Fast Dual-Graph-Based Hotspot Filtering

Andrew B. Kahng, *Senior Member, IEEE*, Chul-Hong Park, *Student Member, IEEE*, and Xu Xu, *Student Member, IEEE*

***Abstract*—As advanced technologies in wafer manufacturing push patterning processes toward lower k_1 subwavelength printing, lithography for mass production potentially suffers from decreased patterning fidelity. This results in the generation of many *hotspots*, which are actual device patterns with relatively large critical-dimension and image errors with respect to on-wafer targets. Hotspots can be formed under a variety of conditions such as the original design being unfriendly to the resolution enhancement technique that is applied, unanticipated pattern combinations in rule-based optical proximity correction (OPC), or inaccuracies in model-based OPC. When these hotspots fall on locations that are critical to the electrical performance of a device, device performance and parametric yield can be significantly degraded. The golden verification signoff tool using a simulation-based approach has occupied the mainstream and has been able to accurately detect hotspots. However, this approach represents a runtime–quality tradeoff point that is high in quality but also high in runtime. There is also little point in trying to replace the golden signoff tool. We are motivated to develop a low-runtime "*prefilter*" that reduces the amount of layout area to be analyzed by the golden tool, without compromising the overall quality of hotspot finding. In this paper, we first describe a novel detection algorithm for hotspots induced by lithographic uncertainty. Our goal is to rapidly detect all lithographic hotspots without significant accuracy degradation. In other words, we propose a filtering method: as long as there are no "false negatives," i.e., we reliably obtain a superset of actual hotspots, then our method can dramatically reduce the layout area processed by golden hotspot analysis. Our hotspot detection algorithm includes layout graph construction, graph planarization, three-level bridging hotspot detection, and necking hotspot detection. We have tested our flow on several industry test cases. The experimental results show that our method is promising: for benchmark designs in 90-nm and 65-nm technologies, 100% of bridging and open hotspots are detected with few falsely detected hotspots. The average runtime of our method is more than 496× faster compared to the commercial tool.**

***Index Terms*—Bridging and necking, defocus, graph-based detection, hotspot, lithography.**

Manuscript received September 25, 2007; revised December 31, 2007 and March 13, 2008. Published August 20, 2008 (projected). A preliminary version of this work appeared in [3]. Extensions beyond [3] include methodologies for detection of line-end-type bridging hotspots as well as open hotspots. An introduction of open faults for necking and shortening is presented in Section I. The three types of pattern configurations for open hotspots are presented in Section II, which also presents a new necking hotspot detection heuristic. This draft of the paper also differs from the previous work [33] in Figs. 2–8, 10, 11, and 16. Tables I and II illustrate the results of our new validation approach described in Section III. This paper was recommended by Associate Editor H. Onodera.

A. B. Kahng is with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093-0404 USA, and also with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407 USA (e-mail: abk@ucsd.edu).

C.-H. Park is with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093-0407 USA (e-mail: chpark@vlsicad.ucsd.edu).

X. Xu is with Blaze DFM, Inc., San Diego, CA 92121 USA (e-mail: xuxu@blaze-dfm.com).

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.2008.927765

I. Introduction and Motivation

MOORE'S law continues to drive higher performance with smaller circuit features. Aggressive technology scaling has introduced new variation sources and made process variation control more difficult. For optical lithography, manufacturability is roughly defined by the k_1 factor from the Rayleigh equation [1]. Beyond the 45-nm CMOS technology node, even using a high-end optical exposure system such as immersion lithography with higher numerical aperture (NA), it is necessary to have a k_1 factor lower than 0.3. The primary risk posed by a lower k_1 is the likelihood of degradation of patterning fidelity on VLSI circuits. A lower k_1 could decrease patterning fidelity and result in the generation of many hotspots; a hotspot is an actual device pattern that has relatively large critical-dimension (CD) and image errors with respect to on-wafer targets. Hotspots include a variety of pattern deformations, e.g., line-end pullback (shortening), corner rounding, necking, and bridging [2]. Pullback is the shrinkage of geometries due to overdose at narrow line ends. Necking is a reduction in linewidth that is induced by a hammerhead or neighboring wide line. We separate hotspots into *open faults* for necking and shortening and *bridging faults* for corner rounding and bridging.

In particular, under ultralow k_1 conditions $(k_1 < 0.3)$, many hotspots may arise anywhere. Hotspots can form under a variety of conditions such as the original design being unfriendly to the resolution enhancement technique that is applied to the chip, pattern combinations unanticipated by rule-based optical proximity correction (OPC), or inaccuracies in model-based OPC. When these hotspots fall on locations that are critical to the electrical performance of a device, they can reduce the yield and performance of the device. It is therefore necessary to detect hotspots earlier in the layout design flow [4]–[6].

Park *et al.* [7] proposed a detection method for critical patterns (hotspots), using a design rule check tool. The approach is a rule-based detection that generates lookup tables with line and space parameters. However, for more complex patterns, the number of layout pattern parameters required to enable detection increases. As a result, the speed advantage of the rule-based approach is reduced. On the other hand, the simulation-based approach has occupied the mainstream and has been able to accurately detect hotspots [8], [9]. Furthermore, software solutions running on customized hardware platforms have

been developed so that aerial image simulation can be quickly carried out [10]. However, hotspots can change according to process conditions. Achieving the required accuracy of hotspot detection strongly depends on qualified optical and process models. Model generation corresponding to process variation represents a significant overhead in terms of validation, measurement, and parameter calibration.

For hotspot detection, there is typically only one golden physical verification signoff tool in the design flow, and even though analogous tools may be *qualified* for other junctures in the flow, there is little point in trying to replace the golden signoff tool. Rather, the golden signoff tool represents a runtime–quality tradeoff point that is high in quality but also high in runtime. The objective of our work is to develop a low-runtime "*prefilter*" that reduces the amount of layout area to be analyzed by the golden tool, without compromising the overall quality of hotspot finding.

