

A Fuzzy-Matching Model With Grid Reduction for Lithography Hotspot Detection

Wan-Yu Wen, Jin-Cheng Li, Sheng-Yuan Lin, Jing-Yi Chen, and Shih-Chieh Chang, *Senior Member, IEEE*

Abstract—In advanced IC manufacturing, as the gap increases between lithography optical wavelength and feature size, it becomes challenging to detect problematic layout patterns called lithography hotspot. In this paper, we propose a novel fuzzy matching model which extracts appropriate feature vectors of hotspot and nonhotspot patterns. Our model can dynamically tune appropriate fuzzy regions around known hotspots. Based on this paper, we develop a fast algorithm for lithography hotspot detection with high accuracy of detection and low probability of false-alarm counts. In addition, since higher dimensional size of feature vectors can produce better accuracy but requires longer run time, this paper proposes a grid reduction technique to significantly reduce the CPU run time with very minor impact on the advantages of higher dimensional space. Our results are very encouraging, with average 94.5% accuracy and low false-alarm counts on a set of test benchmarks.

Index Terms—Design for manufacturability, dimensionality reduction, fuzzy matching, hotspot detection, lithography hotspot, machine learning.

I. INTRODUCTION

AS MODERN IC feature size shrinks, chip manufacturing is limited to printability of optical lithography [8]. The limitation is caused by manufacturing variability of the layout with sub-wavelength feature size. Fig. 1 shows the limitation of lithography manufacturing. The symbol λ represents the wave-length of lithography technique, and d represents the feature size of chips. During the lithography process, as in Fig. 1(b), when λ is larger than d , a diffraction phenomenon will occur and cause layout problems. To detect and avoid problematic layout patterns (called lithography hotspots), many manufacturability-aware methods such as design rule check, optical proximity correction, and other resolution enhancement techniques have been developed.

Despite those efforts, there are still several challenges to detect lithography hotspots. As feature size becomes much smaller than the optical wavelength adopted in the lithography technique, more layout objects are influenced in the optical radius. Consequently, a detection method should consider using a larger area to check whether a design layout is problematic. For example, although the central pattern of the

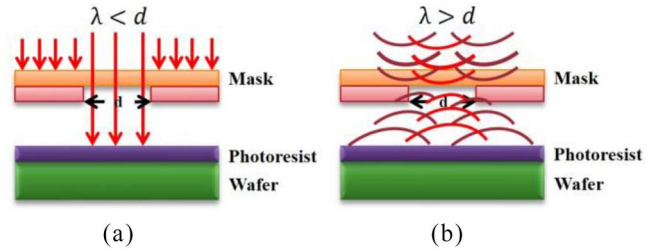


Fig. 1. Limitation of lithography manufacturing. (a) Lithography process. (b) Diffraction phenomenon of lithography.

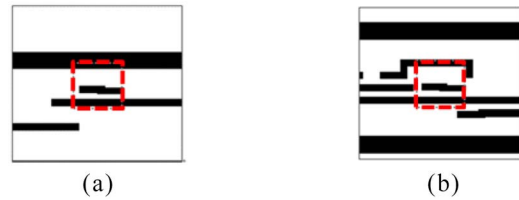


Fig. 2. Layout pattern. (a) Nonhotspot. (b) Hotspot.

layout of Fig. 2(a) is the same as that of Fig. 2(b), only the layout of Fig. 2(b) is identified as a hotspot.

The previous works to detect a “large-area” lithography hotspot can be categorized into two types. The first type uses full-layout lithography simulation [17], [19] which is accurate but suffers from high computation complexity. For this reason, the second type uses geometric methods for better efficiency. Most state-of-the-art techniques adopt the geometric methods.

Fig. 3 shows the flow of a geometric method. This method first collects a set of known hotspot or nonhotspot patterns and then builds a classifying model. After that, given an unknown layout pattern, the classifying model can recognize whether an unknown pattern is a hotspot or nonhotspot.

The previous works using the geometric method can be classified into two categories.

- 1) Pattern matching approaches [1], [7], [16], [18], [23] have full detection capability on previously identified hotspots but are less efficient in classifying unseen patterns.
- 2) Machine learning techniques such as artificial neural network (ANN) [4], [6], [20] and support vector machine (SVM) [3], [4], [12]–[14] may be better in prediction accuracy, (that is, an unseen hotspot can be recognized as a hotspot). Nevertheless, this type of method normally suffers from the problem of false alarm, (that is, many nonhotspots are identified as hotspots).

Manuscript received January 24, 2014; revised April 14, 2014; accepted June 29, 2014. Date of current version October 16, 2014. This paper was recommended by Associate Editor P. Gupta.

The authors are with the Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan (e-mail: sapphirefreshman@gmail.com; lgc80040@gmail.com; hyskyzhe@gmail.com; jy.chen1219@gmail.com; scchang@cs.nthu.edu.tw).

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.2014.2351273

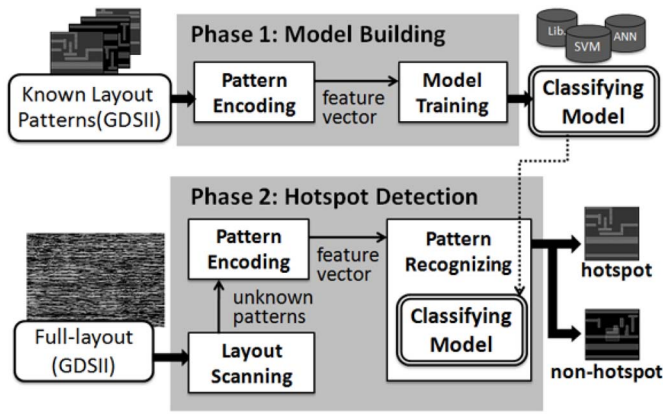


Fig. 3. Geometry hotspot detection flow.

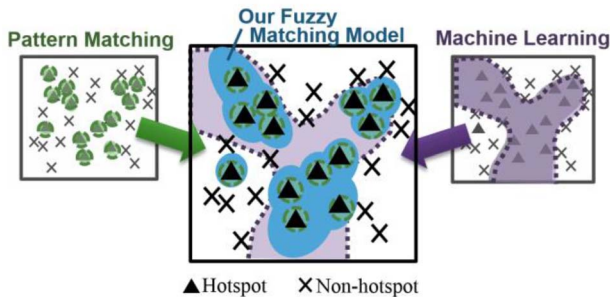


Fig. 4. 2-D-space example of hotspot region decision.

