I*" and "*J*." Totally, seven features were extracted.

Next, we use a rule rectangle to record each topological feature as follows. Given a training pattern, a reference point is set to the bottom-left corner of its pattern window. Each extracted topological feature is modeled as a rule rectangle: a rule rectangle is associated with a width, a height, the relative distance (d_x, d_y) between the reference point, and the bottom-left corner of this rectangle. As shown in Fig. 8, the spacing between polygons *b* and *d* is extracted based on external feature and is recorded by a rule rectangle. We introduce a special mark for each feature that touches the window boundaries. Considering eight possible orientations, eight sets of topological features are generated for a given training data.

On the other hand, we define five types of nontopological features as shown in Fig. 7(e): 1) the number of corners (convex plus concave); 2) the number of touched points; 3) the minimum distance between a pair of internally facing polygon edges; 4) the minimum distance between a pair of externally facing polygon edges; and 5) the polygon density.

By topological classification, the number of critical features is identical for all patterns in a cluster. The equivalent feature number facilitates the subsequent SVM kernel training.

The goal of critical feature extraction is to use the least features to express a given hotspot. Because a hotspot can be formed by variant and divergent layout configurations, a conventional way is to extract all distances among edge pairs of a given hotspot pattern. However, doing so may generate numerous design rules and lower our training speed. Different from the conventional way, our defined features and extraction method can efficiently extract the most fundamental and critical features of a given pattern. By using rule rectangles and relative distances as depicted in Fig. 8. We can naturally form complex pattern geometries. For example, the "*U*" or double "*L*" shape can be composed by rule rectangles *A*, *B*, and *C*.

D. Iterative Multiple SVM-Kernel and Feedback Kernel Learning

To provide the flexibility to identify unseen hotspots, we leverage on machine learning. We devise iterative multiple SVM kernel and feedback kernel learning to fully exploit the strengths of machine learning.

1) C-SVM With Radial Basis Kernel Function: In machine learning, SVM is a popular supervised learning model. A two-class SVM transforms the training data to a high-dimensional space and calculates a hyperplane to separate the data into two classes with a maximum margin. If the SVM kernel function is a symmetric positive semi-definite function, then SVM guarantees a global optimum solution. SVM has showed superior performance in handling a small training data set, nonlinear, and high dimensional classification issues. Based on this evidence, in this paper, we use two-class soft-margin *C*-type SVM and adopt the radial basis function as our kernel to detect hotspots and nonhotspots [17], [18]. Given training data $x_n, n = 1 \ldots N$, with label t_n ($+1$ or -1 for two-class SVM). The dual form of the quadratic programming formulation of *C*-type SVM is given as follows:

$$\begin{aligned} \max\ & f(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m) \\ \text{subject to}\ & 0 \le a_n \le C, \forall n = 1 \ldots N \\ & \sum_{n=1}^{N} a_n t_n = 0 \\ & k(x_n, x_m) = \exp\left(-\tilde{a}\|x_n - x_m\|^2\right) \\ & a = (a_1, \ldots, a_N)^{\mathrm{T}} \end{aligned} \tag{3}$$

where *C* controls the trade-off between the slack variable penalty and the margin, $k(x_n, x_m)$ is the Gaussian radial basis kernel function, and a_n is the Lagrange multiplier. The Gaussian radial basis kernel function is symmetric positive semidefinite thus leading to an optimal classification. Imbalanced population may destroy the soft margin and degrade the training quality; we shall balance the population in Section III-D3.

2) Iterative Learning: Appropriate values of *C* and $\tilde{a}$ may result in a good training quality of an SVM kernel. Therefore, as shown in Fig. 3, we introduce a self-training process to iteratively adapt *C* and $\tilde{a}$ parameters. In our experiments, the initial values of *C* and γ are 1000 and 0.01, respectively. *C* and γ are doubled if the stopping criterion is not satisfied. The stopping criterion of iterative learning is that the number of self-training iterations exceeds a user-defined bound or the hotspot/nonhotspot detection accuracy rate (with respect

