Enabling Online Learning in Lithography Hotspot Detection with Information-Theoretic Feature Optimization *

Hang Zhang , Bei Yu , and Evangeline F. Y. Young

Department of Computer Science and Engineering,
The Chinese University of Hong Kong, NT, Hong Kong
{hzhang,byu,fyyoung}@cse.cuhk.edu.hk

ABSTRACT

With the continuous shrinking of technology nodes, lithography hotspot detection and elimination in the physical verification phase is of great value. Recently machine learning and pattern matching based methods have been extensively studied to overcome runtime overhead problem of expensive full-chip lithography simulation. However, there is still much room for improvement in terms of accuracy and *Overall Detection and Simulation Time (ODST)*. In this paper, we propose a unified machine learning based hotspot detection framework, where feature extraction and optimization is guided by an information-theoretic approach and solved by a dynamic programming model. More importantly, our framework can be naturally extended to online learning scenario, where some newly detected and verified layout patterns are integrated into the learning model. Experimental results show that the proposed batch detection model outperforms all state-of-the-art methods with 3.47% of accuracy improvement and 58.88% of ODST reduction on ICCAD-2012 contest benchmark suite. More importantly, equipped with online learning, our framework can further improve both accuracy and ODST.

1. INTRODUCTIONS

With the continuous shrinking of the transistor feature size, chip manufacturing becomes more and more challenging due to the limitation of conventional 193nm wavelength lithography [1]. Although various design for manufacturability (DFM) techniques have been developed, such as design rule check (DRC), optical proximity correction (OPC), and multiple patterning lithography (MPL), there still exist problematic layout patterns (a.k.a. lithography hotspots) that may cause opens/shorts, performance degradation, or even parametric yield loss [2]. Therefore, how to detect and avoid these lithography hotspots accurately during physical verification phase is critical for yield improving.

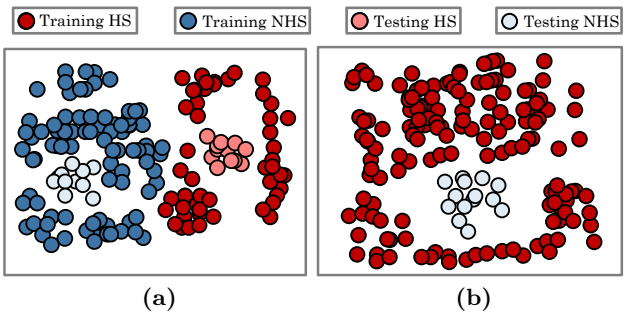

Figure 1: Examples of 2-D feature space of hotspot detection. (a) All testing hotspots and non-hotspots can be correctly detected. (b) All testing non-hotspots become false alarms.

Conventional hotspot detection widely relies on full-chip lithography simulation, which can achieve very high accuracy but may suffer from being extremely computational expensive [3, 4]. To provide quick feedback to circuit designers, recently many fast and coarse-grained hotspot detectors are proposed, which can be roughly classified into pattern matching based [5–8] and machine learning based [9–13]. On one hand, pattern matching works on a pre-defined pattern set, thus it is less efficient on detecting unseen hotspots. Although fuzzy-pattern matching can be applied to dynamically tune appropriate fuzzy regions around known hotspots, it is still case sensitive and lacks the generalization ability to various detection environment. On the other hand, with slightly longer training and prediction time, machine learning approaches have good generalization ability that may result in higher accuracy. Nevertheless, this method confronts high false alarm problem that many non-hotspots may be wrongly identified as hotspots. So far, neither conventional lithography simulation or recent machine learning and pattern matching is performing well enough.

One reason for the unsatisfactory performance is that circuit layout patterns are very complicated. Conventional machine learning methods are batch based, thus it is impossible to obtain all various circuit layout patterns to train a perfect learning model. Therefore, machine learning based methods are facing a big challenge: there are no enough representative instances for model building, which makes it difficult for the evaluation phase. As shown in Fig. 1(a), if training instances cover the feature space of testing instances, the machine learning model is very likely to detect all testing in-

*The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK14206015) and CUHK Direct Grant for Research.

ICCAD '16, November 07-10, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2966986.2967032

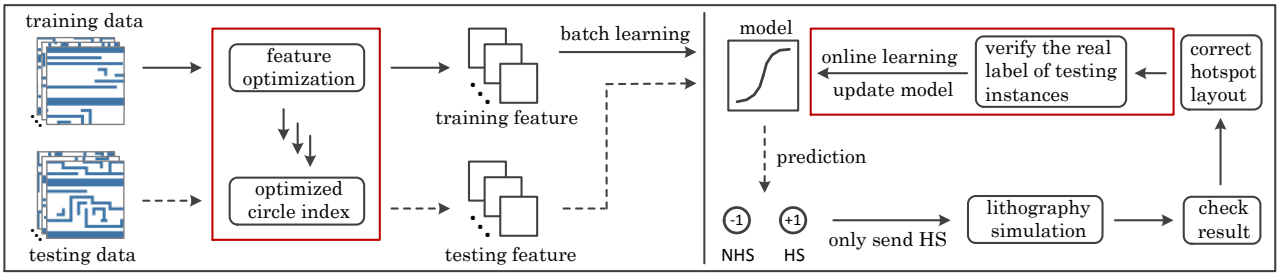

Figure 2: Overall online hotspot detection framework: Firstly, we apply feature optimization techniques proposed in Section 3 to get optimized circle index and extract feature for training data, followed by batch model learning. Then similarly, we extract feature for testing data and perform predictions. After training procedure in the left panel of the figure, we sequentially run lithography simulation for testing instances predicted as hotspots and online update verified instances into our model, as shown in the right panel of the figure. In this way, our model can be more accurate to later testing instances.