Recently, to combine the advantages of the above two categories, hybrid hotspot detection methods have been proposed. Ding *et al.* [5] presented a meta-classifier which combines multiple machine learning classifiers and pattern matchers using a weighted function. Wu *et al.* [15] and Mostafa *et al.* [21] proposed a detection flow that applies pattern matching first and then employs machine learning as the secondary checker. Although, these methods obtain better accuracy than previous works have done, they still suffer from a high false-alarm rate. Furthermore, to achieve such accuracy, these hybrid models use multiple classifiers to check numerous potential problematic locations in the full layout. As mentioned in [9], such hybrid models may perform as much as 10–100 times slower than pattern-matching approaches.

In this paper, we propose a novel fuzzy-matching model for fast lithography hotspot detection. Our model naturally integrates both pattern-matching and machine-learning techniques. Fig. 4 demonstrates our basic idea using an example in 2-D space where a triangle represents a hotspot while a cross represents a nonhotspot. The right diagram of Fig. 4 demonstrates the basic concept of a machine-learning technique. A machine-learning technique divides the space into two region. The hotspot region (the highlighted region) and the nonhotspot region. The left diagram in Fig. 4 demonstrates the basic concept of a pattern-matching approach which records all known-hotspot samples. The results of our fuzzy-matching model are shown in the central diagram in Fig. 4. Our model dynamically tunes the fuzzy region around a known hotspot.

Because the proposed fuzzy matching is based on pattern matching, our method not only can safely detect previously identified hotspots but also can run much faster than hybrid models. To the best knowledge of the authors, this is the first time a fuzzy algorithm has been applied in the pattern-recognizing stage of this problem. In addition, we show that a higher dimensional size of feature vectors can produce higher accuracy, but it takes longer run-time. In this paper, we also apply the principal component analysis (PCA) to significantly reduce the CPU run time with very minor impact on the advantages of higher dimensional space. Experimental results show that our fuzzy-matching model has 94.5% accuracy of hotspot detection with low false-alarm counts.

The remainder of the paper is organized as follows. In Section II, we review the hotspot detection flow and a pattern encoding method for the following article. In Section III, we discuss our fuzzy-matching model in detail. In Section IV, our fuzzy-matching detection flow and a grid-reduction strategy are presented. Section V shows the experimental results and analysis. Section VI concludes this paper.

II. PRELIMINARIES

This section discusses the background information for hotspot detection. Section II-A introduces the general geometry-based detection flow composed of two phases, and Section II-B reviews a pattern encoding method applied in this paper.

A. Hotspot Detection Flow

Fig. 3 shows a conventional flow for the geometric hotspot detection. The flow consists of two phases. The model-building phase and the hotspot-detection phase. The objective of the model-building phase is to build a classifying model. First, a set of known hotspot or nonhotspot patterns is collected. Then, a pattern encoding method is applied. This method extracts and encodes essential features which characterize the geometric properties of layout patterns of hotspots and nonhotspots. After the pattern-encoding stage, feature vectors are obtained and used as data to train various models such as pattern matcher, ANN, or SVM models.

The objective of the second phase is to detect hotspots in a full layout. This phase scans the entire layout and tags potential problematic patterns as unknown patterns to be carefully analyzed later. Then, the same encoding method is applied to extract essential features from these unknown patterns. After that, these vectors are fed into the classifying model trained in the first phase to determine whether an unknown pattern is a hotspot or nonhotspot.

B. Pattern Encoding

In this section, we review the layout pattern method called density-based encoding in [13] and [14]. Fig. 5 illustrates the basic concept. Given a layout pattern of a predefined grid, the method calculates the layout-covering density of each grid. After that, the covering densities of grids are encoded as an ordered feature vector. As in Fig. 5, we divide the original

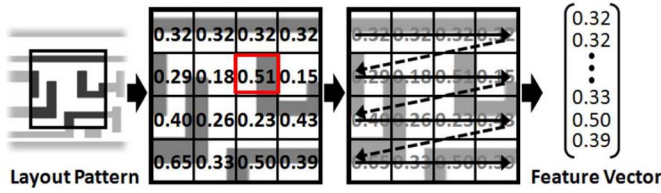


Fig. 5. Density-based layout pattern encoding.

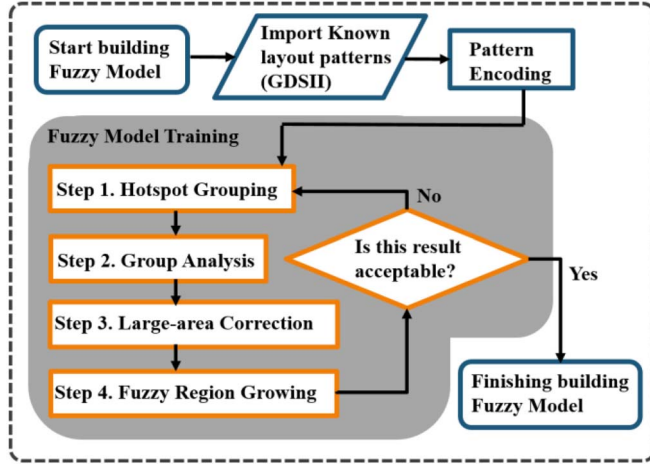


Fig. 6. Building flow of our fuzzy-matching model.

space into 4×4 sub-partitions. We encode the each sub-pattern within its sub-partition by density-based encoding. Let us assume the ordered feature vector to be $\{0.32, 0.32, 0.32, 0.32, 0.29, 0.18, 0.51, 0.15, 0.4, 0.26, 0.23, 0.43, 0.65, 0.33, 0.5, 0.39\}$ in the 16-dimensional space.

For clarity, in this paper, a layout pattern is encoded into a node in the multidimensional space through the following three steps.

- 1) Define grid on the pattern and calculate the covering densities of each grid.
- 2) Encode the covering densities as an ordered feature vector.
- 3) Map the feature vector to a node in the multidimensional space.