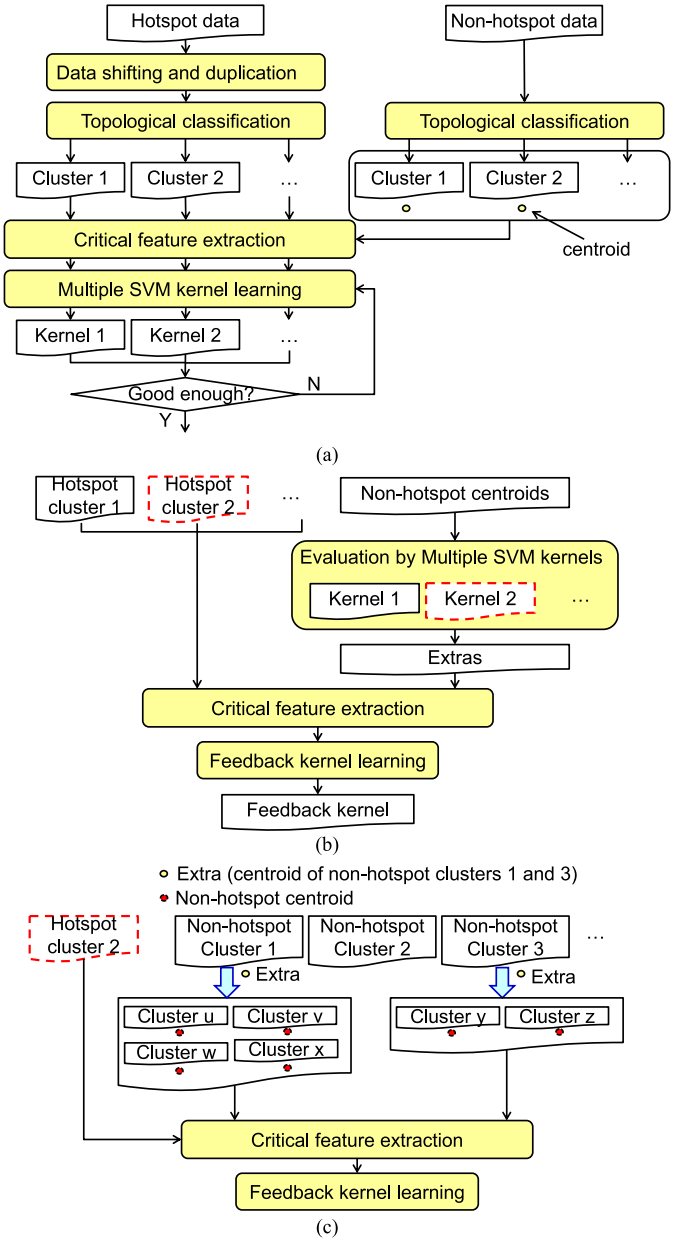

Fig. 9. Training phase. (a) Multiple SVM-kernel learning. (b) Feedback kernel learning. (c) Details of our feedback kernel learning.

to the training data) exceeds a user-defined training accuracy, say 90% in our experiments.

3) Population Balancing and Multiple Kernels: Fig. 9 details our training phase, including topological classification, population balancing, multiple SVM kernel learning, and feedback kernel learning steps.

Redundant nonhotspot training patterns and imbalanced population between hotspot and nonhotspot samples may degrade the accuracy and increase the false alarm. Consequently, to balance population, we upsample hotspot training patterns and downsample nonhotspot training patterns. To upsample hotspot training patterns and compensate the layout clip extraction error (Section III-E), we slightly shift each hotspot training pattern upward, downward, leftward, and rightward to create several derivatives before topological classification. Multiple hotspot and nonhotspot clusters are generated after topological classification. To filter out redundant nonhotspot training patterns (downsample), only the centroid of each nonhotspot cluster is selected. Our nonhotspot training data set is formed by all nonhotspot centroids. It is clear that resampling can effectively balance hotspot and nonhotspot population, and eliminate noises contributed by nonhotspots.

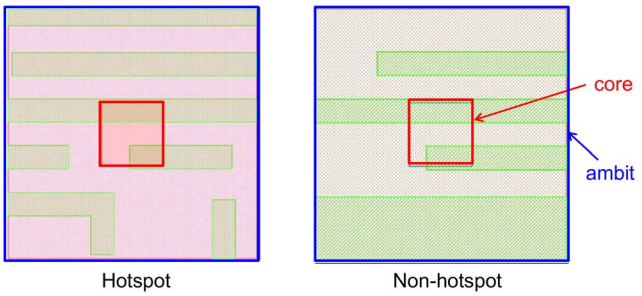

Fig. 10. Hotspot clip and a nonhotspot clip contain an almost identical core pattern.

Since each hotspot cluster has its own characteristics, we construct one SVM kernel to learn the specific critical features of each hotspot cluster. Each SVM kernel is constructed based on one hotspot cluster and the newly formed nonhotspot training data set. We then have multiple SVM kernels. For example, to train SVM kernel 2 shown in Fig. 9(a), we consider hotspots contained in cluster 2 plus all nonhotspot centroids as our training data.

Because the training set of each SVM kernel is much smaller than the original one, we can have a shorter training time after population balancing. Assume that it takes $O((n_1+n_2)^2)$ time to train an SVM kernel with a training set of n_1 hotspot patterns and n_2 nonhotspot patterns. After topological classification, m_1 hotspot clusters and m_2 nonhotspot clusters are generated, and the average set size is $n_1/m_1 + m_2$, the total training time of our method will be $O(m_1(n_1/m_1+m_2)^2)$, typically much smaller than $O((n_1+n_2)^2)$.

4) Feedback Kernel: Here, we present our feedback kernel, which is used to reduce the false alarm (the extra count) while maintaining the same accuracy (the hit rate). The feedback kernel is applied after multiple SVM kernels to reclaim an extra (see Definition 3) to be a nonhotpot.

If we perform self-evaluation again after the iterative learning process, one might notice that some nonhotspot centroids (training data) may still be classified as hotspots. The main reason is that in the training process of multiple SVM kernels, only the core region is considered during critical feature extraction and multiple SVM kernel learning.

In most cases, considering only the core region is sufficient to distinguish a hotspot clip from a clip set. In fact, the pattern within the ambit region may also affect the core region during the lithography process. Fig. 10 shows an example where both hotspot and nonhotspot clips contain an almost identical core pattern. The only way to classify them is to take the ambit region into consideration.