stances correctly. However, if it happens that some testing instances are in such feature space shown in Fig. 1(b), conventional batch based model can never predict those testing instances correctly even with some high dimensional kernels. Therefore, we propose an online hotspot detection framework to dynamically update our model with instances predicted as hotspots, which makes our model more capable of predicting new coming testing instances. For example in Fig. 1(b), those testing non-hotspots will be predicted as hotspots and their real label will be verified by lithography simulator. Given the labeled instances, online learning is performed to update our learning model and this can further enhance the prediction power of the model. Our online hotspot detector also shows its advantages of being compatible to our batch model (i.e. Our method can perform online learning without any modification of the batch model). In addition, our learning model will only be updated with testing instances predicted as hotspots whose label will be verified by lithography simulator in conventional methods. Therefore, there will be no additional lithography simulation time compared to conventional batch based methods. Equipped with online learning, our framework can further improve the runtime performance of the whole detection flow.

Besides the learning model, it is also imperative to capture key characteristics of hotspot and non-hotspot layout patterns effectively. Current feature extraction methods focus on density, shape, topology and other geometrical information of layout patterns (e.g. [8,14]). However, these methods have the following drawbacks: 1) It is hard for them to be adaptive to different layout designs; 2) Some complicated feature representation may cause severe over fitting problem. Hotspots are formed due to light passing through photomasks and causing interference with each other. Therefore, we should consider the impact of light propagation and interference [15] during feature extraction.

To overcome the drawbacks of conventional layout features, we propose a novel Maximal Circle Mutual Information (MCMI) scheme based on information theory. Under this scheme, the feature representation is optimized by a dynamic programming model. To break the limitation of conventional batch based methods, we develop an online learning method to integrate new instances to improve the learning model. To the best of our knowledge, this is the first time an online learning update is used to solve hotspot detection problem. Moreover, this is also the first time a novel information-theoretic scheme MCMI is proposed to perform feature optimization in hotspot detection problems. Our key contributions are as follows:

1. A novel MCMI scheme based on information theory is proposed to perform feature extraction and optimization for layout pattern representation, which can be adaptive to different layout designs;
2. An online boosting method is designed to bridge batch learning and online learning. A lossless online weak learning classifier matching with the layout feature property is constructed.
3. 3.47% of accuracy improvement, 58.23% and 58.88% reduction in the number of false alarm and *Overall Detection and Simulation Time (ODST)* can be achieved respectively comparing to state-of-the-art methods.

The rest of paper is organized as follows. In Section 2, we describe the online hotspot detection framework, evaluation metrics as well as problem formulations. In Section 3, our proposed layout feature is described. In Section 4, online learning model is derived. Section 5 presents the experimental results, followed by conclusion in Section 6.

2. PRELIMINARIES

In this section, we first discuss the online hotspot detection framework. Then we define several useful terms to describe the quality of our algorithm and give the problem formulation of batch hotspot detection as well as the online hotspot detection.

2.1 Online Hotspot Detection Framework

Conventional machine learning based hotspot detectors are batch based, where the learning model is trained on a fixed batch set of data. The conventional hotspot detection flow is shown in Fig. 2 without the two red boxes. When the model is finalized, testing instances are fed into the model and each gets a predicted value indicating the labels. Instances identified as hotspots are sent to lithography simulator and be further verified. Due to expensive lithography simulation, numerous false alarms would cause serious time overhead problem. Conventional batch learning based methods ignore the model updating step in the whole detection flow and shows its limitations in the following two aspects: 1) The learning model cannot be dynamically updated with new instances; 2) Numerous false alarms cause too much lithography simulation time.

Lithography simulation is an absolutely necessary step of the whole detection flow. In order to achieve better performance, it is natural to apply both lithography simulation and online model updating (as shown by the red box on the right

panel of Fig. 2) for sequentially coming testing instances . The superiority of our proposed framework is that it can not only increase the detection accuracy, reduce false alarms, but also significantly improve the runtime performance of the whole hotspot detection flow.

2.2 Evaluation Metrics

To quantify the performance of the proposed framework, we will define several terms. The most important issue of hotspot detection is to detect correctly as many hotspots as possible. Thus, to evaluate the accuracy of hotspot detection, we define the following term:

Definition 1 (Accuracy [16]). *The rate of correctly predicted hotspots among the set of actual hotspots.*

It may happen that a detector regards a testing instance as a hotspot whose real label is non-hotspot. We define the following term to evaluate the performance of this aspect:

Definition 2 (False Alarm [16]). *The number of incorrectly predicted non-hotspots.*

In the hotspot detection flow, we would like to achieve high accuracy and minimum false alarms, where high accuracy means hotspots are correctly detected while low false alarms lead to less lithography simulation time. Besides accuracy, no matter reducing the model testing time or lithography simulation time, our goal is to reduce the total runtime of the detection flow. Therefore, we define a new term, Overall Detection and Simulation Time (ODST), to measure the sum of model evaluation time and lithography simulation time.

Definition 3 (ODST). *The sum of the lithography simulation time for layout patterns detected as hotspots (including real hotspots and false alarms) and the learning model evaluation time.*

2.3 Problem Formulation

Here we give two problem formulations: conventional batch hotspot detection and proposed online hotspot detection.