Note that during the density encoding, many “similar” layout patterns have a similar covering density. For example, there are many different patterns whose covering density is 0.5. As a result, the density encoding is regarded as a “fuzzy” encoding.

III. FUZZY-MATCHING MODEL

In this section, we present the flow of our fuzzy-matching model as shown in Fig. 6. The training of our fuzzy-matching model iterates the following four steps. In Step 1, we partition hotspot patterns into a set of groups in Section III-A. Step 2 extracts characteristics of a hotspot group in Section III-B. In Step 3, we design a correction method to lower the false-alarm rate in Section III-C. Step 4 presents the fuzzy-region growing algorithm to decide the fuzzy region around known hotspots in Sections III-D and III-E. Finally, Section III-F describes the condition needed to terminate the iteration. Once the model

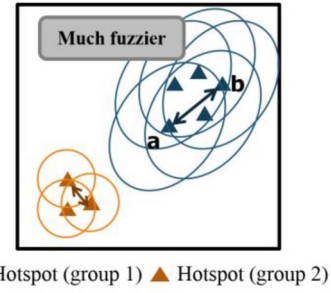


Fig. 7. Example of fuzzy-level grouping.

is built, Section III-G explains the procedure to recognize whether an unknown pattern is a hotspot or not.

A. Hotspot Grouping

The objective of hotspot grouping is to partition hotspots into groups. Hotspots with certain similarities are assigned to the same group. The similarity is measured by Cityblock distance as defined below.

Definition 1: We define the Cityblock distance, $D_c(a, b)$ to be the distance between two nodes, (a, b) in

$$D_c(a, b) = \sum_{i=1}^{\text{dimension}} |a_i - b_i|, \text{ for } a, b \in \text{nodes} \quad (1)$$

where a_i and b_i are defined to be the value of i th dimension of node a and b , respectively.

If D_c between two hotspots is smaller than an important variable called group distance, these two hotspots are assigned to the same group. The group distance is iteratively updated and will be discussed in Section III-F. Using the Cityblock distance, two hotspots which are conceptually close in the multidimensional space are assigned to the same group.

Finally, we observe that group 1 is “fuzzier” than group 2 if the maximum distance of any two hotspots in group 1 is larger than those in group 2. Let us consider the example in Fig. 7 which has two groups. It can be seen that group 1 is fuzzier than group 2.

B. Group Analysis

The previous section has shown how to partition hotspots into a set of groups. In this section, for a group of hotspots, we extract a special characteristic for the group. Before describing how the characteristic is mathematically computed, we present our basic ideas. First, if all hotspots in the same group are closer in certain dimensions, we consider these dimensions to be conceptually more representative for the group. For example, in Fig. 8(a), ten hotspots marked as triangles are in the same group.

Since the ten hotspots are closer in i -dimension than in j -dimension, we consider i -dimension to be more representative than j -dimension for that group. The less representative dimension is considered fuzzier. The hotspot region will grow more in the fuzzier dimension. Let us consider the example, in Fig. 8(a) where j -dimension is fuzzier than i -dimension. As a result, in Fig. 8(b), the hotspot region for this group grows more in the j -dimension than in the i -dimension.

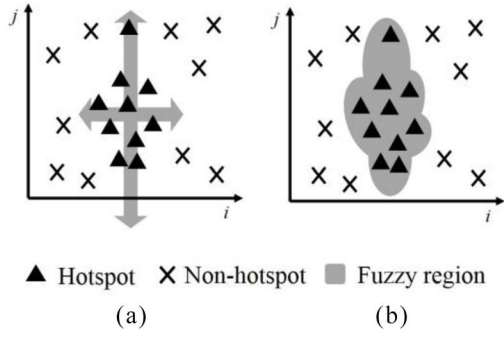


Fig. 8. Example of representative dimension. (a) Hotspot fuzzy level. (b) Hotspot representative region.

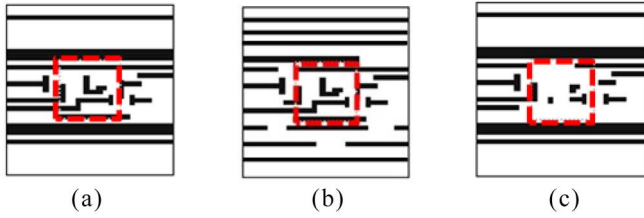


Fig. 9. Example of large-area effect. (a) and (b) Hotspot. (c) Nonhotspot.

To describe the characteristic of a group, we use the dimensional weight defined as follows. A higher weight implies the corresponding dimension is more representative. Let us consider one hotspot group, assuming that the number of dimensions is S .

Definition 2: The Dimensional weight $W_d(i)$ for dimension i is defined as follows:

$$\text{disp}(i) = \sum |a_i - b_i| \begin{cases} \text{for all } a, b \in \text{hotspot in the same group} \\ \text{for } i \in S \end{cases} \quad (2)$$

$$W_d(i) = \max.\text{disp} + 1 - \text{disp}(i) \text{ for } i \in 1 \text{ to } S \quad (3)$$

where $\max.\text{disp}$ is the maximum of all $\text{disp}(i)$'s.

From (3), dimensional weight is calculated for all dimensions. We find that a dimensional weight is larger if hotspots in the group are closer in this dimension.

C. Large-Area Correction

In this section, we discuss how to modify the dimensional weight to take into account the large-area effect within a single layout pattern.

As mentioned in the introduction, current detection methods must consider a layout pattern in a large area. Within that pattern, the inner area is more important than the outer area. For example, patterns in Fig. 9(a) and (b) are both hotspots which exhibit different characteristics in the outer area. Patterns in Fig. 9(a) and (c) are similar, but the pattern in Fig. 9(c) is a nonhotspot since the core area is different [10], [11], [14]. To take into account the large-area effect, we assign more weight for an inner area of a pattern. We define the range weight as follows.

Definition 3: The Range weight $W_r(i)$ for dimension i is a predefined constant.