Therefore, to train our feedback kernel, as shown in Fig. 9(b), we firstly apply the nonhotspot centroids

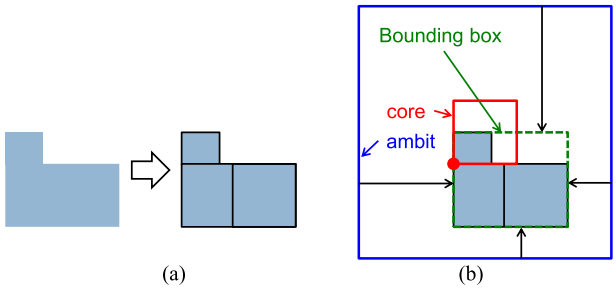

Fig. 11. Layout clip extraction. (a) Polygon dissection. (b) Layout clip.

[the same data used in Fig. 9(a)] to our SVM kernels. Second, we form our nonhotspot training data set by using extras (see Definition 3) classified by one kernel (kernel 2 in this example) after self-evaluation; we form our hotspot training data set by using only the hotspots from the corresponding hotspot cluster. As shown in Fig. 9(c), we assume that the centroids of nonhotspot clusters 1 and 3 are extras reported by kernel 2. To form our nonhotspot training data set, with the ambit information, we apply topological classification on nonhotspot clusters 1 and 3 to form new sub-clusters u, v, w, x, y, and z. These sub-clusters contain similar topologies within the core regions, but different in the ambit regions. Next, we generate our new nonhotspot training set by using centroids contributed by clusters u, v, w, x, y, and z. The hotspot training set is formed by hotspot cluster 2.

It can be seen that, except the investigated kernel, all of other kernels acquire enough training to correctly classify these nonhotspot centroids. Thus, we only need to enhance the training data provided to the investigated kernel. During the critical feature extraction and the learning process, both of the core and ambit regions of a given training pattern are considered. In addition, if the extras are contributed by more than one kernel, then the hotspots of both kernels will be treated as hotspot training data in feedback kernel learning.

At the evaluation phase, a layout clip is flagged as a hotspot if one kernel classifies it as a hotspot during the multiple kernel evaluation process. This mechanism makes each kernel focus on the critical features specific to its own hotspot cluster thus achieving a good training quality. Next, these flagged hotspots are applied to the feedback kernel (which focuses on the miss-predicted nonhotspot centroids). With considering the ambit region, the false alarm can be reduced.

Moreover, data shifting generates several derivatives from a training pattern thus introducing adequate fuzziness into each cluster. Hence, our kernels have a flexibility to identify previously unseen patterns. Later, our results show that compared with a single huge SVM kernel, our multiple kernel learning achieves high accuracy.

E. Layout Clip Extraction

A testing layout contains tremendous sites that need to be evaluated. To avoid time-consuming full-layout scanning, we extract only possible layout clips based on the polygon distribution.

As shown in Fig. 11(a), each layout polygon is first horizontally sliced into rectangles. These rectangles are then cut into smaller pieces if their widths or heights are greater than the hotspot core side length (l_c in Fig. 2).

As shown in Fig. 11(b), a core and its ambit are set with respect to the bottom left corner of each rectangle. The corresponding layout clip is extracted if the polygon distribution within this clip meets the user-specified requirements; otherwise, the clip is discarded. The polygon distribution here means the polygon density, the polygon count, and the distances between the clip boundary and the bounding box that covers all polygon rectangles in the clip [indicated by four arrows in Fig. 11(b)]. The user-defined requirements are positively correlated to the critical features extracted for SVM kernels. It can be seen that if the polygon distribution requirements are met, each polygon must be included by at least one layout clip. Moreover, the possible misalignment between an extracted clip and an actual hotspot can be compensated by data shifting described in Section III-D3. Later, in Section V, experimental results will show that the size of our extracted layout clips is much smaller than the window-based method. The evaluation time is thus greatly reduced.

F. Redundant Clip Removal

After layout clip extraction, clip cores may strongly overlap in an area with high polygon or corner density. Those redundant cores can be eliminated or merged, and then be presented by fewer clip cores. As shown in Fig. 12(a), after SVM kernel evaluation, reported hotspot cores strongly overlap in an area with high polygon density. These strongly overlapped reported cores actually target to the same core pattern. We devise redundant clip removal to reduce the redundancy. Moreover, our redundant clip removal can greatly reduce the false alarm without sacrificing the accuracy.

First of all, we merge reported hotspot cores into several regions; we merge a hotspot clip into an existing merging region if its core overlaps with some hotspot core of the region. A merging region is the minimum bounding box covering all hotspot cores in this region [see Fig. 12(b)].

Secondly, a merging region containing more than four hotspot cores is reframed. The goal of reframing is to minimize the number of reported hotspots without missing any possible actual hotspots. Fig. 12(c) shows the clip reframing, where the distance l_s between two reframed cores should be less than the core side length l_c. Our clip reframing guarantees that the core of an arbitrary actual hotspot is overlapped by at least one reframed core. Moreover, we further remove redundant clips located in the overlapping area of two merging regions. A hotspot core is discarded under two conditions: 1) all polygons within this core are covered by other hotspot cores and 2) each corner of this core overlaps with other hotspot cores inside some merging region [see Fig. 12(d)]. Thirdly, we check the distances between the clip boundary and the bounding box that covers all polygon rectangles in the clip. If the distance is greater than the user-specified value, the clip's center is shifted to the polygon's center of gravity along x-axis or y-axis. Finally, we apply the merging and reframing process again. Fig. 12(e) shows the redundant clip removal result of Fig. 12(a).