In this paper, we describe a novel detection algorithm for hotspots induced by lithographic uncertainty. Our approach utilizes a layout-derived graph that reflects pattern-related CD variation. Our goal is to detect a set of potential lithographic hotspots within minutes without degrading accuracy (i.e., without missing any actual hotspots). The key intuition behind our approach is that generally CD variation is the result of "bad" patterns or effects. Hotspots are the location with relatively large CD variation caused by several spatially close "bad" patterns. We assume that this effect is accumulative and the total effect of several spatially related "bad" patterns can be represented by the total weight of one merged face (or merged dual node in the dual graph).

The main steps for bridging hotspot detection are given as follows.

- **Layout graph construction:** Given a layout L, the *layout graph* $G = (V, E_c \cup E_p)$ consists of nodes V, corner edges E_c, and proximity edges E_p. A face in the layout graph includes several close features and the edges between them. Edge weight can be calculated from a traditional 2-D model or a lookup table.
- **Graph planarization:** For any two crossing edges, delete the edge with the smaller weight.
- **Three-level hotspot detection:** 1) Edge-level detection finds the hotspot caused by two close features or "L-shaped" features; 2) face-level detection finds the pattern-related hotspots that span several close features; and 3) merged-face-level detection finds hotspots with more complex patterns. That is, we construct the dual graph G^D and sort the dual nodes according to their weights. We merge the sorted dual nodes (i.e., the faces in G) that share the same feature in sequence.

A local-wiring-density-based hotspot filter is used to reduce the number of falsely detected bridging hotspots. A normalized image log slope (NILS) has been used to evaluate the quality of pattern and susceptibility of the hotspot pattern to focus and exposure errors. The mask enhanced error factor (MEEF) for the dense pitch in general is higher than for isolated, and the higher MEEF causes lower NILS [11], [12] i.e., the lower the NILS in the dense pitch, the higher the probability of a bridging hotspot. As a result, the use of density filters may improve the detection accuracy of bridging hotspots.[1] Necking hotspot detection is done by comparing the total weight of each node with a given threshold value.

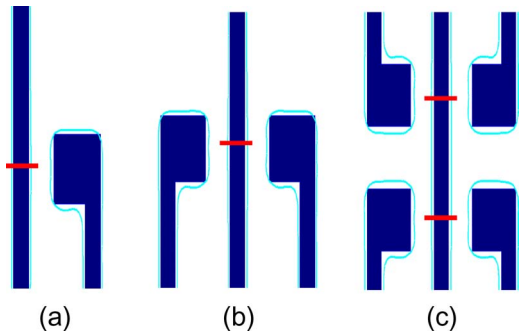

Fig. 1. Test patterns to evaluate CD variation induced by pattern complexity. (Red lines) CD measurement location. (Blue contours) Simulation results at worst case DOF. Two CD values are averaged in the case of (c).

The remainder of this paper is organized as follows. In Section II, we describe the problem formulation, dual-graph-based hotspot detection algorithm, and implementation details. Section III presents our evaluation flow and experimental results. We conclude in Section IV with directions for ongoing research.

II. Dual-Graph-Based Hotspot Detection

Recall that *hotspots* are the locations in the design where the magnitude of edge displacement is exceptionally large. In other words, hotspots are printed features whose CD variations are greater than a given threshold value.

A. Problem Formulation

We formulate the fast hotspot detection problem as follows.

Hotspot Detection Problem

Given: Layout L and threshold of CD variation that defines a hotspot.

Detect: Hotspots that may result in large CD variation.

To Minimize: The number of undetected hotspots and falsely detected hotspots.

The basic function for detection depends on process variations (i.e., defocus and exposure) and pattern parameters (i.e., width and space). To reduce the number of process conditions for hotspot validation, the effects of pattern complexity must be comprehended. Fig. 1 shows patterns with three different complexities: 1) one wide metal line [Fig. 1(a)]; 2) two wide metal lines [Fig. 1(b)]; and 3) four wide metal lines [Fig. 1(c)]. Fig. 2 shows that different complexities lead to different CD variations. Two CD values are averaged in the case of Fig. 1(c). The CD variation may also be affected by different process conditions. However, the patterns with more complex configuration, e.g., Fig. 1(c), have larger CD variations than the patterns with simple complexity, e.g., Fig. 1(a), at all process

[1]A dense pattern has a higher potential of a hotspot. However, the local wiring density filter cannot filter out all hotspots since the hotspot is also a complex function of the distance between two features, overlapped projection length, the widths of the two lines, etc.

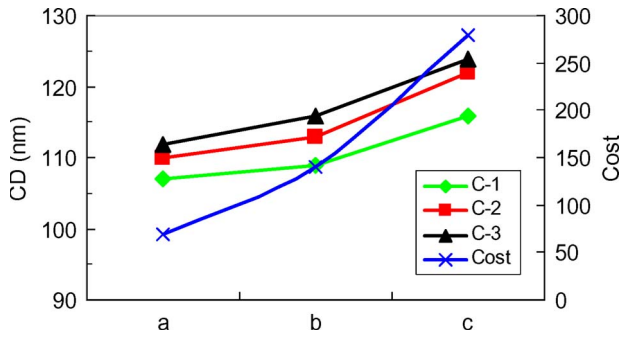

Fig. 2. Evaluation of CD variation and cost in a graph-based approach with various test patterns and process conditions. C-1: NA $= 0.85$ and $\sigma = 0.96/0.76$. C-2: NA $= 0.75$ and $\sigma = 0.75/0.55$. C-3: NA $= 0.75$ and $\sigma = 0.75/0.45$. a, b and c, respectively correspond to (a), (b), and (c) shown in Fig. 1.