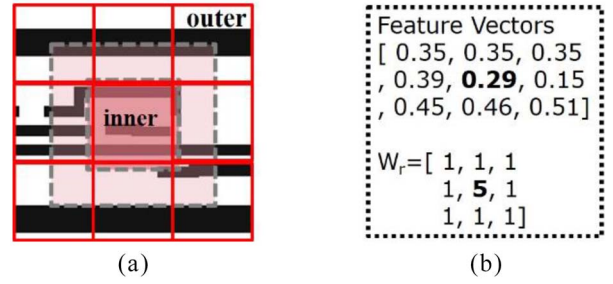


Fig. 10. Example of large-area correction method. (a) Region partition into grid. (b) Feature vectors and weight corresponding to (a).

For example, in Fig. 10(a), we can partition a rectangular region into nine grids, giving a weight to each grid according to its distance to the center grid. The region with the longest distance is assigned the lowest weight, while the shortest one is assigned the highest weight shown by the matrix in Fig. 10(b). Using the encoding method in Section II-B, we can encode the layout pattern in the grids as the matrix in Fig. 10(b) where each element of the matrix represents a dimension in the multidimensional space, called feature vectors.

To simultaneously consider the dimensional weight and the range weight, we modify the Cityblock distance to be the Cityblock weighted distance defined below.

Definition 4: We define Cityblock weighted distance D_{cw} to be the distance between two nodes considering the dimensional weight and the range weight in the multidimensional space as follows:

$$D_{cw}(a, b) = \frac{\sum_{i=1}^S (|a_i - b_i| * W_d(i) * W_r(i))}{\sum_{i=1}^S W_d(i) * \sum_{i=1}^S W_r(i)}, \text{ for } a, b \in \text{nodes}. \quad (4)$$

In this section, we use the Cityblock weighted distance D_{cw} in (4) to represent the “effective” distance between every two nodes in the multidimensional space. In other words, when we say the distance of two nodes which can be hotspots or nonhotspots, we refer to the Cityblock weighted distance of these two nodes.

D. Fuzzy-Region Growing

In this section, we describe how to obtain the fuzzy region of a hotspot group. An example of the fuzzy region is shown as the highlighted area in Fig. 11. Any layout pattern whose feature vector is located inside the fuzzy region will be considered as a hotspot. As a result, the larger the area of the fuzzy region, the fuzzier the hotspot group will be.

The fuzzy region of a hotspot group is the union of fuzzy regions of all hotspots. For example, in Fig. 11(a), the fuzzy regions of the three hotspots are the union of the three circles, each of which represents the fuzzy region of a hotspot.

The fuzzy region of a hotspot is determined by the “fuzzy distance” of the hotspot. We now describe how the fuzzy distance of a hotspot is defined. First, the fuzzy distance is not determined by a set of equations; rather, the value is determined by an iterative algorithm.

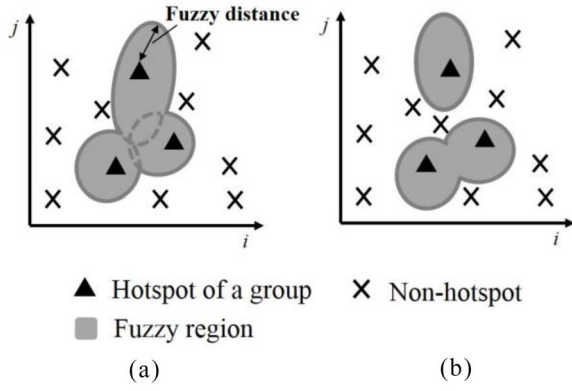


Fig. 11. Example of the fuzzy region of a group. (a) Legal hotspot group. (b) Illegal hotspot group.

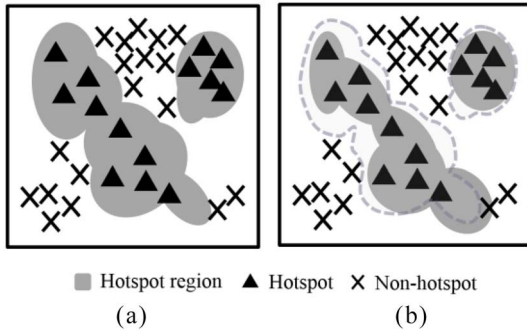


Fig. 12. Hotspot and nonhotspot region growing interaction. (a) Hotspot region. (b) Hotspot region considering non-hotspot region.

Definition 5: We define the Fuzzy distance, D_{fuzzy} of a hotspot to be the largest distance satisfying the following two rules. Again, the distance is measured as the Cityblock weighted distance in (4).

Rule 1: Any space within the fuzzy distance of a hotspot belongs to the fuzzy region of the hotspot.

Rule 2: The fuzzy region never covers a nonhotspot.

With these two rules, the fuzzy distance of a hotspot can grow gradually from zero until Rule 2 is violated, as we showed in Fig. 12(a). The oval charts indicate the hotspot growing regions. It is clear that when a growing region approaches the nonhotspot pattern (represented by a cross), the region will stop growing due to Rule 2. The fuzzy distance can guarantee that the hotspot region will not contain any nonhotspot. However, locations very close to high density nonhotspots are usually also nonhotspots. Thus, the region grows till the nonhotspot becomes too “aggressive” and causes a false alarm in the matching flow. Since Rule 2 is too restrictive, to obtain higher accuracy and lower false-alarm counts, we propose a new fuzzy-region growing method in the next section.

E. Fuzzy-Region Growing Considering Nonhotspots

In this section, we describe how to use the information of nonhotspots to limit the growing of hotspot regions. We use the concept of the density of nonhotspots. In general, when the density of a nonhotspot is high, it is likely the neighborhood is mainly nonhotspots. Our intention is to reduce the growing of

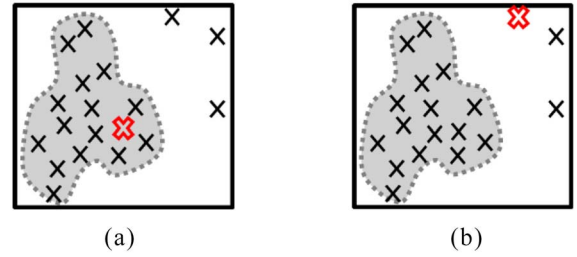


Fig. 13. Nonhotspot density based WNH computing. (a) Located in high-density part. (b) Located outside the high-density part.

a hotspot region into the region where there is a high density of nonhotspots. Consider the example, in Fig. 12(a), which shows the hotspot region obtained from the previous section. In Fig. 12(a), the hotspot region grows till it reaches nonhotspots. However, there are high densities of nonhotspots in the top middle and bottom left areas. For these two areas, our aim in Fig. 12(b) is to reduce the hotspot region in these two areas. In contrast, there are a few nonhotspots in the bottom right area. For that area, we intend to increase the hotspot region so that the highlighted area in Fig. 12(b) is smaller than the hotspot region in Fig. 12(a). Note that in the bottom right area, when growing the hotspot region, the hotspot region may cover nonhotspots. To avoid false alarm issues, we need to do checking for those nonhotspots.