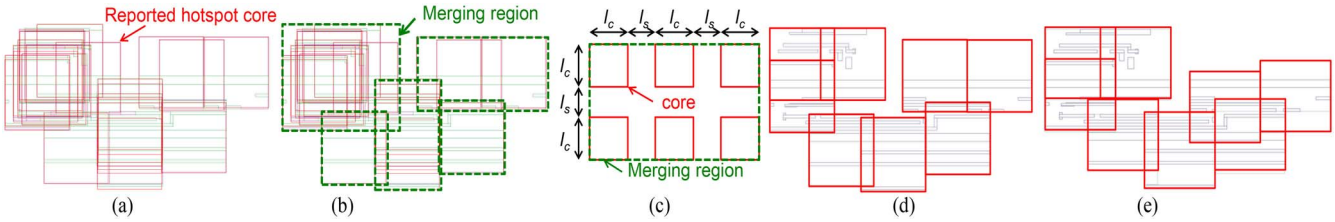

Fig. 12. Redundant clip removal. (a) Hotspots reported by our SVM kernel. (b) Clip merging of (a). (c) Clip reframing. (d) Clip reframing of (b). (e) Clip shifting of (d).

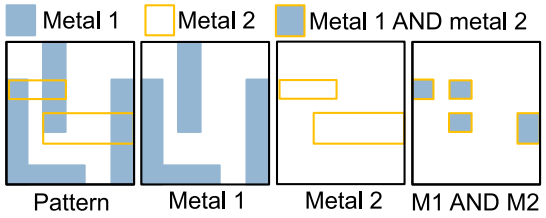

Fig. 13. Multilayer pattern handling.

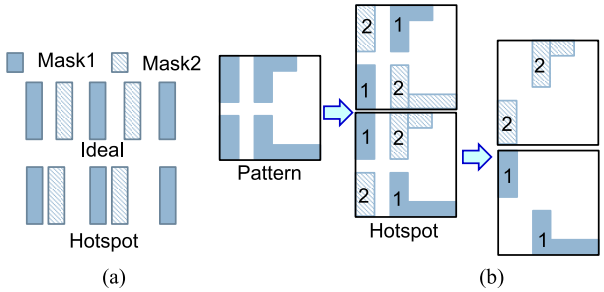

Fig. 14. (a) Hotspots caused by mask misalignment. (b) Three sets of hotspot features.

G. Multithreaded Computing

Multithreaded computing can be easily applied to our framework. In the training phase, as shown in Fig. 9(a), all kernels can be trained simultaneously because of their independency. In the clip extraction process, a testing layout can be easily cut apart into smaller pieces (with a small amount of overlap). We can generate layout clips from these pieces at the same time.

The training phase and the clip extraction process are mutually independent. Thus, we can execute them in parallel. For a small testing layout, the training phase dominates the execution time. When a testing layout is larger, the clip extraction might be the bottleneck of the whole process.

IV. Extension

In this section, we discuss how to extend our hotspot detection framework to handle multilayer hotspot detection and double (multiple) patterning.

A. Multilayer Hotspot Detection

Our proposed method can be extended to handle multiple layers. In a real design, hotspots can also be formed by layout patterns on multiple metal layers.

Given a set of training data with m layers, we do topological classification first on one randomly selected layer. Then, for each of the newly generated training data, we extract: 1) m sets of pattern features from each metal layer and 2) $m - 1$ sets of pattern features from the overlapped polygons of adjacent metal layers. Fig. 13 shows a training pattern with two metal layers. Three sets of features will be extracted, one from metal 1 layer, another from metal 2 layer, and the other from the overlapped polygons of metal 1 and 2 layers. Only diagonal and internal features are extracted from the overlapped polygons.

During our clip extraction process, we do our extraction on the same layer as topological classification. The rest of the process will be the same as single-layer hotspot detection.

B. Double (or Multiple) Patterning

In double patterning, a hotspot may be induced by the misalignment between two masks. As shown in Fig. 14(a), this type of hotspot is unpredictable during the mask alignment process. If a hotspot is irrelevant to the mask misalignment or if the foundry can provide the pattern decomposition information on layout and on training patterns, we can still apply our hotspot detection method. As shown in Fig. 14(b), a given pattern has two decomposition results, one with a higher risk to generate hotspot. To capture the feature of this hotspot pattern, we extract three sets of features. The first set is from mask 1 pattern, the second set is from mask 2 pattern, and the third set is from the given pattern itself. Extracted rules from first and second sets will have mask marks on them. Now, given a to-be-identified layout clip, we can extract its features with this manner.

V. Experimental Results

Our algorithm was implemented in the C++ programming language with a GDSII library Anuvad [19] and the SVM library LIBSVM [20]. We parallelized our framework by using POSIX threads (pthread) [21]. We executed the program on a platform with two Intel Xeon 2.3 GHz CPUs and with 64 GB memory. Experiments are conducted on six 32 and 28 nm industrial designs released by [16] as listed in Table I, with a highly imbalanced population between hotspot and non-hotspot training patterns. We do four sets of experiments to compare the overall performance with state-of-the-art works, demonstrate the effectiveness of our multiple SVM kernel training, redundant clip removal, clip extraction, feedback kernel, and show our rapid training convergence.