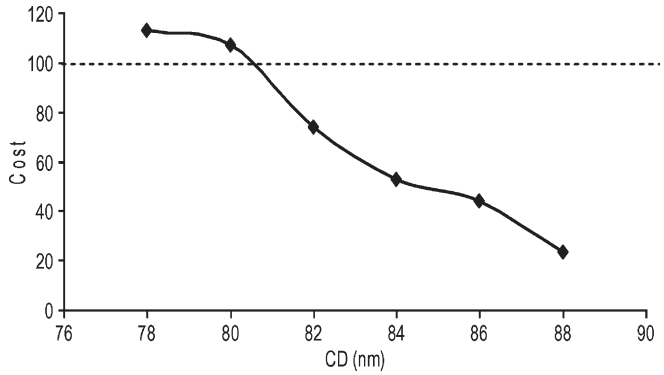

Fig. 3. Plot of average cost versus CD.

conditions. Therefore, our key observation is that the higher the pattern complexity, the higher the probability of a hotspot.

We propose to use a cost derived from our graph-based approach, described in detail in Section II-B, to represent the pattern complexity and, hence, the hotspot probability. For bridging hotspots, the cost, as shown in Fig. 2, can track the hotspot probability well according to the change of pattern complexity. Fig. 3 shows the plot of average cost versus CD for necking hotspots. In this example, the features with a printed CD smaller than 80 nm are viewed as necking hotspots. We can see that our proposed cost has good correlation with printed CD. The cost from the graph is thus closely related to hotspot probability.

We now propose a new graph-based hotspot detection method that is very fast and accurate. We group pattern-induced bridging-type CD variations into three cases.

1) **Corner-induced CD variation:** As shown in Fig. 4(a), two orthogonal connected features form a corner, which may lead to "corner-rounding" CD variations.
2) **Proximity induced CD variation:** As shown in Fig. 4(b) and (c), two close features may lead to "bridging" CD variations.
3) **Line-end-induced CD variation:** One line-end feature may lead to "bridging" CD variations. In fact, this effect can be treated as a special proximity-induced CD variation.

Pattern-induced open-type CD variations can also be grouped into three cases.

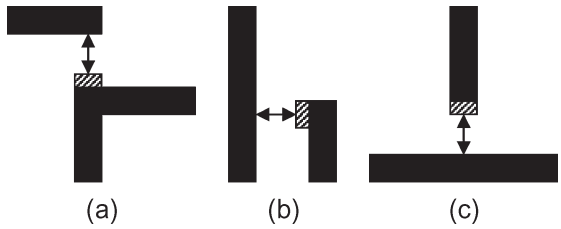

Fig. 4. Effects leading to bridging hotspots. (a) Corner-induced CD variation. (b) Proximity-induced CD variation. (c) Line-end-induced CD variation.

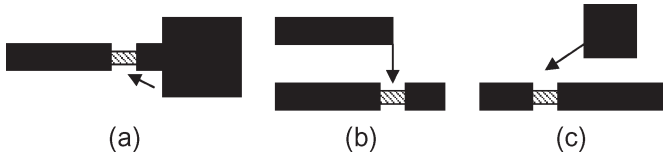

Fig. 5. Effects leading to open hotspots. (a) Wide-line-induced variation. (b) Line-end-induced CD variation. (c) Wide-line-proximity-induced CD variation.

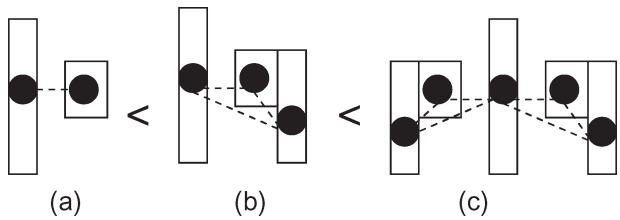

Fig. 6. Effects leading to hotspots. (a) Edge level. (b) Face level. (c) Merged-face level.

1) **Wide-line-induced variation:** As shown in Fig. 5(a), one wide line on a thin line may lead to "open" CD variations.
2) **Line-end-induced CD variation:** As shown in Fig. 5(b), the line end of one of two parallel features may lead to "open" CD variations.
3) **Wide-line-proximity-induced CD variation:** As shown in Fig. 5(c), one wide line close to a thin line may lead to "open" CD variations.

In lithography, a given hotspot may be the result of a single effect, as shown in Fig. 6(a), or the combination of several effects in an accumulative way, as shown in Fig. 6(b) and (c). The accumulative property of hotspots makes detection and filtering very difficult. In our approach, we try to formulate this accumulative effect with an iterative merging process. As noted above, our proposed hotspot detection for bridging flow is given as follows.

- **Layout graph construction:** Construct the layout graph $G = (V, E_c \cup E_p)$ for a given layout L, which consists of nodes V, corner edges E_c, and proximity edges E_p.
- **Graph planarization:** For any two crossing edges, delete the edge with the smaller weight.
- **Three-level hotspot detection:** Perform edge-level [Fig. 6(a)], face-level [Fig. 6(b)], and merged-face-level [Fig. 6(c)] detection to find hotspots with complex patterns.

For the necking hotspot detection, the total weight of each node is compared to the threshold value.

Input: Layout L, ε
Output: $G = (V, E_c \cup E_p)$
1. **For** all line
2. **If** (length $> l_0$)
3. Dived the line into segments whose length $< l_0$
4. **For** all line segment create a node v
5. **For** any two orthogonal connected line segments
6. connect two nodes with a corner edge $e \in E_c$
7. **For** any two closely proximate line segments
8. connect two nodes with a proximity edge $e \in E_p$

Fig. 7. Layout graph construction.

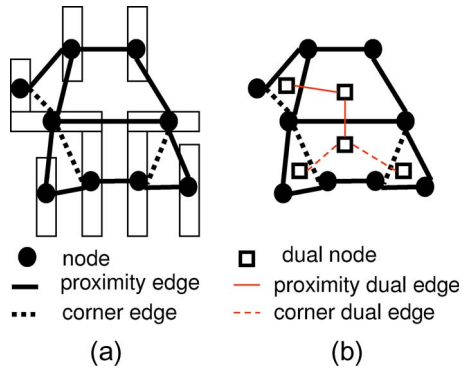

Fig. 8. (a) Example of a layout graph. (b) Its dual graph.