Problem 1 (Batch Hotspot Detection). *The inputs are circuit layout data containing hotspot, non-hotspot layout patterns and their labels, the objective for batch hotspot detection is to train an efficient machine learning model that can maximize the accuracy and minimize the number of false alarms.*

As mentioned in previous sections, circuit layout patterns may be very complicated and we need to dynamically update testing instances to improve the existing model. The evaluation terms for conventional hotspot detection are accuracy and false alarm, however, it would be more reasonable to use accuracy and ODST to evaluate the performance of online hotspot detection flow. The problem formulation of *online hotspot detection* is as follows.

Problem 2 (Online Hotspot Detection). *The inputs are trained learning model, verified testing instances which are circuit layouts containing hotspot and non-hotspot layout patterns as well as their labels. The objective is to dynamically update the learning model to maximize accuracy and reduce ODST.*

3. LAYOUT FEATURE EXTRACTION

As one of the pattern recognition problems, hotspot detection is targeting at detecting potential problematic layout patterns. These patterns vary a lot among different designs and different circuit layers, thus extracting discriminative layout features plays an essential role in the detection flow. However, most of existing approaches rely on a pre-defined structure to extract layout features. Besides, these methods ignore the impact of light propagation during the extraction procedure. Although the feature extraction method in the work [17] intends to reflect light propagation by applying concentric circle sampling, it cannot be adaptive to different layout designs due to the manually set positions of those circles. Therefore, to better capture different layout pattern characteristics, we propose a novel feature selection method to extract layout features based on concentric circle sampling. We formulate it as a circle selection problem as described in Problem 3 and propose a Maximal Circle Mutual Information (MCMI) scheme, where the circle selection problem is formulated as an optimization problem and solved by a dynamic programming method.

Problem 3 (Circle Selection Problem). *Given a set of indices of potential circle position candidates* $\mathcal{I} = \{i|\ 1 \leq i \leq r_{max},\ i \in \mathbb{N}\}$ *(i denotes the circle index and also the radius of the circle in that position,* r_{max} *represents the largest circle radius that can be sampled in a layout clip and the distance between two adjacent circles is* $1nm$*) and the target layout pattern classification variable* y *(*$y \in \mathcal{Y}$ *and* $\mathcal{Y} = \{-1, 1\}$*), the objective is to find a subset of* n_c *circle position candidates indexed by* $\mathcal{I}_{n_c} \subseteq \mathcal{I}$ *(*n_c *is the number of circles we will select) that have the highest dependency with* y*.*

3.1 Circle Selection via Mutual Information

At first, we encode each circle to a more compact representation. We uniformly sample p points on each circle, every point is a binary number 0 or 1, where 1 means that the sampling point contains circuit layout and 0 means that the sampling point contains only blank area. We then concatenate all 0s and 1s on the i^{th} circle to form a bit sequence denoted as $\mathbf{z}_i$ (z_i^j stands for the j^{th} sampling point on the i^{th} circle and $\mathbf{z}_i$ is a vector variable with length p and $z_i^j \in \{0, 1\}$). We then convert them into decimal numbers. The computing method is as follows:

$$c_i = \sum_{j=0}^{p-1} z_i^j \cdot 2^j, \tag{1}$$

where c_i is the encoded decimal number and $c_i \in \mathcal{C}$, $\mathcal{C} = \{0, 1...2^p - 1\}$. For example, the circle in Fig. 3 should be encoded to 29.

For circle selection, it is desirable to maximize the dependency of selected circles on the classification variable y. *Mutual information* [18] is a fundamental notion in information theory that defines the amount of information held in a random variable. We use mutual information to compute the dependency between the i^{th} circle and the classification variable y. Let C_i be a random variable defined in $\mathcal{C}$ and Y be a random variable defined in the space $\mathcal{Y}$, then the mutual information is denoted as $I(C_i; Y)$:

$$I(C_i; Y) = \sum_{c_i \in C_i} \sum_{y \in Y} p(c_i, y) \log \frac{p(c_i, y)}{p(c_i)p(y)}, \tag{2}$$

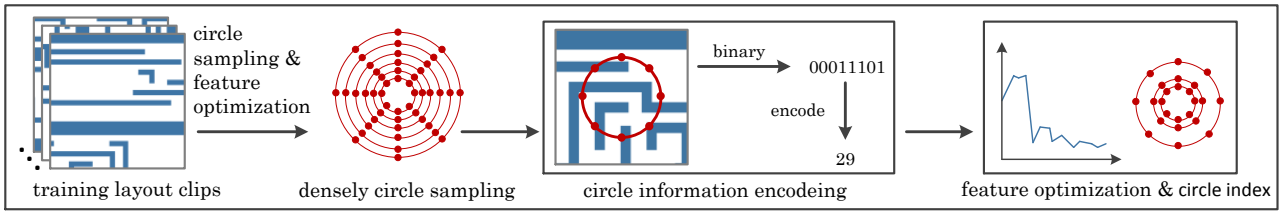

Figure 3: The overall feature optimization framework: Firstly, we densely sample a set of candidate circles from the training data and encode their binary bit sequences to decimal numbers; Then a dynamic programming model is applied to get an optimized circle index. At last, the features of both training and testing data are extracted by this optimized circle index.

where $p(c_i, y)$ is the joint probabilistic distribution function of random variable C_i and Y, and $p(c_i)$ and $p(y)$ are the marginal probability distribution function of C_i and Y respectively. $I(C_i; Y)$ in Eq. (2) can be easily computed using our training data.