In the first set of experiments, we compare our approach with 2012 CAD contest at ICCAD winners and [14]. Table II

TABLE I
2012 CAD CONTEST AT ICCAD BENCHMARK STATISTICS

Training data			Testing layout			
Name	#hs	#nhs	Name	#hs	area (um^2)	process
MX_benchmark1_clip	99	340	Array_benchmark1	226	12,516	32nm
MX_benchmark2_clip	176	5,285	Array_benchmark2	499	106,954	28nm
MX_benchmark3_clip	923	4,643	Array_benchmark3	1,847	122,565	28nm
MX_benchmark4_clip	98	4,452	Array_benchmark4	192	82,010	28nm
MX_benchmark5_clip	26	2,716	Array_benchmark5	42	49,583	28nm
			MX_blind_partial	55	224,975	32nm

#hs: number of hotspots; #nhs: number of non-hotspots.
The core size is $1.2\times1.2um^2$, while the clip size is $4.8\times4.8um^2$.

TABLE II
COMPARISON WITH 2012 CAD CONTEST WINNERS AND [14]

Testing layout (Training data)	Methods	#hit	#extra	accuracy	hit/extra	Runtime
Array_benchmark1 (MX_benchmark1_clip)	1st place	212	1,826	93.81%	1.16E-01	0m05.1s
	2nd place	98	188	43.36%	5.21E-01	1m50.2s
	3rd place	157	728	69.47%	2.16E-01	0m06.7s
	[14]	183	3,356	82.10%	5.45E-02	0m14.4s
	Ours	214	1,416	94.69%	1.51E-01	1m01.6s
	Ours_low	129	251	57.08%	5.14E-01	0m50.4s
	Ours_med	184	920	80.70%	2.00E-01	0m53.7s
	Ours_nopara	214	1,416	94.69%	1.51E-01	1m39.6s
Array_benchmark2 (MX_benchmark2_clip)	1st place	489	20,383	98.00%	2.40E-02	8m11.9s
	2nd place	108	548	21.64%	1.97E-01	23m40.8s
	3rd place	337	5,878	67.54%	5.73E-02	6m10.2s
	[14]	385	1,842	75.80%	2.09E-01	3m04.8s
	Ours	490	10,761	98.20%	4.55E-02	1m02.7s
	Ours_low	210	946	42.08%	2.22E-01	0m33.2s
	Ours_med	388	1,693	77.75%	2.29E-01	0m50.4s
	Ours_nopara	490	10,761	98.20%	4.55E-02	4m12.2s
Array_benchmark3 (MX_benchmark3_clip)	1st place	1,696	20,764	91.82%	8.17E-02	18m44.0s
	2nd place	1,491	9,579	80.73%	1.56E-01	118m56.8s
	3rd place	1,840	71,328	99.62%	2.58E-02	7m58.1s
	[14]	1,271	2,407	68.80%	5.03E-01	4m07.0s
	Ours	1,697	13,025	91.88%	1.30E-01	12m24.9s
	Ours_low	1,326	2,377	71.79%	5.58E-01	8m49.9s
	Ours_med	1,657	1,0376	89.71%	1.59E-01	11m16.6s
	Ours_nopara	1,697	13,025	91.88%	1.30E-01	15m22.7s
Array_benchmark4 (MX_benchmark4_clip)	1st place	161	3,726	83.85%	4.32E-02	1m15.9s
	2nd place	124	956	64.58%	1.30E-01	21m57.9s
	3rd place	152	13,582	79.17%	1.14E-02	1m42.9s
	[14]	138	1,488	72.00%	8.46E-02	1m43.3s
	Ours	165	3,437	85.94%	4.80E-02	5m29.1s
	Ours_low	139	1,042	72.39%	1.33E-01	2m00.2s
	Ours_med	150	1,548	78.12%	9.68E-02	3m02.2s
	Ours_nopara	165	3,437	85.94%	4.80E-02	6m49.5s
Array_benchmark5 (MX_benchmark5_clip)	1st place	39	2,014	92.86%	1.94E-02	0m26.6s
	2nd place	26	31	61.90%	8.39E-01	5m25.6s
	3rd place	20	245	47.62%	8.16E-02	0m40.0s
	[14]	28	444	63.40%	6.30E-02	0m44.6s
	Ours	39	1,111	92.86%	3.51E-02	0m07.8s
	Ours_low	29	192	69.04%	1.51E-01	0m06.0s
	Ours_med	35	413	83.33%	8.47E-02	0m06.6s
	Ours_nopara	39	1,111	92.86%	3.51E-02	0m21.7s

1st place is executed on a platform with 2 Intel Xeon 2.3 GHz CPUs and with 64 GB memory, while 2nd and 3rd places are executed on 4 Intel Xeon 2.0 GHz CPUs and with 72 GB memory. We directly quote the experimental results of [14]. [14] is executed on 2.66 GHz quad-core CPU and with 8 GB memory. 'Ours_nopara' means our method without multi-threaded computing. 'Ours_med' means our method targeting medium accuracy with medium hit/extra ratio; 'Ours_low' means our method targeting low accuracy with high hit/extra ratio.

summarizes the experimental results. To demonstrate the effectiveness of our approach under different accuracy targets, "ours" means our proposed method, "ours_low" means our method targeting low hit rate but high hit/extra ratio, while "ours_med" means our method targeting medium hit rate but medium hit/extra ratio. "Ours_nopara" means our method without multithreaded computing. Overall, our approach outperforms these works on accuracy and has good hit/extra rates. For array_benchmark3, compared with the third place winner, we have lower accuracy but with a significantly lower false alarm. Compared with [14], we can have higher hit rate and better hit/extra ratio for all cases. Moreover, compared with the second and third place winners, on average, ours_med or ours_low can achieve higher hit rate and better hit/extra ratio.