Intuitively, the single effects of "bad" patterns are represented by the weight of one edge, and the accumulative effect of several closely related effects is represented by the total weight of a merged face, which includes several connected edges.

We present each step in detail in the following sections.

B. Layout Graph Construction for Bridging Hotspots

To quickly detect the hotspots, the first step of our algorithm is to build a ***layout graph*** that reflects the pattern-related CD variation. As shown in Fig. 7, given a layout L, the ***layout graph*** $G = (V, E_c \cup E_p)$ consists of nodes V, corner edges E_c, and proximity edges E_p. The procedure is described as follows.

1) Every horizontal or vertical line is divided into line segments whose length is smaller than a threshold value l_0. For each line segment, create a node $v \in V$ located in the middle of the line segment.
2) For two orthogonal connected lines, connect two corresponding nodes with a corner edge $e \in E_c$ whose weight is a constant w_c.
3) Create a proximity edge $e \in E_p$ between two closely proximate lines having the same direction, where the weight of the edge is a function of the separation distance, the overlapped projection length, and the widths of the two lines. Since the line-end effect and all necking effects are special proximity-induced effects, we use the proximity edges for these effects with different weighting functions.

Fig. 8(a) shows an example of a layout graph for the layout. The layout graph has nine nodes representing nine lines, three corner edges (dashed edges), and ten proximity edges (solid edges).

One crucial issue is the edge weighting scheme. We propose both closed-form-formula-based and lookup-table-based weighting schemes. In the closed-form formula scheme, we assume that the weights of corner edges are a constant c. As shown in Fig. 9, the weights of the proximate edges are given by $[(w_1 \times w_2 \times f(l))/(d \times d)]$, where w_1 and w_2 are the widths of the two features, and d is the distance between the two features. $f(l)$ is the function of the length of the overlapped projection l, where $f(l) = 100$ if l is between -50 and 300 nm and $f(l) = 0$ otherwise, as shown in Fig. 10. This means that the hotspot can occur within a particular distance between corners of two features. Empirically, the wire bridging happens near the line end due to OPC correction. Therefore, we use a simple model in which the proximity effects only exist when there is small overlap between two lines. In addition, the proximate effect is intuitively more obvious for larger width features with a smaller distance. In a lookup-table-based weighting scheme, the weights of the proximate edges are determined by the feature widths, spacing, and length of the overlapped projection. Although the lookup-table-based weighting scheme is more accurate, it also brings overhead in parameter tuning. In this paper, we use only a closed-form-formula-based weighting scheme.

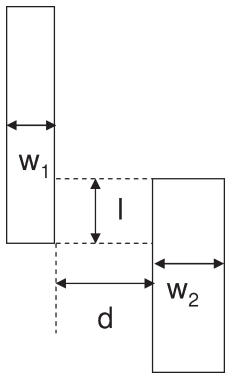

Fig. 9. Weights of the proximate edges.

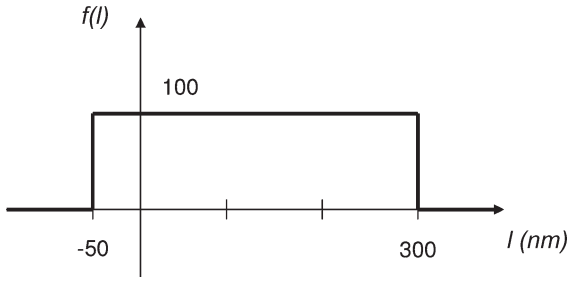

Fig. 10. Function of the length of the overlapped projection l.

C. Layout Graph Construction for Necking Hotspots

The layout graph G construction for necking hotspots is shown as follows.

1) Every horizontal or vertical line is assigned a node $v \in V$ located in the middle of the line.
2) For any wide line crossing a thin line, connect two corresponding nodes with an edge $e \in E$ whose weight is a constant w_0.

Input: $G = (V, E_c \cup E_p)$
Output: Dual graph G^d
1. **For** any two crossing edges in G
2. Delete the edge with smaller weight
3. Represent each face in G with a dual node
4. Represent each edge in G with a dual edge

Fig. 11. Dual-graph generation.

3) For any two parallel thin lines next to each other, create an edge $e \in E$ whose weight is a constant w_1.
4) For any wide line close to a thin line, connect two corresponding nodes with an edge $e \in E$ whose weight is a constant w_2.
5) For any two nodes v_1 and v_2 that connect to the same node v, connect v_1 and v_2 with a zero-weight edge $e \in E$.

The purpose of the last step is to ensure that there is always a face between two neighboring edges of one node.

D. Dual-Graph Generation

The next step is to convert the layout graph $G = (V, E_c \cup E_p)$ into its ***dual graph*** $G^D = (V^D, E_c^D \cup E_p^D)$. One fact is that the dual graph G^D exists if G is a *planar* graph, i.e., there is no crossing edge. Therefore, as shown in Fig. 11, we need to delete the edge with a smaller weight for any two crossing edges in G. In practice, the impact of deleted edges is negligible since the deleted edge has a smaller weight, which implies smaller CD variation effects. In addition, the number of deleted edges is relatively small since the edges are always added between neighboring or touching features, and it is unusual to have crossing edges. For our test cases, the number of deleted nonzero-weight edges is below 0.1%.

The **dual graph** G^D of the layout graph G is constructed by representing every face f of G with a dual node n whose weight is equal to the sum of the edge weights of f. An edge e that belongs to faces f_1 and f_2 in G is represented by a dual edge $e^d = \{n_1, n_2\}$ in G^D having the same weight as e.