The circle with a higher score of mutual information has a higher correlation with the label variable, and we would like to select a subset from all r_{max} circles that can help to predict the label variable better. We can see from Fig. 4 that circles closer to the clip center are more likely to get higher mutual information score than those far away from the clip center, thereby telling us that the hotspot is more sensitive to the nearer layout patterns. This observation is consistent with the impact of real light propagation and interference. Fig. 4 also shows that as the distance increases, the mutual information value has some slight phenomenon of periodic variation, which also corresponds to the property of light propagation. In addition, we can observe the possibility to select the circle subset adaptively from different layout design by the curve plots in Fig. 4.

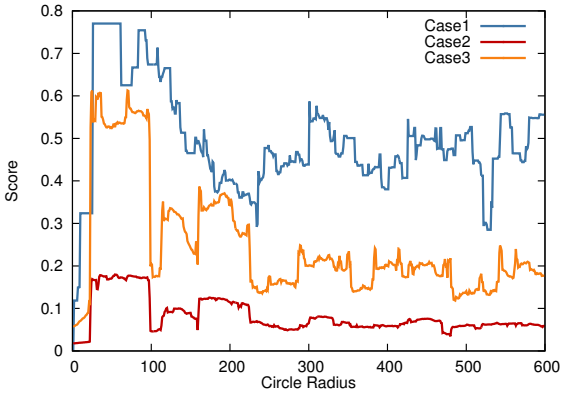

Figure 4: The relation between circle radius and the score of that circle.

3.2 MCMI and Circle Subset Selection

The target is to select a circle subset to represent our layout patterns. Fig. 4 shows that the circle nearer to the center is more likely to have higher mutual information score. However, if we only select the circles with highest scores, much redundant information will be included since adjacent circles contain similar information. Besides, useful information from circles with larger radius may be lost. Therefore, we propose a Maximal Circle Mutual Information (MCMI) scheme for our layout feature optimization.

$$\mathcal{I}^*_{n_c} = \arg\max_{\mathcal{I}_{n_c} \subseteq \mathcal{I}} \sum_{i \in \mathcal{I}_{n_c}} I(C_i; Y), \quad \text{s.t.} \quad |i-j| >= d,\ \forall\ i, j \in \mathcal{I}_{n_c}, \tag{3}$$

where $\mathcal{I}^*_{n_c}$ is selected circle index, $\mathcal{I}$ is defined in Problem 3 and $I(C_i; Y)$ is defined in Eq. (2). d is the minimum distance between two adjacent selected circles.

As we mentioned in the Problem 3, the distance between two adjacent circles is $1nm$, thus, for a 1200×1200 layout clip, we can densely sample 600 circles. However, only a few of these circles would contribute to the hotspot formation.

Therefore, we formulate an optimization problem to perform circle subset selection as follows:

$$\begin{aligned} \max \quad & \mathbf{v}^\top \mathbf{w}, \\ \text{s.t.} \quad & \sum_{i=1}^{r_{max}} w_i = n_c, \quad \forall\ i,\ w_i \in \{0, 1\}, \\ & |i-j| \geq d, \quad \forall\ i \neq j, w_i = 1, w_j = 1, \end{aligned} \tag{4}$$

where n_c is the desired number of circles. Variable $\mathbf{v}$ is a r_{max} dimensional vector, where v_i represents the mutual information score of each circle computed by $I(C_i; Y)$. Variable $\mathbf{w}$ is also a r_{max} dimensional vector, where w_i is used to indicate whether the i^{th} circle is selected. Since circles closer to each other are more likely to contain similar information, we use variable d to keep the distance between adjacent circles large enough, which may avoid over fitting and eliminate redundant information. Variable n_c is used to control the number of circles, as we only want to select the most informative circles.

In order to solve the optimization problem in Eq. (4), we apply the dynamic programming technique. Let $r_{max} = i, n_c = j$, then $D[i, j]$ is the optimal solution of Eqn. (4). Then we obtain the following recursion:

$$D[i, j] = \max\{v[i] + D[i-d, j-1], D[i-1, j]\}, \tag{5}$$

where $i \in \{1, ..., r_{max}\}, j \in \{1, ..., n_c\}$. After updating Eq. (5) for all i, j in D, we trace back from the $D[r_{max}, n_c]$ and get all the desired circles. After this feature optimization procedure, we can obtain the index of selected concentric circles $\mathcal{I}_{n_c} = \{i|\ w_i = 1\}$ and $\mathcal{I}_{n_c} \subseteq \mathcal{I}$. The final feature vector can be constructed from the index variable $\mathcal{I}_{n_c}$.

Performance comparison between selected circle subset and conventional circle subset [19] is shown in Fig. 5. We conduct our experiments in the same environment and under the same parameter setting except for the step of feature extraction. As we can see from Fig. 5, feature extraction using selected circle subset can increase accuracy and reduce false alarms over the one using conventional circle subset, which demonstrated

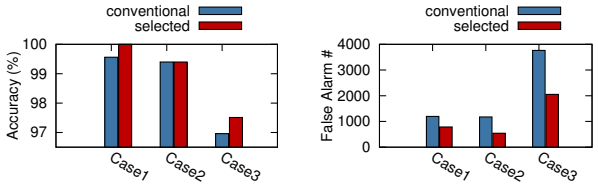

Figure 5: Performance comparison between optimized circle subset and regular circle subset. (a) The impact on accuracy; (b) The impact on false alarm number.

the superiority of our method. In particular, equipped with selected feature, false alarms are dramatically reduced and this saves a lot of lithography simulation time.