To realize the concept of nonhotspot density, we define the following variable.

Definition 6: For a nonhotspot x , we define Nonhotspot weight $W_{\text{NH}}(x)$, which describes how it is centralized between x and other nonhotspots.

The value W_{NH} is calculated for each nonhotspot by (7) using (5) and (6). Basically, W_{NH} of a nonhotspot characterizes how close the nonhotspot is with respect to other nonhotspots. If a nonhotspot is close to many nonhotspots, its W_{NH} will be large; otherwise it is small. In Fig. 13, for example, the graph shows the distribution of high density nonhotspots where a cross symbol represents a nonhotspot. The highlighted nonhotspot in Fig. 13(a) is in the high density part; thus, the W_{NH} of the nonhotspot in Fig. 13(a) will be larger than that of the highlighted nonhotspot in Fig. 13(b), which is located outside the high-density part

$$\text{dispN}(x, i) = \sum |x_i - y_i| \begin{cases} \text{for all } y \in \text{nonhotspot} \\ \text{for } i \in S \end{cases} \quad (5)$$

$$W_{\text{ce}}(x, i) = \max(\text{dispN}(i) + 1 - \text{dispN}(x, i) \text{ for } i \in 1 \text{ to } S) \quad (6)$$

$$W_{\text{NH}}(x) = \sum_{i=1}^S W_{\text{ce}}(x, i). \quad (7)$$

With W_{NH} , we modify D_{fuzzy} in Definition 5 into D'_{fuzzy} where T_{fuzzy} is a scaling factor used to regulate the trade-offs of accuracy and false-alarm. The value is obtained by iteratively training through the whole hotspot fuzzy-region algorithm described in Section III

$$D'_{\text{fuzzy}} = \frac{T_{\text{fuzzy}}}{W_{\text{NH}}} * D_{\text{fuzzy}}. \quad (8)$$

Once all the fuzzy distances have been found, it is possible that the resulting fuzzy region can form several disconnected

Algorithm 1: Fuzzy-region growing**Inputs:** *Hotspots(H), Non-hotspots(NH), W_d, W_r, W_{NH}* **Outputs:** $D'_{fuzzy}()$ for all *Hotspots**Initialization:* $D'_{fuzzy}() = 0, global_D = 0;$ **for each** $a \in H$ **do** $D'_{fuzzy}(a) = 0;$ **while** all $b \in NH$, and $D_{cw}(a, b) > D'_{fuzzy}(a)$ **do** $D'_{fuzzy}(a)$ increases;**end while****end for****for each** $a \in H$ **do****if** $D'_{fuzzy}(a)$ cannot cover any H in the same group **then**

Return illegal;

end if**end for**Find $global_D = \max(D'_{fuzzy}());$ Base on Rule 2, let $D'_{fuzzy}()$ of alone hotspot grow until $global_D$

Fig. 14. Algorithm of fuzzy-region growing.

regions as in Fig. 11(b). When the resulting fuzzy region is not connected, we call the corresponding hotspot group illegal.

If there is an illegal hotspot group, we need to modify the group distance described in Section III-F to reduce hotspots in a group. Our fuzzy-region growing algorithm is outlined in Algorithm 1 of Fig. 14. There is no step size to increase the fuzzy distance in our algorithm, and the fuzzy distance is decided by equations in by Definitions 5 and 6 shown in Algorithm 1.

F. Determination of Group Distance

Note that we iterate Steps 1–4 to train our fuzzy-matching model in Fig. 6. In this section, we discuss when to terminate the iteration and how to determine the group distance mentioned in Section III-A.

The group distance is an important index to partition hotspots into groups. The larger the group distance is, the greater the number of hotspots that can be assigned to the same group. However, when there are many hotspots in the same group, we may encounter the illegal problem described in Section III-E. Initially, we set the value of the group distance to be one. We increase the value of the group distance in the next iteration.

We continue the process until the group distance causes the illegal problem.

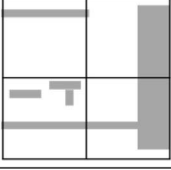
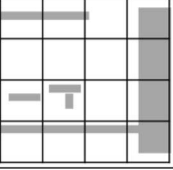
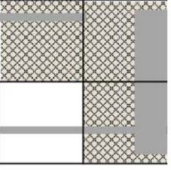
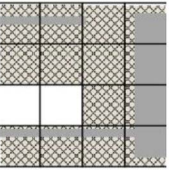
The group distance, fuzzy distances, and fuzzy regions all are trained and dynamically tuned in our fuzzy model. When more known hotspots and nonhotspots are provided as data for our training model, our model will have better accuracy and fewer false-alarm counts.

G. Pattern Recognizing

Given a fuzzy-matching model and an unknown pattern UN in the feature vector format, the pattern is recognized as a hotspot if any one of the following inequalities holds:

$$D_{cw}(UN, \{H\}_i) \leq \left\{ D'_{fuzzy} \right\}_i \text{ for } i = 1 \text{ to } \# \text{Hotspot.} \quad (9)$$

TABLE I
SIMILARITY BETWEEN HOTSPOT AND NONHOTSPOT OF
DIFFERENT GRID SCATTERING

Grid	2x2 grid	4x4 grid
Pattern 1		
Pattern 2		
Similarity	3/4	14/16

Otherwise, the unknown pattern is a nonhotspot. In other words, if the distance between the unknown pattern and any hotspot is less than or equal to the fuzzy distance of the corresponding hotspot, this unknown pattern is a hotspot; otherwise, the pattern is a nonhotspot.