We then calculate the total edge weight for each node in lines 2–4. A node is selected as a candidate hotspot if its total weight is greater than a threshold value ε. Please note that we use the total edge weight of a face (i.e., a dual node in the dual graph) for bridging hotspots since bridging hotspots are related to two or more features. For necking hotspots, we just use the total edge weight of a node since the necking hotspots are located in one feature (node). Therefore, we do not need to construct the dual graph for necking hotspots. The time complexity is dominated by the graph construction, which is $O(n)$.

E. Three-Level Hotspot Detection

The intuition behind our hotspot detection method is that we view the hotspot as the result of the combination of several locally related "bad" patterns. With the assumption that the CD variation effect is cumulative, the effect can be reflected by the dual-node weight, i.e., the total edge weight of one face. However, a hotspot may also relate to the lines of several faces. Therefore, we need to consider dual nodes merging to capture all possible hotspots. Our proposed iterative dual-node merging heuristic is shown in Fig. 12. The heuristic starts with the layout graph G construction in line 1. We perform edge-level detection in lines 2–4. We then delete the edge with the smaller weight for any pair of crossing edges to make G a planar graph and construct the dual graph G^D from G. In lines 7–9, we perform face-level hotspot detection. Finally, we perform merged-face-level detection by sorting dual nodes according to their weights and sequentially merging spatially adjacent dual nodes (i.e., the dual nodes connected with dual edges). Intuitively, nodes with a larger weight represent the location with a higher CD variation. The weight of the merged node is equal to the sum of dual-node weights minus the dual-edge weight.[2] The purpose of deleting found hotspots in lines 4, 9, and 15 is to eliminate redundant hotspot detection. Since the simulation window, defined as 4×4 μm for each hotspot, is large enough to cover any possible neighboring hotspots, it is not necessary to include any spatially close hotspots in the final hotspot set. A local-wiring-density-based hotspot filter is used to reduce the number of falsely detected hotspots. In this filter, we first find the center of the hotspots, and then, the local wiring density is the density within the box of 1 μm $\times$ 1 μm around the center. After the hotspot detection, a bounding box that covers all features in the edge/faces is drawn as the hotspot marker.

Input: Layout L, ε_0, ε_1, ε_2, d_0
Output: A list of hotspots in L
1. Construct layout graph G from L, $S \leftarrow \emptyset$
2. **For** all edges e whose weight $> \varepsilon_0$
// **edge-level detection**
3. $S \leftarrow S \cup \{e\}$
4. Delete e from G
5. Perform graph planarization to delete one of crossing edges
6. Construct dual graph G^D from G
7. **For** all dual nodes n whose weight $> \varepsilon_1$
// **face-level detection**
8. $S \leftarrow S \cup \{n\}$
9. Delete n from G^D
10. **While** ($\exists$ dual nodes can be merged)
// **merged-face-level detection**
11. Sort all dual nodes according to weight
12. Sequentially, merge each node with all spatially adjacent dual nodes
13. **If** (the weight of the merged node $n_c > \varepsilon_2$)
14. $S \leftarrow S \cup \{n_c\}$
15. Delete n_c from G^D
16. **For** all hotspots in S
17. **If** (local wiring density $< d_0$)
18. Remove it from S

Fig. 12. Iterative dual-node merging heuristic for bridging hotspots.

The time complexity for graph construction is $O(n)$, where n is the number of features. The edge-level hotspot detection (lines 2–4) is $O(n)$. Dual-graph generation time is $O(m \log m)$, where m is the number of edges. The face-level hotspot generation (lines 7–9) time is $O(k)$, where k is the number of faces. The merged-face-level hotspot generation (lines 10–15) time is $O(k \log k)$. The density-based filter time is linear with respect to the number of detected hotspots.

[2]In current mode, we use a simple model that assumes that the accumulative effect can be represented by the sum of weights. Although more complicated models such as weighted sum may lead to better solution quality, there will be more parameters to be extracted and tuned.

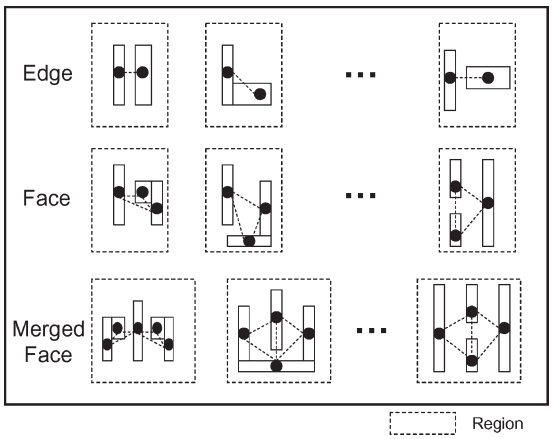

Fig. 13. Illustration of a test layout used to calibrate our model.

F. Model Parameter Extraction

One key issue with our proposed method is how to determine the model parameters. We use the following steps to build the model.

1) Create a test layout, as illustrated in Fig. 13, that includes a list of isolated regions. Within each region, there is a pattern with varying line width, line space, and overlap length.
2) Construct dual graph on the test layout. Each isolated region is marked as edge, face, or merged face, and the total weight for each region is calculated.
3) Run golden simulation tools on the test layout to detect all hotspots.
4) ε_0 is chosen as the smallest weight among all of the edge regions that have one or more hotspots.
5) The values of ε_1 and ε_2 can be similarly determined. d_0 is the smallest wiring density of all regions having hotspots.

III. Experimental Results

In this section, we empirically test our approach on several real designs within a standard industry flow using leading-edge tools. We measure the number of truly detected hotspots and falsely detected hotspots, relative to area and runtime.

A. Experimental Setup

For the evaluation according to various process conditions, as shown in Table I, we use one benchmark design in our experiments, which is the *alu*128 core with 8700 instances from *Artisan* libraries in a 90-nm technology using *Synopsys Design Compiler v2003.06-SP1* [13]. The chip size is $335\ \mu\text{m} \times 285\ \mu\text{m}$. The synthesized netlist is placed with a row utilization of 70% using *Cadence SOC Encounter v3.3* [14]. The netlist of the design has been obtained from *OpenCores* [15]. For the validation according to various metal layers as shown in Table II, we use five benchmark designs from major chipmakers in 65-nm technology. We have implemented our proposed iterative dual-node merging heuristic in C++.