4. LEARNING MODEL

As mentioned in Section 1, conventional batch based machine learning model cannot capture all the complicated layout patterns due to the limited number of training instances. Thus, we apply online learning method to further improve existing model by querying the label of testing instances predicted as hotspots using lithography simulator. As mentioned in [13], Adaboost [20] with decision tree [21] have better performance than SVM [22] and ANN [23] in hotspot detection problems, simply because SVM is hard to select a suitable kernel for hotspot detection and ANN is difficult to define the right number of layers and hidden units. Inspired from that, we further improve the prediction power of machine learning by choosing the most suitable boosting classifier and corresponding weak classifier for our hotspot detection.

In the next two subsections, we will describe a boosting method with several good properties at first. Then we describe the intuition of hotspot formation and its relations with our layout feature. At last we design a lossless [25] (The classifier stays the same no matter batch learning or online learning) online Naive Bayes classifier modified from [24] to enhance the performance of our layout feature.

4.1 Online Ensemble Method

In practice, there exists malicious noises when generating the label of a training layout pattern in our hotspot detection problem and the noise may also happen during the layout feature extraction. Therefore, we would like to eliminate these noises by applying a more robust classifier with the ability to be online updated by new instances. However, Adaboost cannot handle the problem of malicious noise occurred in training data and is hard to be extended to an online version. Nevertheless, Smooth Boosting [25] is a good candidate with several good properties: 1) It can deal with noise and outliers that often occur in hotspot detection problems; 2) It can efficiently avoid over fitting problem; 3) It can be used to boost weak classifiers with real value output; 4) It can be easily extended to the online scenario and bridge batch learning and online learning.

Let $\mathcal{X} \subseteq \mathbb{Z}^n$ (our features are represented by integral values) be the input feature space. The training data for our boosting method is $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$. Variable γ, θ, T are parameters defined in Smooth Boosting [25], The formulation of batch based Smooth Boosting is shown in Algorithm 1.

The difficulty of adjusting Adaboost to an online version is

Algorithm 1 Smooth Boosting

Input: $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$, γ, $\theta = \frac{\gamma}{2+\gamma}$, T.

```
for i ← 1 to n do
    M_1(i) ← 1;
    N_0(i) ← 0;
end for
for t ← 1 to T do
    Run weak classifier to get h_t such that
    (1/2) Σ_{j=1}^{n} M_t(j)|h_t(x_j) − y_j| ≤ 1/2 − γ;
    for j ← 1 to n do
        N_t(j) ← N_{t−1}(j) + y_j h_t(x_j) − θ;
    end for
    for j ← 1 to n do
        M_{t+1}(j) ← min{1.0, (1 − γ)^{N_t(j)/2}};
    end for
end for
return f ←sign((1/T) Σ_{t=1}^{T} h_t);
```

the procedure of updating instance weights, where Adaboost requires to see all examples to determine the instance weights. Although online Adaboost [26] uses poisson sampling process to approximate the weighting scheme of Adaboost, it can only ensure a good hypothesis asymptotically. Luckily, we can adopt weight updating scheme in Smooth Boosting [25] for our online version by updating the weight term $M_{t+1}(j)$ as follows:

$$M_{t+1}(j) \leftarrow \min\{1.0, (1-\gamma)^{\frac{N_t(j)}{2}}\}, \tag{6}$$

where $N_t(j)$ can be viewed as the cumulative error on the j^{th} instance over the previous t rounds of weak classifier update, γ is the guaranteed advantage of the hypotheses returned by the weak learner [25].

The instance weight updating scheme of Smooth Boosting guarantees that the weight update is independent for each instance (only depends on $N(j)$, the cumulative error of the j^{th} instance). Therefore, Smooth Boosting provides a convenient way to update streaming instance without seeing all instances. In addition, Smooth Boosting can bridge batch learning and online learning by trivial modifications of the original algorithm, while online Adaboost [26] cannot. Algorithm 2 shows how online Smooth Boosting incorporates streaming instances.

Algorithm 2 Online Boosting

Input: Streaming instance $(\mathbf{x}, y)$, batch Smboost classifier.

```
M_1 ← 1, N_0 ← 0;
for t ← 1 to T do
    online update h_t(x, y);
    N_t ← N_{t−1} + y h_t(x) − θ;
    M_{t+1} ← min{1.0, (1 − γ)^{N_t/2}};
end for
return f ←sign((1/T) Σ_{t=1}^{T} h_t);
```

4.2 Online Weak Classifier

Smooth Boosting methods can efficiently boost the performance of weak classifiers and can be online updated with instances predicted as hotspots. However, Smooth Boosting requires efficient online weak classifier to guarantee the overall performance, and moreover, this weak classifier should also

be related to our layout feature. In addition, it is also important to derive a lossless online weak classifier, that is, as data streams in, its performance will stay the same compared with that of batch learning. In this work, we apply a modified Naive Bayes model as our weak classifier, because it not only is a lossless online classifier but also can work well with our proposed layout feature.

In our proposed layout feature, we sample points from several concentric circles and each sampled point tells whether the location contains layout pattern or not. Thus, it is natural to consider the probability of one sampling point contributing to the hotspot formation. Let $(\mathbf{x}_i, y_i)$ be the data of i^{th} training instance, where $\mathbf{x}_i$ is its feature vector and y_i is the label. The probability of the j^{th} sampling point in the i^{th} instance is as follows:

$$P(x_i^j|y), \tag{7}$$

where x_i^j is the value in the j^{th} dimension of the i^{th} instance's feature vector.