IV. GRID REDUCTION

In this section, we first discuss how grid increase can affect the accuracy, false-alarm count, and run time. Then, we introduce our grid-reduction strategy for improving accuracy and run time.

A. Effect of Different Grids on Accuracy and False Alarm Count

In Table I, we illustrate how grid may affect the detection result. In column 2, we show a 2×2 grid. In column three, we show the 4×4 grid. Let us assume that density encoding is used. From this example, it can be found that two different patterns, pattern 1 and pattern 2, have 3/4 similarity when using a 2×2 grid. On the other hand, the two patterns have 14/16 similarity when the grid is increased. From this example, it can be seen that when grid is increased, if density based encoding is used, it is likely to increase the similarity between two patterns. Increases in similarity between two patterns tend to increase both the false-alarm count and accuracy.

Our experimental results, as explained below, have confirmed the above phenomenon. In Fig. 15, each bar shows different grids; the blocks in each bar indicate the accuracy (%) of benchmarks.

We show the cumulative accuracy for different grids. From the figure, it can be clearly seen that the accuracy increases are due to the increase of grids. These results are not surprising, since higher resolution provides more information to distinguish hotspots or nonhotspots. Based on our analysis from Tables IV and V, we find that when the dimensionality increase, the accuracy and false-alarm and runtime will increase.

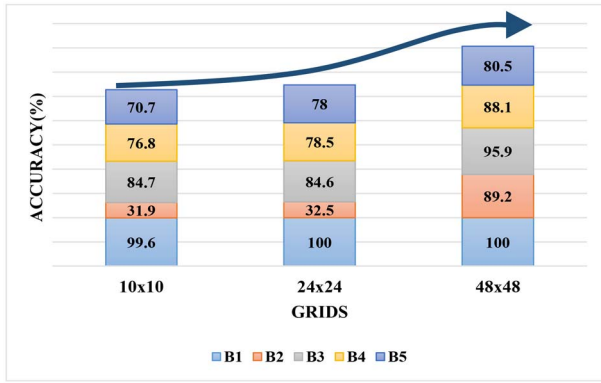


Fig. 15. Three different grids in comparison of accuracy.

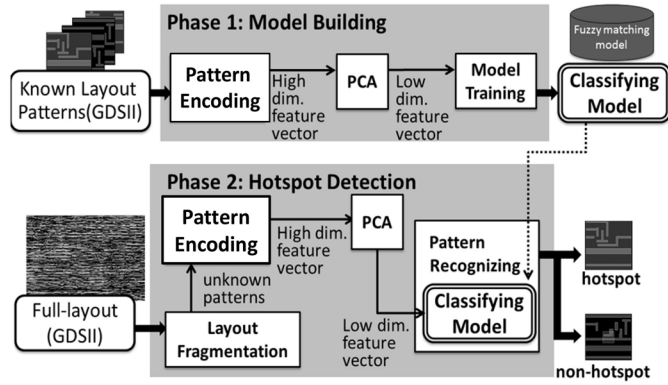


Fig. 16. Overall flow of our fuzzy-matching model.

B. PCA

Although higher encoding resolution obtains better results, higher resolution also increases the run-time significantly. To constrain the run-time, we applied a grid-reduction strategy to extract the representative PCA-features of the high dimensional space.

We used a dimension-reduction method called PCA. PCA is a technique to convert a set of possibly correlated variables into a set of linearly uncorrelated variables [1]. The input of the PCA tool is a set of high-dimensional vectors and PCA can automatically return a set of lower-dimensional vectors. Three purposes of applying PCA are to scatter all data in the multidimensional space, to make the significant dimension more representative, and to transform higher dimension data into lower dimensions. Fig. 16 shows our hotspot-detection flow consisting of PCA.

However, we cannot directly apply PCA for grid reduction. That is because in the hotspot-detection problem, the quantity of available nonhotspot patterns is usually much greater than that of hotspot patterns. If we use hotspot-detection flow directly with PCA, PCA will emphasize the representativeness of nonhotspot patterns in our fuzzy-matching model. Thus, our fuzzy-matching model distributes most of multidimensional space as a nonhotspot region. To resolve that problem, we propose to proliferate the hotspot patterns, using a strategy called virtual hotspot, described in the next section.

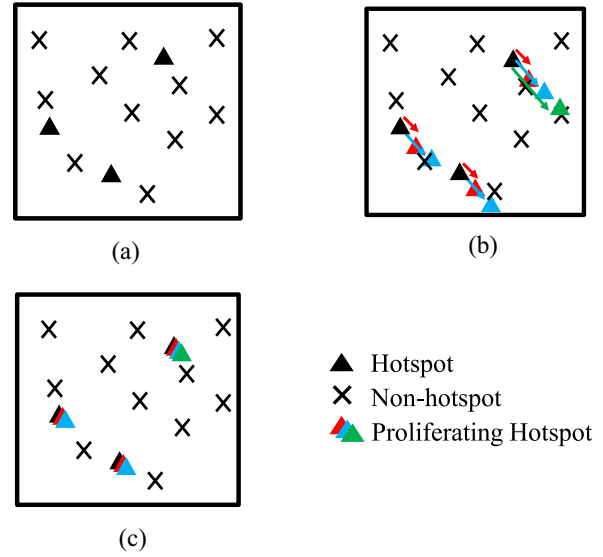


Fig. 17. Proliferation of hotspot patterns. (a) Initial dimensional space. (b) Process of proliferating hotspot. (c) Final dimensional space.

TABLE II
BENCHMARK LAYOUTS FOR EXPERIMENTS [11]

Benchmark	B1	B2	B3	B4	B5
Technology	32nm	28nm	28nm	28nm	28nm
Hotspot Count	223	508	1763	175	41
Layout Area (μm^2)	12516	106954	122565	82010	49583

C. Virtual Hotspot

The number of nonhotspot patterns is usually much more than that of hotspot patterns. Without using virtual hotspots, during PCA flow, PCA may wrongly considers the large quantity of nonhotspots as representative dataset. To avoid such wrong consideration of representative data, we propose to use virtual hotspot to balance both quantity of hotspots and nonhotspots.