On the lithography side, *CalibreOPC and CalibreORC* from *Mentor Graphics Calibre v9.3 5.11* [16] are used for model-based OPC and optical rule check (ORC), respectively. For a 90-nm design, simulation is performed with wavelength $\lambda = 193$, NA $= 0.75$, and annular aperture $\sigma = 0.75/0.50$. We used a 0-μm DOF model and 0.35 aerial image threshold for OPC and then evaluated the OPCed layer under the various values of DOF and threshold. For five 65-nm designs, simulations are performed with wavelength $\lambda = 193$, NA $= 0.85$, and annular aperture $\sigma = 0.96/0.76$. OPC and ORC are performed with a 0-μm DOF model and 0.28 aerial image threshold and a 0.1-μm DOF model and 0.30 aerial image threshold, respectively.

B. Experimental Results

We use the layout sizing technique to mark the hotspots and compare simulation-based detection with our dual-graph-based detection (DG). Simulation-based detection makes fragments on the pattern and decides whether each fragment is a hotspot based on the magnitude of the edge displacement. As a result, there may be several marked layers on a line end and a corner of a pattern. On the other hand, our graph-based detection marks all patterns affecting hotspot, and hence, it is difficult to compare hotspots of simulation-based with graph-based methods. We size all layers marked as a hotspot after ORC by 0.5 μm.[3] Then, all sized layers are merged into one layer, which includes hotspots and is used for comparison. Fig. 14 shows an example of a hotspot marking layer that is the result of merging of two layers in the ORC result.

We also notice that bridging hotspots depend on the local pattern density. To reduce the number of falsely detected bridging hotspots, a filter based on local pattern density has been used. Fig. 15 shows the results of bridging hotspot detection: 1) no hotspot pattern [Fig. 15(a)] and 2) a hotspot pattern [Fig. 15(b)]. The pattern in a denser region is a bridging hotspot, while the same pattern in a sparse region is not a bridging hotspot. The results of the dual-graph-based method match the simulation-based method. In addition, we show an example of necking hotspot detection in Fig. 16. The necking hotspot causes a reduction in the linewidth due to combined effect of pullback and corner rounding at wide lines. The dual-graph-based method can thus detect hotspots induced by pattern density and wide lines, which cannot be achieved by a rule-based approach.

For a 90-nm design, we evaluated the hotspot detection under four different process conditions. The number of hotspots can increase with a higher threshold (exposure dose) and defocus. The values of ε_0, ε_1, ε_2, and d_0 in Fig. 12 are chosen according to the different conditions shown in Table I. We can see that only the values of ε_2 need to be changed if only exposure time is changed, while we need to change all parameters if defocus is changed. Our proposed hotspot detection method achieves good accuracy (100% of hotspots are detected) with smaller false detection overhead. Our approach can also track the hotspot well according to the change of process condition. The relative area *(Rel. area)*, as shown in Table I, is formulated as $\#\text{hotspot} * (\text{simulation window})/(\text{chip area})$, where the simulation window defines $4 \times 4\ \mu$m. Table I shows that 100%

[3]The size amount is reasonable when we consider proximity range 0.6 μm at 90-nm technology.

TABLE I

RESULT OF BRIDGING HOTSPOT DETECTION FOR 90-nm TEST CASES. THE RUNTIME OF OUR METHOD IS THE SUM OF GRAPH GENERATION, HOTSPOT DETECTION, AND HOTSPOT REPORT. IT IS ACHIEVED THAT 100% HOTSPOTS ARE DETECTED WITH FEW FALSELY DETECTED HOTSPOTS. THE AVERAGE RUNTIME FOR FOUR TEST CASES IS MORE THAN 287× FASTER COMPARED TO THE ORC TOOL

Simulated Condition	# Hotspots				Runtime (sec)		Parameters			
	ORC	Detected	Falsely Detected	Rel. Area(%)	ORC	Dual Graph	ε_0	ε_1	ε_2	d_0
Defocus(μm)=0.1, ET=0.36	17	17	13	0.5	690	1.37	0.7	0.36	0.67	0.17
Defocus(μm)=0.1, ET=0.37	21	21	22	0.72	690	1.52	0.7	0.36	0.64	0.17
Defocus(μm)=0.1, ET=0.38	25	25	46	1.19	690	2.32	0.7	0.36	0.625	0.17
Defocus(μm)=0.2, ET=0.38	152	152	1291	24.18	690	4.38	0.65	0.34	0.537	0.14
Average	53.75	53.75	343	6.65	690	2.4				

TABLE II

COMPARISON OF HOTSPOT DETECTION EFFICIENCY OF ORC AND OUR PROPOSED METHOD AT 65-nm NODE. THE VALUES OF ε_0, ε_1, ε_2, AND d_0 ARE 0.55, 0.83, 0.91, AND 0.18, RESPECTIVELY. FILTERED ORC + DG REPRESENTS THE TOTAL RUNTIME, WHICH IS THE SUM OF RUNTIME FOR OUR DG AND ORC RUNTIME FOR THE HOTSPOT AREA FILTERED BY OUR DG. FOR BOTH BRIDGING AND OPEN HOTSPOTS, 100% DETECTION IS ACHIEVED WITH FEW FALSELY DETECTED HOTSPOTS. THE AVERAGE RUNTIME FOR FIVE TEST CASES IS MORE THAN 496× FASTER COMPARED TO THE COMMERCIAL TOOL. REL. AREAS OF BRIDGING AND OPEN FAULTS ACHIEVE 0.04% ∼ 14.6% AND 0.002% ∼ 0.23% AREAS, RESPECTIVELY