As hotspot is caused by all the layout patterns in the neighborhood, we will measure all the influence that nearby points can contribute to the hotspot by:

$$P(\mathbf{x}_i|y) = P(x_i^1, x_i^2, ..., x_i^n|y), \tag{8}$$

where n is the feature dimension. However, it is very hard to measure this complicated joint probability distribution function due to its high dimension and insufficient training data. Therefore, we propose two assumptions for this classifier related to our layout feature. One assumption is that sampling points within a circle are dependent to the hotspot formation (It corresponds to interference from light source at equal distance from the hotspot). Another assumption is that different sampling circles are independent to each other (It corresponds to our circle selection procedure). In practice, circles may not be independent to each other and may also cause light interference. However, we adopt this assumption to reduce the model complexity and avoid over fitting. Thus, we modify conventional Naive Bayes classifier and adjust it to our assumptions as follows:

$$p(x_i^1, x_i^2, ..., x_i^n|y) = \prod_{j=0}^{\lfloor \frac{n}{p} \rfloor - 1} p(x_i^{j \cdot p}, ..., x_i^{(j+1)\cdot p-1}|y), \tag{9}$$

where p is the number of sampling points in one circle as we mentioned in Section 3. Therefore, the classification rules for our classifier are as follows:

$$y_i^* = \arg\max_y \ p(y) \prod_{j=0}^{\lfloor \frac{n}{p} \rfloor - 1} p(x_i^{j \cdot p}, ..., x_i^{(j+1)\cdot p-1}|y), \tag{10}$$

where y_i^* is the predicted result of the i^{th} testing instance. The prior probability parameters can be easily computed by optimizing maximum likelyhood function. The online version of Naive Bayes classifier can be obtained easily. When new instance comes, we will update the corresponding prior probabilistic terms, $p(y)$ and $p(x_i^{j \cdot p}, ..., x_i^{(j+1)\cdot p-1}|y)$.

5. EXPERIMENTAL RESULTS

The proposed framework for both batch learning and online learning are implemented in `Python` programming languages, and accelerated by `Cython` on a machine with four-core 3.7GHz CPUs and 16GB memory. The performance of the proposed framework are evaluated on ICCAD-2012 CAD contest benchmark suite [16], which consists of one $32nm$ circuit layout and four $28nm$ circuit layouts (detailed in Table 1). For each test case, a set of layout clips are used as training data to generate the learning model, while another set of layout clips are used as testing data to evaluate the quality of the learning model. For training data, rows "**Training HS**" and "**Training NHS**" give the hotspot number and non-hotspot number, respectively. For testing data, row "**Testing HS**" lists the hotspot number for each case. Recall in Section 2, overall detection and simulation time (ODST) consists of two parts: lithography simulation time and learning model evaluation time. In this paper, we use an industry lithography simulator [27] to carry out simulation on hotspots reported by each learning model. Observe that through utilizing multi-core parallelism, the simulation time for each layout core is around 10 seconds, thus in ODST calculation, we set the simulation runtime penalty of each hotspot reported by learning model to 10 seconds.

Table 1: ICCAD-2012 benchmark statistics [16]

	Case1	Case2	Case3	Case4	Case5
Technology	$32nm$	$28nm$	$28nm$	$28nm$	$28nm$
Training HS	99	174	909	95	26
Training NHS	340	5285	4643	4452	2716
Testing HS	226	498	1808	177	41
Testing area (mm^2)	12516	106954	122565	82010	49583

Table 2: Comparison with hotspot detector [13]

	SPIE'15 [13]			batch		
	FA#	CPU(s)	Accuracy	FA#	CPU(s)	Accuracy
Case1	**0**	**7**	**100.00%**	**0**	**7**	**100.00%**
Case2	**0**	351	98.60%	**0**	**51**	**99.40%**
Case3	**0**	297	97.20%	3	**66**	**97.51%**
Case4	1	170	87.01%	**0**	**35**	**97.74%**
Case5	**0**	69	92.86%	**0**	**24**	**95.12%**
average	**0.2**	178.8	95.13%	0.6	**36.8**	**97.95%**
ratio	-	4.86	0.97	-	**1.0**	**1.0%**

5.1 Effectiveness of Batch Learning

In the first experiment, we compare our batch learning based hotspot detector, denoted as "**batch**", with a recent hotspot detector [13] on the ICCAD-2012 benchmark, and the detailed comparisons are shown in Table 2. For each detector, columns "**FA#**", "**CPU(s)**" and "**Accuracy**" list the corresponding false alarm number, runtime of model evaluation in seconds, and the accuracy of our detector, respectively. Note that the detector of [13] utilizes pre-filtered layout clips that all hotspots are known to be the center of the clips. In other words, during testing layout scanning, only the core area of each clip will be verified, which by nature may report less false alarm number. To have a fair comparison, in this experiment, our detector scans the core area of each clip as well. We can see from Table 2 that comparing with [13] our framework can achieve around 5× speed-up and increase detection accuracy from 95.13% to 97.95%. Meanwhile, for all the five cases, only 3 false alarms in total are reported. The superiority of our method compared to [13] may be related to the MCMI scheme, where layout feature can be adaptive to different layout designs.