TABLE III
DETAILED COMPARISON ON OUR FEATURES

Benchmark	Methods	#hs/#nhs	#hit	#extra	accuracy	Runtime
Array_benchmark1 (MX_benchmark1_clip)	1st place	-	212	1,826	93.81%	0m05.1s
	Basic	0.29	164	1,126	72.57%	0m02.7s
	+Topology	0.79	214	2,259	94.69%	0m56.4s
	+Removal	0.79	214	1,491	94.69%	0m57.3s
	Ours	0.79	214	1,416	94.69%	1m01.6s
Array_benchmark2 (MX_benchmark2_clip)	1st place	-	489	20,383	98.00%	8m11.9s
	Basic	0.03	288	2,828	57.72%	3m42.8s
	+Topology	0.08	490	24,629	98.20%	0m55.0s
	+Removal	0.08	490	11,826	98.20%	1m00.4s
	Ours	0.08	490	10,761	98.20%	1m02.7s
Array_benchmark3 (MX_benchmark3_clip)	1st place	-	1,696	20,764	91.82%	18m44.0s
	Basic	0.19	1,600	31,811	86.63%	7m42.8s
	+Topology	0.46	1,697	55,758	91.88%	10m58.4s
	+Removal	0.46	1,697	13,774	91.88%	12m00.9s
	Ours	0.46	1,697	13,025	91.88%	12m24.9s
Array_benchmark4 (MX_benchmark4_clip)	1st place	-	161	3,726	83.85%	1m15.9s
	Basic	0.02	119	1,388	61.98%	0m26.7s
	+Topology	0.08	165	4,879	85.94%	5m09.3s
	+Removal	0.08	165	3,651	85.94%	5m12.1s
	Ours	0.08	165	3,437	85.94%	5m29.1s
Array_benchmark5 (MX_benchmark5_clip)	1st place	-	39	2,014	92.86%	0m26.6s
	Basic	0.01	33	1,227	78.57%	0m13.0s
	+Topology	0.05	39	2,332	92.86%	0m06.5s
	+Removal	0.05	39	1,193	92.86%	0m07.6s
	Ours	0.05	39	1,111	92.86%	0m07.8s
MX_blind_partial (MX_benchmark1_clip)	1st place	-	51	66,818	92.73%	2m31.7s
	Basic	0.29	38	31,148	69.09%	1m18.1s
	+Topology	0.79	51	79,221	92.73%	1m06.3s
	+Removal	0.79	51	54,979	92.73%	1m59.7s
	Ours	0.79	51	49,343	92.73%	2m10.6s

In the second set of experiments, as listed in Table III, we demonstrate the effectiveness of our approach. "Basic" means the baseline SVM which uses one single huge SVM kernel (i.e., without topological classification, feedback kernel, and redundant clip removal); "+topology" means the baseline with topological classification; "+removal" means the baseline with topological classification and redundant clip removal; ours means our whole framework (i.e., the baseline with topological classification, feedback kernel, and redundant clip removal). In our experiments, we use the following parameters to demonstrate our flow: the respective initial values of C and γ of our SVM kernel are 1000 and 0.01, the expected cluster count is 10, the stopping criterion of self-training is 90% accuracy, data shifting is 120 nm ($= l_c/10$), the maximum distance between the clip boundary and the bounding box of clip extraction is 1440 nm, the minimum overlapping of clip merging is 20%, and the separating distance of core reframing is 1150 nm. First of all, our critical features are effective. For example, single SVM achieves over 78% accuracy for array_benchmark3 and array_benchmark5. Secondly, our topological classification and population balancing indeed works well, and thus our multiple SVM kernel learning has adequate

TABLE IV
ACCURACY AND TRAINING DATA

Benchmark	Methods	Data	#hit	#extra	accuracy	Runtime
Array_benchmark1	1st place	100.0%	212	1,826	93.81%	0m05.1s
(MX_benchmark1_clip)	Ours	65.0%	214	1,367	94.69%	1m57.5s
Array_benchmark2	1st place	100.0%	489	20,383	98.00%	8m11.9s
(Array_benchmark3,4)	Ours	0.6%	494	16,479	99.00%	1m09.3s
Array_benchmark3	1st place	100.0%	1,696	20,764	91.82%	18m44.0s
(MX_benchmark3_clip)	Ours	1.0%	1,712	15,613	92.69%	2m36.2s
Array_benchmark4	1st place	100.0%	161	3,726	83.85%	1m15.9s
(MX_benchmark4_clip)	Ours	99.0%	164	2,599	85.42%	6m54.1s
Array_benchmark5	1st place	100.0%	39	2,014	92.86%	0m26.6s
(MX_benchmark5_clip)	Ours	92.0%	40	1,196	95.24%	0m20.3s
MX_blind_partial	1st place	100.0%	50	49,223	90.91%	15m04.9s
(Array_benchmark3)	Ours	100.0%	52	40,913	94.55%	19m38.1s

TABLE V
CLIP EXTRACTION

Testing layout	Area	#clip Window-based	#clip Ours
Array_benchmark1	0.110mm×0.115mm	34,953	12,602
Array_benchmark2	0.327mm×0.327mm	295,936	182,535
Array_benchmark3	0.350mm×0.350mm	338,724	207,096
Array_benchmark4	0.286mm×0.286mm	226,576	32,204
Array_benchmark5	0.222mm×0.222mm	136,900	20,210
MX_blind_partial	0.750mm×0.299mm	623,251	284,515

Window-based clip extraction method is at 50% window overlap. The window size is $1.2\times1.2\text{um}^2$.

fuzziness and delivers very high accuracy, $85.9 \sim 98.2\%$. Thirdly, our feedback kernel evaluation and redundant clip removal process can further reduce the false alarm for all cases without sacrificing accuracy.