Testcase	# instance	Layer	Bridging HS				Open HS				Runtime (sec)		
			ORC	Det.	False	Rel. Area (%)	ORC	Det.	False	Rel. Area (%)	ORC	DG	Filtered ORC + DG
1	5M	M2	8	8	328	0.13	6	6	7	0.005	50823	125	194
		M3	23	23	1127	0.46	440	440	158	0.23	50912	328	679
		M4	0	0	0	0	29	29	3	0.01	40431	19	23
2	6M	M2	4	4	697	0.28	4	4	4	0.003	51840	91	238
		M3	1	1	106	0.04	114	114	61	0.07	51807	85	142
		M4	0	0	0	0	44	44	8	0.02	45644	20	29
3	4M	M3	0	0	0	0	4	4	2	0.002	42761	5	6
4	42K	M2	10	10	125	14.6	0	0	0	0	964	2	143
5	27K	M2	1	1	14	5.8	0	0	0	0	674	1	40
Average			5.22	5.22	266.44	2.368	71.22	71.22	27.0	0.038	37317.3	75.1	166

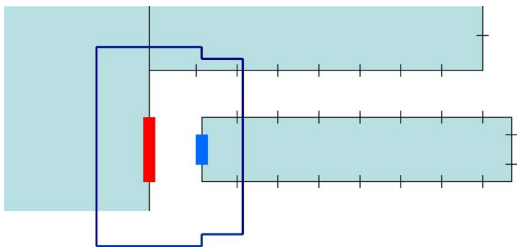

Fig. 14. Example of a layer marking hotspot patterns.

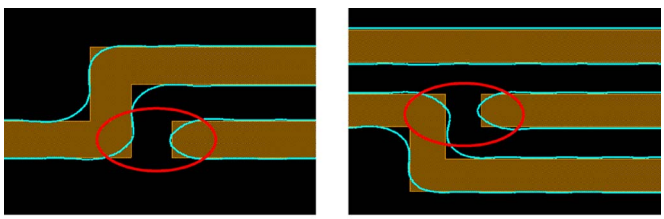

Fig. 15. Results of hotspot detection: comparison of (a) no hotspot patterns versus (b) hotspot patterns.

hotspot detection is achieved with a small number of falsely detected hotspots. The *Rel. areas* are from 0.5% to 24.2%. The average runtime (including graph generation, hotspot detection, and hotspot report) for four test cases is more than 287× faster compared to the ORC tool.

For five 65-nm designs, we detect 5 bridging and 71 open hotspots on the average, which perfectly matches the results of golden commercial tools. The parameter values of ε_0, ε_1, ε_2, and d_0 are 0.55, 0.83, 0.91, and 0.18, respectively. For all designs, we can use the same parameters for the hotspot detection if lithography conditions are not changed. For necking

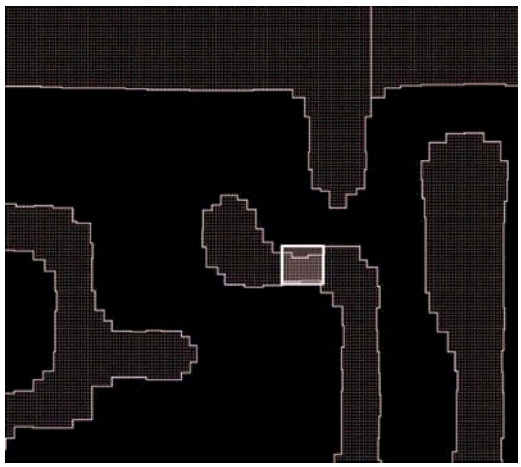

Fig. 16. Example of open hotspot detection.

hotspots, we set the weight of each feature as 0.3 and ε as 0.9. *Filtered ORC + DG* represents the total runtime, which is the sum of runtime for our dual-graph-based detection and ORC runtime for the filtered hotspot area.[4] Average runtime of our method is more than 496× faster compared to the commercial tool. The average runtime of *Filtered ORC + DG* is also more than 224× faster compared to the commercial tool. The *Rel. areas* of bridging and open faults are 0.04% ∼ 14.6% and 0.002% ∼ 0.23% of areas, respectively. We thus save between 85.4% and 99.9% of areas for hotspot rechecking. The results are summarized in Table II.

[4]The commercial tool needs to confirm whether patterns detected by our DG are truly detected hotspots or falsely detected hotspots. The total runtime includes our DG and ORC rechecking over the hotspot area filtered by our detection approach.

IV. Conclusion

With the continued shrinkage of minimum feature sizes, hotspots, i.e., printed images with large CD variation, present an important threat for manufacturing yield. Therefore, it becomes more and more important to quickly and accurately detect the hotspots in a layout. In the current design flow, there is only one golden physical verification signoff tool, and there is little point in trying to replace the golden signoff tool. However, the golden signoff tool is high in quality but also high in runtime. The goal of our work is to develop a low-runtime "*prefilter*" that reduces the amount of layout area to be analyzed by the golden tool, without compromising the overall quality of hotspot finding.

In this paper, we first describe a novel fast dual-graph-based lithographic hotspot detection algorithm without significant accuracy degradation. For four test cases in 90-nm technology, our method can detect all bridging hotspots, while the average runtime improvement is more than 287× faster compared to the commercial tool. Areas that may be checked again by commercial golden tools save between 75% and 99.5%. For five benchmark designs in 65-nm technology, we achieve that 100% bridging and open hotspots are detected with few falsely detected hotspots. The average runtime of our method is more than 496× compared to the commercial tool.

Our ongoing work includes the fast hotspot detection engine in a detailed router to improve the yield. We also plan to explore the idea of a "*corner-density-based filter*," which may reduce falsely detected hotspots since more vertices within a proximity radius provides the higher probability of a hotspot. This would be a complement to the currently proposed graph-based detection approach.