In the second experiment, we further compare our batch learning based method with two state-of-the-art hotspot detectors [8, 14], and the comparison is shown in Table 3. Dif-

Table 3: Comparisons with two state-of-the-art hotspot detectors [8, 14]

	TCAD'14 [8]				TCAD'15 [14]				batch			
	FA#	CPU(s)	ODST(s)	Accuracy	FA#	CPU(s)	ODST(s)	Accuracy	FA#	CPU(s)	ODST(s)	Accuracy
Case1	1714	11	17151	100.00%	1493	38	14968	94.69%	**788**	**10**	**7890**	**100.00%**
Case2	4058	287	40867	**99.80%**	11834	234	118574	98.20%	**544**	**103**	**5543**	99.40%
Case3	9486	417	95277	93.80%	13850	778	139278	91.88%	**2052**	**110**	**20630**	**97.51%**
Case4	**1120**	102	11302	91.00%	3664	356	36996	85.94%	3341	**69**	33478	**97.74%**
Case5	199	49	2039	87.80%	1205	**20**	12070	92.86%	**94**	41	**980**	**95.12%**
avg.	3315.4	173.2	33327.2	94.48%	6409.2	285.2	64377.2	92.71%	**1363.8**	**67.5**	**13705.5**	**97.95%**
ratio	-	-	2.43	0.96	-	-	4.70	0.95	-	-	**1.0**	**1.0**

Table 4: Batch learning v.s. online learning

	batch				online						
	FA#	CPU(s)	ODST(s)	Acccuracy	HS#	NHS#	FA#	FA# Redu.	CPU(s)	ODST(s)	Accuracy
Case1	788	10	7890	**100.00%**	226	704	**704**	84	12	**7048**	**100.00%**
Case2	544	103	5543	**99.40%**	495	308	**308**	236	113	**3183**	**99.40%**
Case3	2052	110	20630	97.51%	1764	1819	**1819**	233	133	**18299**	**97.57%**
Case4	3341	69	33478	**97.74%**	173	2096	**2096**	1245	81	**21019**	**97.74%**
Case5	94	41	980	95.12%	40	82	**82**	12	43	**847**	**97.56%**
avg.	1363.8	67.5	13705.5	97.95%	539.6	1008.8	**1008.8**	362	76.6	**10079.2**	**98.45%**
ratio	-	-	1.36	0.99	-	-	-	-	-	**1.0**	**1.0**

ferent from Table 2, for each detector, column "**ODST(s)**" lists the overall detection and simulation time in seconds. In this experiment, the layout is decomposed into a set of continuous but independent grids, which is with the same size of the core area in training layout. Then all grids are scanned in ordered and are verified by our batch based hotspot detector. Only the grids covering actual hotspot are labelled as actual hotspot grids. We can see from Table 3 that our method achieves the highest average accuracy, lowest average number of false alarms, lowest average run time and lowest average ODST. Our method improve the accuracy by 5.24% and 3.47% on average (and a maximum of 6.74%) compared with [14] and [8] respectively. We also reduce the number of false alarms by 78.39% and 58.23% on average compared with [14] and [8] respectively. As described in Section 2, ODST can better represent the runtime performance in hotspot detection, as those instances predicted as hotspots will be verified by lithography simulator. Table 3 shows that our methods improve the ODST by 78.71% and 58.88% on average compared with [14] and [8] respectively. Besides the proposed MCMI scheme, Smooth Boosting can efficient eliminate malicious noise and modified Naive Bayes allows us to measure correctly the dependency of different sampling points on a same or different circle, thus our framework significantly outperforms other methods.

5.2 Effectiveness of Online Learning

Although we have demonstrated the superiority of our batch learning method in Table 2 and Table 3, some layout patterns still cannot be predicted correctly, because there is no enough training data to capture all the layout patterns needed, as shown in Fig. 1. In the third experiment, we extend our conventional batch learning to online learning scenario. Here we enable dynamically learning model update at testing stage and hopefully we can predict the coming instances more accurately. As we know, no matter for real hotspots or false alarms, once a layout pattern is predicted as hotspot by the learning model, lithography simulation will be performed to further verify its real label. Therefore, it is necessary to perform model updating after lithography simulation, which makes our model more accurate for the coming testing instances.

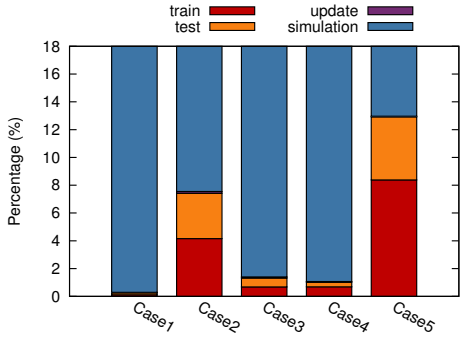

Figure 6: Runtime breakdown for ICCAD benchmark.

We compare our online learning model with our batch learning model to verify the robustness of our online learning framework. Detailed comparisons of our batch learning model and online learning model are shown in Table 4. Our online learning model is denoted as "**online**" in Table 4 and columns "**HS#**", "**NHS#**", "**FA# Redu.**" list the corresponding number of updated testing hotspots, number of updated non-hotspots and the number of reduced non-hotspots compared to batch learning model, respectively ("**NHS#**" equals "**FA#**", because the model will update all instances predicted as hotspots). We also evaluate our online learning framework in terms of accuracy, false alarms and ODST. ODST includes both model evaluation time and lithography simulation time, which is the real runtime for the practical hotspot detection flow. We can see from Table 4, our online learning framework not only increases the detection accuracy from 97.95% to 98.45%, but also further reduces the ODST by 26.5%. It should be noted that through utilizing online learning, we can significantly reduce the false alarm number, as shown in column "**FA# Redu.**".

Fig. 6 illustrates the runtime breakdown of our online learning model on the ICCAD-2012 benchmark suite, where total runtime of online hotspot detection consists of four parts: training, testing, online update, and lithography simulation. We can see from Fig. 6 that in our framework, through online scheme, updating time is only a small portion of the whole online detection flow, which means that we have saved a lot

of runtime on lithography simulation by applying our online hotspot detection framework. Moreover, we can see from the figure clearly that the number of false alarms in Case 2 and Case 5 are dramatically reduced, since lithography simulation time occupies less proportion. Compared to batch based methods, there is no extra lithography simulation time in our online learning framework. In contrast, our framework can further improve the runtime performance.