In the third set of experiments, we show the impact of training data on accuracy as listed in Table IV. "Data" means the ratio of the used training pattern count over the whole training pattern count. It can be seen that using different training data may achieve higher accuracy and lower false alarm, e.g., array_benchmark2 and MX_blind_partial. Secondly, we have a rapid convergence on our training quality. We may use a small amount of training data to achieve high accuracy, especially for array_benchmark3 and array_benchmark5, thus shortening the runtime.

In the fourth set of experiments, we compare our clip extraction method with a window sliding method as shown in Table V. The window-based clip extraction method has 50% overlap between two adjacent windows. It can be seen that our method can extract much fewer clips in all cases thus leading to a faster running time during the evaluation phase.

Fig. 15 shows the tradeoff between accuracy (the hit rate) and false alarm. Here, we combine all MX benchmarks and randomly select 5% data as the training data set. We also combine all array benchmarks and MX_blind_partial benchmark together, and then randomly select 5% data as the testing layout. Our primary objective in this paper is to maximize the hit rate. Although a high hit rate implies a high extra count, but it can be seen that the extra count of our method grows only linearly when the hit rate exceeds 90%. In addition, if 80% is an acceptable hit rate, we can keep a very low extra count, and the extra count is very stable when the hit rate ranges from 80% to 84%.

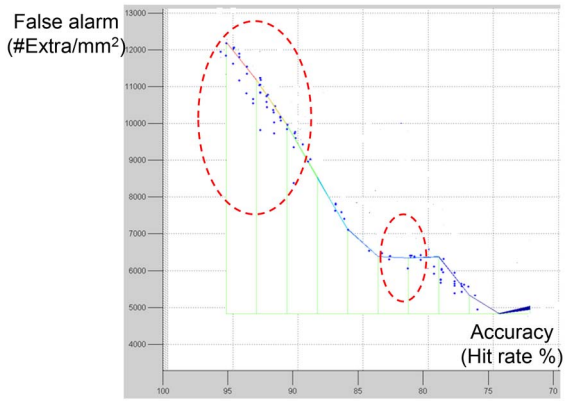

Fig. 15. Trade-offs between accuracy and false alarm.

VI. CONCLUSION

In this paper, we proposed a novel machine-learning-based hotspot detection framework. By combining topological classification and critical feature extraction, our machine-learning kernel achieved high accuracy. Our clip extraction effectively speeded up the evaluation phase. Our feedback learning and redundant clip removal greatly reduced the false alarm. Experimental results showed that our framework not only is very accurate but also has a rapid accuracy convergence using a small amount of training data. Moreover, our framework outperformed the 2012 CAD contest at ICCAD winner on accuracy and false alarm with very competitive runtime.

APPENDIX

Here, we show that two core patterns have the same topology if their strings are matched each other.

Lemma 1: The pattern topology within each vertical slice (or horizontal slice) can be uniquely represented by two binary sequence codes, one in the downward (leftward) side, the other in the upward (rightward) side.

Lemma 1 can be derived from [4].

Theorem 2: Considering eight possible orientations, two core patterns have the same topology if and only if any two strings at adjacent sides of one core pattern exist in the counterclockwise or clockwise composite string of the other core pattern.

Proof: Assume that there can be two or more different topologies map to the same decimal string in the downward side. The same decimal string implies that those different topologies must contain the same vertical slicing number. And at the same slicing location, based on Theorem 1, they must contain the same pattern topology which contradicts our assumption. Now, based on current proof and Theorem 1, we can guarantee that one pattern topology can only be represented by a unique string. If this unique string is found in the counterclockwise or clockwise composite string of the other pattern, then two patterns must have the same topology. ■

REFERENCES

[1] (2013). *International Technology Roadmap for Semiconductors*. [Online]. Available: http://www.itrs.net/reports.html