Our basic purpose is to take the original hotspot as a basic hotspot sample, and duplicate the basic hotspot so that the number of hotspot patterns is close to the number of non-hotspot patterns. For example, in the beginning [Fig. 17(a)] we have 11 nonhotspot and three hotspot patterns. We duplicate original hotspot patterns for the purpose of balancing the quantity in Fig. 17(b). After duplicating, we still have 11 nonhotspot patterns but a total of ten hotspot patterns, including three original hotspot patterns and seven virtual hotspot patterns.

V. EXPERIMENTAL RESULTS

We implemented our fuzzy-matching model and the grid-reduction strategy in C and performed experiments on several benchmark layouts. The benchmark layouts consisted of four 28 nm circuit and one 32 nm circuits provided by 2012-ICCAD CAD contest [10], [11]. Table II shows the detailed information of these benchmark layouts. In the experiment, a set of training data and the corresponding clipped benchmark layouts provide the input data for our program. The qualities evaluated for comparisons are accuracy in (10),

TABLE III
HOTSPOT-DETECTION RESULTS WITHOUT PCA FLOW AND
NONHOTSPOT CONSIDERATION IN 10×10 GRID

Benchmark	B1	B2	B3	B4	B5	Avg.
Accuracy(%)	100	100	99.8	99.4	100	99.8
False-alarm	1949	10950	40117	3639	673	11465.6
Runtime(s)	11	286	414	102	49	172.4

TABLE IV
HOTSPOT-DETECTION RESULTS WITHOUT PCA FLOW IN 10×10 GRID

Benchmark	B1	B2	B3	B4	B5	Avg.
Accuracy(%)	96.5	31.9	84.7	76.8	70.7	72.1
False-alarm	1030	191	3943	348	69	1116.2
Runtime(s)	12	275	409	102	50	169.8

TABLE V
HOTSPOT-DETECTION RESULTS WITHOUT PCA FLOW IN 48×48 GRID

Benchmark	B1	B2	B3	B4	B5	Avg.
Accuracy(%)	100	89.2	95.9	88.1	80.5	90.7
False-alarm	1491	939	7636	390	100	2112.2
Runtime(s)	20	618	2487	175	58	671.6

TABLE VI
HOTSPOT-DETECTION RESULTS WITH PCA FLOW IN 48×48 GRID

Benchmark	B1	B2	B3	B4	B5	Avg.
Accuracy(%)	100	99.8	93.8	91.0	87.8	94.5
False-alarm	1714	4058	9486	1120	199	3315.4
Runtime(s)	11	287	417	102	49	173.2

false-alarm in (11), and runtime as in [9]. All experiments were performed on the Linux workstation with 2.66 GHz quad-core CPU and 8 GB memory

$$\text{Accuracy} = \frac{\#(\text{Correctly recognized Hotspot})}{\# \text{Real Hotspot}}. \quad (10)$$

$$\text{False alarm} = \#(\text{Non Hotspot but recognized as Hotspot}). \quad (11)$$

To demonstrate the effectiveness of the technique considering nonhotspots and Section IV grid reduction with PCA, (in Section IV), we provide incremental results with and without using these two techniques. In Table III, we show the results of [22], which do not use both techniques, in 10×10 grid. Table IV shows the results of using the technique considering nonhotspots, but without PCA, in 10×10 grid. Table V shows the results using the same approach as that for Table IV, but in 48×48 grid. Table VI shows the final results of our overall algorithm.

TABLE VII
HOTSPOT-DETECTION COMPARISON AMONG OUR METHOD,
ICCAD CONTESTS AND [24]

Methods	Accuracy	False-alarm	Runtime
Ours results (Based on Avg. column of Table VI)	94.5	3315.4	02m53s
1 st place	92.1	9742.6	05m45s
2 nd place	54.4	2260.4	34m22s
3 rd place	72.7	18352.2	03m20s
Yu [24]	92.7	6409.2	05m09s

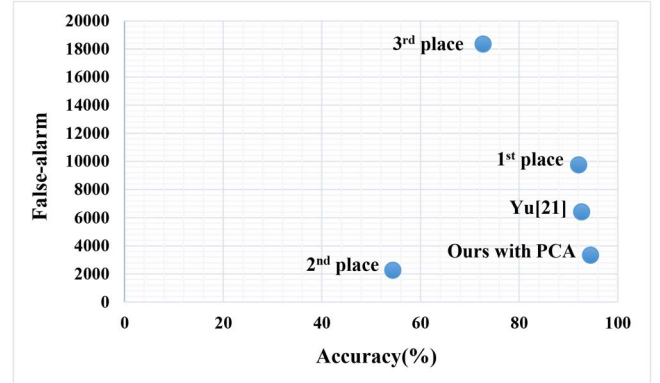


Fig. 18. Comparison of accuracy and false-alarm.

TABLE VIII
CONFIDENCE FACTOR UNDER DIFFERENT CONFIDENCE LEVEL

Confidence Level	95%	97%	<u>97.45%</u>	99%
Confidence Interval	4.72	5.23	<u>5.38</u>	6.2
Confidence Factor	0.88	0.97	<u>1</u>	1.15

In Tables III–VI, row 1 shows the name of a benchmark layout, row 2 shows the results of accuracy, row 3 shows the amount of false-alarm, and the last row shows the runtime. For example, considering benchmark B1 in Table VI, we achieve accuracy of 100%, i.e., all hotspots are identified; there are 1714 false-alarm patterns (wrongly reported to be hotspots when they are not); the run time for B1 is 11 s.

The run time for Table V shows the worst outcome. But Tables V and VI seem to have balanced results in accuracy and false alarm. In Table VII and Fig. 18, we compare our method with top three teams of ICCAD-2012 CAD contest and Yu *et al.* [24] in DAC 2013. We would like to mention that other contest participants did not have access nor possible feedback from the final blind test-cases to further improve their programs. As a result, it may not be strictly fair in the above comparison.

We calculate the confidence factor of our experiment and the results are shown in Table VIII. As can be found from the table, our confidence factor of 1 is located at when confidence level is 97.45%. In addition, regarding over-fitting, in the

journal version, we consider the effect of nonhotspot grouping, which allows us to exclude those regions occupied by nonhotspots.

We believe that the superiority of our results is due to the following reasons. First, we use a matching-based approach, which is inherently faster. Second, we apply the PCA that efficiently extracts representative features from layout patterns. And last, but most important, we design an efficient fuzzy-matching model to achieve high accuracy.