6. CONCLUSION

In this paper, we have proposed a novel MCMI scheme based on information theory to perform feature extraction and optimization. Guiding by the MCMI scheme, our proposed layout feature can be adaptive to different layout designs. Besides, we constructed an ensemble classifier using Smooth Boosting and modified Naive Bayes, which outperforms state-of-the-art methods in terms of accuracy, false alarms and ODST in hotspot detection. More importantly, we extend our framework to the online learning scenario, which can further reduce the time needed for lithography simulation. With increasing design complexity and prohibitive runtime overhead of full-chip simulation, we anticipate the proposed online learning methodology will become more and more relevant and expect to see a lot of researches.

Acknowledgment

The authors would like to thank Yi Zou from ASML Brion for helpful discussions.

7. REFERENCES

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

[2] Hirokazu Nosato, Hidenori Sakanashi, Eiichi Takahashi, Masahiro Murakawa, Tetsuaki Matsunawa, Shimon Maeda, Satoshi Tanaka, and Shoji Mimotogi. Hotspot prevention and detection method using an image-recognition technique based on higher-order local autocorrelation. *Journal of Micro/Nanolithography, MEMS, and MOEMS (JM3)*, 13(1):011007, 2014.

[3] Juhwan Kim and Minghui Fan. Hotspot detection on Post-OPC layout using full chip simulation based verification tool: A case study with aerial image simulation. In *Proceedings of SPIE*, volume 5256, 2003.

[4] Ed Roseboom, Mark Rossman, Fang-Cheng Chang, and Philippe Hurat. Automated full-chip hotspot detection and removal flow for interconnect layers of cell-based designs. In *Proceedings of SPIE*, volume 6521, 2007.

[5] Jingyu Xu, Subarna Sinha, and Charles C. Chiang. Accurate detection for process-hotspots with vias and incomplete specification. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 839–846, 2007.

[6] Yen-Ting Yu, Ya-Chung Chan, Subarna Sinha, Iris Hui-Ru Jiang, and Charles Chiang. Accurate process-hotspot detection using critical design rule extraction. In *ACM/IEEE Design Automation Conference (DAC)*, pages 1167–1172, 2012.

[7] Sheng-Yuan Lin, Jing-Yi Chen, Jin-Cheng Li, Wan-Yu Wen, and Shih-Chieh Chang. A novel fuzzy matching model for lithography hotspot detection. In *ACM/IEEE Design Automation Conference (DAC)*, pages 68:1–68:6, 2013.

[8] Wan-Yu Wen, Jin-Cheng Li, Sheng-Yuan Lin, Jing-Yi Chen, and Shih-Chieh Chang. A fuzzy-matching model with grid reduction for lithography hotspot detection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 33(11):1671–1680, 2014.

[9] Dragoljub G. Drmanac, Frank Liu, and Li-C. Wang. Predicting variability in nanoscale lithography processes. In *ACM/IEEE Design Automation Conference (DAC)*, pages 545–550, 2009.

[10] Duo Ding, J. Andres Torres, and David Z. Pan. High performance lithography hotspot detection with successively refined pattern identifications and machine learning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 30(11):1621–1634, 2011.

[11] Duo Ding, Bei Yu, Joydeep Ghosh, and David Z. Pan. EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation. In *IEEE/ACM Asia and South Pacific Design Automation Conference (ASPDAC)*, pages 263–270, 2012.

[12] Yen-Ting Yu, Geng-He Lin, Iris Hui-Ru Jiang, and Charles Chiang. Machine-learning-based hotspot detection using topological classification and critical feature extraction. In *ACM/IEEE Design Automation Conference (DAC)*, pages 671–676, 2013.

[13] Tetsuaki Matsunawa, Jhih-Rong Gao, Bei Yu, and David Z. Pan. A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction. In *Proceedings of SPIE*, volume 9427, 2015.

[14] Yen-Ting Yu, Geng-He Lin, Iris Hui-Ru Jiang, and Charles Chiang. Machine-learning-based hotspot detection using topological classification and critical feature extraction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 34(3):460–470, 2015.

[15] Chris Mack. *Fundamental Principles of Optical Lithography: The Science of Microfabrication*. John Wiley & Sons, 2008.

[16] Andres J. Torres. ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 349–350, 2012.

[17] Tetsuaki Matsunawa, Bei Yu, and David Z. Pan. Optical proximity correction with hierarchical bayes model. In *Proceedings of SPIE*, volume 9426, 2015.

[18] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[19] Allan Gu and Avideh Zakhor. Optical proximity correction with linear regression. *IEEE Transactions on Semiconductor Manufacturing (TSM)*, 21(2):263–271, 2008.

[20] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *Annals of Statistics*, 28(2):337–407, 2000.

[21] Leo Breiman, Jerome Friedman, Charles J. Stone, and Richard A. Olshen. *Classification and Regression Trees*. CRC press, 1984.

[22] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, 2001.

[23] M.Y. Rafiq, G. Bugmann, and D.J. Easterbrook. Neural network design for engineering applications. *Computers & Structures*, 79(17):1541–1552, 2001.

[24] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

[25] Rocco A. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4:633–648, 2003.

[26] Nikunj C. Oza. Online bagging and boosting. In *IEEE International Conference on Systems, Man, and Cybernetics (ICSMC)*, volume 3, pages 2340–2345, 2005.

[27] Shayak Banerjee, Zhuo Li, and Sani R. Nassif. ICCAD-2013 CAD contest in mask optimization and benchmark suite. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 271–274, 2013.