[2] P. Gupta, A. B. Kahng, S. Nakagawa, S. Shah, and P. Sharma, "Lithography simulation-based full-chip design analyses," *Proc. SPIE*, vol. 6156, Mar. 2006, Art. ID 61560T.
[3] A. B. Kahng, C.-H. Park, and X. Xu, "Fast dual graph based hotspot detection," *Proc. SPIE,* vol. 6349, Oct. 2006, Art. ID 63490H.
[4] H. Yao, S. Sinha, C. Chiang, X. Hong, and Y. Cai, "Efficient process-hotspot detection using range pattern matching," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2006, pp. 625–632.
[5] J. Xu, S. Sinha, and C. Chiang, "Accurate detection for process-hotspots with vias and incomplete specification," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2007, pp. 839–846.
[6] 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. ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Jose, CA, USA, Jun. 2012, pp. 1163–1168.
[7] J. Guo, F. Yang, S. Sinha, and C. Chiang, "Improved tangent space based distance metric for accurate lithographic hotspot classification," in *Proc. ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2012, pp. 1169–1174.
[8] 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 South Pac. Design Autom. Conf. (ASP-DAC)*, Yokohama, Japan, Jan. 2011, pp. 775–780.
[9] J.-Y. Wuu, F. G. Pikus, A. Torres, and M. Marek-Sadowska, "Rapid layout pattern classification," in *Proc. Asia South Pac. Design Autom. Conf. (ASP-DAC)*, Yokohama, Japan, Jan. 2011, pp. 781–786.
[10] J.-Y. Wuu, F. G. Pikus, A. Torres, and M. Marek-Sadowska, "Efficient approach to early detection of lithographic hotspots using machine learning systems and pattern matching," *Proc. SPIE*, vol. 7974, Apr. 2011, Art. ID 79740U.
[11] S. Mostafa, J. A. Torres, P. Rezk, and K. Madkour, "Multi-selection method for physical design verification applications," *Proc. SPIE*, vol. 7974, Apr. 2011, Art ID 797407.
[12] 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 South Pac. Design Autom. Conf. (ASP-DAC)*, Sydney, NSW, Australia, Jan. 2012, pp. 263–270.
[13] 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. ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, Jun. 2013, pp. 472–477.
[14] S.-Y. Lin, J.-Y. Chen, J.-C. Li, W.-Y. Wen, and S.-C. Chang "A novel fuzzy matching model for lithography hotspot detection," in *Proc. ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, Jun. 2013, pp. 478–483.
[15] J.-R. Gao, B. Yu, and D. Z. Pan, "Accurate lithography hotspot detection based on PCA-SVM classifier with hierarchical data clustering," *Proc. SPIE*, vol. 9053, Mar. 2014, Art. ID 90530E.
[16] J. A. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2012, pp. 349–350.
[17] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. Workshop Comput. Learn. Theory (COLT)*, Pittsburgh, PA, USA, 1992, pp. 144–152.
[18] C. Cortes and V. Vapnik, "Support-vector networks," *J. Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.
[19] SoftJin Technologies Pvt. Ltd. (Nov. 2004). *Anuvad: A Free Suite of GDSII and OASIS Libraries.* [Online]. Available: http://www.softjin.com
[20] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Tech. (TIST)*, vol. 2, no. 3, Apr. 2011, Art. ID. 27. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm
[21] D. R. Butenhof, *Programming with POSIX Threads*. Reading, MA USA: Addison-Wesley, 1997.

Yen-Ting Yu received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2005 and 2009, respectively, where he is currently pursuing the Ph.D. degree in electronic design automation.

His current research interests include design for manufacturability and physical design optimization.

Mr. Yu was the recipient of the First Place Award at the 2012 CAD Contest at ICCAD on fuzzy pattern matching.

Geng-He Lin received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2011 and 2013, respectively.

He is currently a Research and Development Engineer with TSMC, Hsinchu. His current research interests include physical design automation on nanometer ICs.

Mr. Lin was the recipient of the First Place Award at the 2012 CAD Contest at ICCAD on fuzzy pattern matching and the Outstanding Graduate Student of NCTU Electrical Computer Engineering College.

Iris Hui-Ru Jiang (M'07) received the B.S. and Ph.D. degrees in electronics engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1995 and 2002, respectively.

From 2002 to 2005, she was with VIA Technologies, Inc., New Taipei, Taiwan. She is currently a Professor with the Department of Electronics Engineering and the Institute of Electronics, NCTU. Her current research interests include physical design optimization and interaction between logic design and physical synthesis. She has published over 70 technical papers and holds 11 patents.

Dr. Jiang was the recipient of the 2011 Chinese Institute of Electrical Engineering Outstanding Young Electrical Engineer Award. She and her students was the recipient of the First Place Award at the 2012 CAD Contest at ICCAD on fuzzy pattern matching, the Third Place Award at the 2013 TAU Variation Aware Timing Analysis Contest, and the Third Place Award at the 2014 TAU Contest on Removing Common Path Pessimism. She has served on several technical program committees of conferences, including ICCAD and Asia and South Pacific Design Automation Conference (ASP-DAC). She is a member of the Phi Tau Phi Scholastic Honor Society.

Charles Chiang (SM'98) received the bachelor's degree from Tunghai University, Taichung, Taiwan, and the Ph.D. degree from the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA, in 1980 and 1991, respectively.

He joined Synopsys, Inc., Mountain View, CA, USA, in 2001. He was at IBM, Endicott, NY, USA, and Rochester, MN, USA, and other electronic design automation companies such as Cadence Design Systems, Inc., San Jose, CA, USA, and Monterey Design Systems, Inc., Mountain View, CA, USA, for ten years. His current research interests include routing, placement, floorplanning, design for manufacturability (DFM), 3-D IC integration, and mask synthesis. He has co-authored a DFM book entitled *Design for Manufacturability and Yield for Nano Scale CMOS* in 2007. He has published over 70 technical papers and holds 33 patents.

Dr. Chiang was the recipient of the 2007 Synopsys Distinguished Inventor Award. He has served on several technical program committee of conferences, including ICCAD and ASP-DAC.