VI. CONCLUSION

In this paper, we present a fast lithography hotspot-detection flow with high accuracy and low false-alarm rate. A fuzzy matching model is designed to dynamically determine the appropriate fuzzy region around known hotspots. And a grid reduction is applied to extract representative features in the high dimensional space. On average, our model achieves 94.5% accuracy with 3315.4 false-alarm count, which is better than the results of previous works.

REFERENCES

- [1] H. Abdi and L. J. Williams, "Principal component analysis," in *Wiley Interdiscipl. Rev. Comput. Statist.*, vol. 2, no. 4, pp. 433–459, 2010.
- [2] B. A. Kahng, C.-H. Park, and X. Xu, "Fast dual graph-based hotspot detection," *Proc. SPIE*, vol. 6349, no. 14, 2006, Art. ID 63490H.
- [3] D. G. Drmanac, F. Liu, and L.-C. Wang, "Predicting variability in nanoscale lithography processes," in *Proc. 46th ACM/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, pp. 545–550, 2009.
- [4] D. Ding, A. J. Torres, F. G. Pikus, and D. Z. Pan, "High performance lithographic hotspot detection using hierarchically refined machine learning," in *Proc. 16th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Yokohama, Japan, 2011, pp. 775–780.
- [5] 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. 17th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Sydney, NSW, Australia, 2012, pp. 263–270.
- [6] D. Ding, X. Wu, J. Ghosh, and D. Z. Pan, "Machine learning based lithographic hotspot detection with critical-feature extraction and classification," in *Proc. IEEE Int. Conf. IC Design Technol.*, Austin, TX, USA, 2009, pp. 219–222.
- [7] H. Yao *et al.*, "Efficient range pattern matching algorithm for process-hotspot detection," in *Proc. IET Circuits Devices Syst.*, 2008, pp. 2–15.
- [8] ITRS Lithography team. (2011). *International Technology Roadmap for Semiconductors* [Online]. Available: <http://www.itrs.net/Links/2011ITRS/2011Chapters/2011Lithography.pdf>
- [9] 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) Special Session*, San Jose, CA, USA, 2012, pp. 349–350.
- [10] J. A. Torres. (2012, Nov. 30). *ICCAD-2012 CAD Contest in Fuzzy Pattern Matching for Physical Verification and Benchmark Suite* [Online]. Available: http://cad_contest.cs.nctu.edu.tw/CAD-contest-at-ICCAD2012/ICCAD_P3_results_teamno.pdf
- [11] J. A. Torres. (2012, Nov. 30). *Fuzzy Pattern Match for Physical Verification* [Online]. Available: http://cad_contest.cs.nctu.edu.tw/CAD-contest-at-ICCAD2012/problems/p3/p3.html
- [12] 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.
- [13] J.-Y. Wu, F. G. Pikus, A. Torres, and M. Marek-Sadowska, "Detecting context sensitive hotspots in standard cell libraries," *Proc. SPIE*, vol. 7275, Mar. 2009, Art. ID 727515.
- [14] J.-Y. Wu, F. G. Pikus, A. Torres, and M. Marek-Sadowska, "Rapid layout pattern classification," in *Proc. 16th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Yokohama, Japan, 2011, pp. 781–786.
- [15] J.-Y. Wu, 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.
- [16] J. Xu, S. Sinha, and C. 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, 2007, pp. 839–846.
- [17] J. Kim and M. Fan, "Hotspot detection on post-OPC layout using full-chip simulation-based verification tool: A case study with aerial image simulation," *Proc. SPIE*, vol. 5256, Dec. 2003, Art. ID 919.
- [18] J. Ghan, N. Ma, S. Mishra, C. Spanos, and K. Poolla, "Clustering and pattern matching for an automatic hotspot classification and detection system," *Proc. SPIE*, vol. 7275, Mar. 2009, Art. ID 727516.
- [19] M. Cote and P. Hurat, "Layout printability optimization using a silicon simulation methodology," in *Proc. Int. Symp. Quality Electron. Design*, Washington, DC, USA, 2004, pp. 159–164.
- [20] N. Nagase *et al.*, "Study of hotspot detection using neural networks judgment," *Proc. SPIE*, vol. 6607, May 2007, Art. ID 66071B-1.
- [21] 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.
- [22] 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. Design Autom. Conf. (DAC)*, Austin, TX, USA, 2013, pp. 1–6.
- [23] V. Dai, J. Yang, N. Rodriguez, and L. Capodiceci, "DRC plus: Augmenting standard DRC with pattern matching on 2D geometries," *Proc. SPIE*, vol. 6521, Mar. 2007, Art. ID 65210A.
- [24] Y.-T. Yu, G.-H. Lin, I. H.-R. Jiang, and C. Chiang, "Machine-learning-based hotspot detection using topological classification and critical feature extraction," in *Proc. Design Autom. Conf. (DAC)*, Austin, TX, USA, 2013, pp. 1–6.



Wan-Yu Wen received the B.S. degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2013, where she is currently pursuing the master's degree in computer science.

Her current research interests include low-power design improvement using dynamic voltage scaling implementation by statistical methods and power consumption optimization.



Jin-Cheng Li received the B.S. degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2013, where he is currently pursuing the master's degree in computer science.

His current research interests include network intrusion detection, graphics processing unit programming, and related computer-aided design techniques.



Sheng-Yuan Lin received the B.S. and M.S. degrees in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2011 and 2013, respectively.

He is currently a Digital IC Designer with MStar Semiconductor, Hsinchu.



Jing-Yi Chen received the B.S. and M.S. degrees in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2011 and 2013, respectively.

She is currently a Firmware Engineer with Silicon Motion Technology Corporation, Hsinchu.



Shih-Chieh Chang (SM'10) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1987, and the Ph.D. degree in electrical engineering from the University of California, Santa Barbara, CA, USA, in 1994.

He is currently a Professor and a Department Chair with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan. From 1995 to 1996, he was with Synopsys, Inc., Mountain View, CA. His current research interests

include logic synthesis, functional verification for state of charge (SoC), and noise analysis.

Dr. Chang was the recipient of the Best Paper Award at the Design Automation Conference in 1994. He is currently an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.