References

[1] B. J. Lin, "The k_3 coefficient in nonparaxial λ/NA scaling equations for resolution, depth of focus, and immersion lithography," *J. Microlithogr. Microfabr. Microsyst.*, vol. 1, no. 1, pp. 7–12, Apr. 2002.

[2] V. Singhal, C. B. Keshav, K. G. Sumanth, and P. R. Suresh, "Transistor flaring in deep submicron—Design considerations," in *Proc. 15th Int. Conf. VLSI Des.*, 2002, vol. 7695, pp. 299–304.

[3] A. B. Kahng, C.-H. Park, and X. Xu, "Fast dual-graph based hot-spot detection," in *Proc. 27th BACUS Symp. Photomask Technol. Manag.*, 2006, vol. 6281, p. 628 104.

[4] K. Hashimoto, S. Usui, S. Nojima, S. Tanaka, E. Yamanaka, and S. Inoue, "Hot spot management in ultra-low k1 lithography," in *Proc. SPIE—Microlithography*, 2005, vol. 6156, pp. 1207–1219.

[5] S. Nojima, S. Mimotogi, M. Itoh, O. Ikenaga, S. Hasebe, K. Hashimoto, S. Inoue, M. Goto, and I. Mori, "Flexible mask specifications," in *Proc. SPIE 22nd Annu. BACUS Symp. Photomask Technol.*, 2002, vol. 4889, pp. 187–196.

[6] M. Cote and P. Hurat, "Standard cell printability grading and hot spot detection," in *Proc. Int. Symp. Quality Electron. Des.*, 2005, pp. 264–269.

[7] C.-H. Park, Y.-H. Kim, J.-S. Park, K.-D. Kim, M.-H. Yoo, and J.-T. Kong, "A systematic approach to correct critical patterns induced by the lithography process at the full-chip level," in *Proc. SPIE—Microlithography*, 1999, vol. 3679, pp. 622–629.

[8] E. Sahouria, Y. Granik, N. Cobb, and O. Toublan, "Full-chip process simulation for silicon DRC," in *Proc. Int. Conf. MSM*, 2000, pp. 32–35.

[9] M. Cote and P. Hurat, "Layout printability optimization using a silicon simulation methodology," in *Proc. Int. Symp. Quality Electron. Des.*, 2004, pp. 159–164.

[10] Y. Cao, Y.-W. Lu, L. Chen, and J. Ye, "Optimized hardware and software for fast, full chip simulation," in *Proc. SPIE—Microlithography*, 2005, vol. 5754, pp. 407–414.

[11] K. Yeh and W. Loong, "Simulations of mask error enhancement factor in 193 nm immersion lithography," *Jpn. J. Appl. Phys.*, vol. 45, no. 4A, pp. 2481–2496, 2006.

[12] M. V. Plat, K. B. Nguyen, C. A. Spence, C. F. Lyons, and A. Wilkison, "Impact of optical enhancement techniques on the mask error enhancement function (MEEF)," in *Proc. SPIE—Microlithography*, 2000, vol. 4000, pp. 206–214.

[13] *Synopsys Design Compiler User's Manual*, Synopsys, Mountain View, CA, 2003. [Online]. Available: http://www.synopsys.com/

[14] *Cadence Encounter User's Guide*, Cadence, San Jose, CA, 2006. [Online]. Available: http://www.cadence.com

[15] [Online]. Available: http://www.opencores.org/

[16] *Mentor Graphics Calibre RET User's Manual*, Mentor, San Jose, CA, 2006. [Online]. Available: http://www.mentor.com

[17] [Online]. Available: http://www.insightful.com/

Andrew B. Kahng (SM'07) received the A.B. degree in applied mathematics (physics) from Harvard College, Cambridge, MA, and the M.S. and Ph.D. degrees in computer science from the University of California at San Diego (UCSD), La Jolla.

He joined the Department of Computer Science, University of California, Los Angeles (UCLA), as an Assistant Professor in July 1989 and became an Associate Professor in July 1994 and a Full Professor in July 1998. In January 2001, he joined UCSD as a Professor in the Department of Computer Science and Engineering and Department of Electrical and Computer Engineering. He served as an Associate Chair of the Department of Computer Science and Engineering, UCSD, from 2003 to 2004. He has been an executive committee member of the MARCO Gigascale Systems Research Center since its inception in 1998. In October 2004, he cofounded Blaze DFM, Inc., San Diego, CA, and served as the Chief Technical Officer of the company until resuming his duties at UCSD in September 2006. He is the author of more than 300 journal and conference papers. Since 1997, his research in IC design for manufacturability has pioneered methods for automated phase-shift mask layout, variability-aware analyses and optimizations, CMP fill synthesis, and parametric-yield-driven and cost-driven methodologies for chip implementation.

Prof. Kahng was the Founding General Chair of the 1997 ACM/IEEE International Symposium on Physical Design, a Cofounder of the ACM Workshop on System-Level Interconnect Prediction, and defined the physical design roadmap as a member of the Design Tools and Test Technology Working Group (TWG) for the 1997, 1998, and 1999 renewals of the International Technology Roadmap for Semiconductors. From 2000 through 2003, he was the Chair of both the U.S. Design TWG and the Design International TWG (ITWG). He continues to serve as a Cochair of the Design ITWG. He is a recipient of NSF Research Initiation and Young Investigator Awards, 11 best paper nominations, and six best paper awards.

Chul-Hong Park (S'03) received the B.S. and M.S. degrees in mathematics from Kyung Hee University, Seoul, Korea, in 1992 and 1994, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California at San Diego (UCSD), La Jolla, in 2008.

From 1994 to 2003, he was with CAE, Semiconductor R&D Center, Samsung Electronics, Yongin, Korea. He is the author of more than 30 papers. He is also the holder of ten patents. His research interests include lithographic design for yield and VLSI design–manufacturing interface.

Dr. Park was the recipient of the 1998 Honorable Outstanding Researcher Award and two Best Paper Awards from Samsung Electronics.

Xu Xu (S'01) received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1998 and the Ph.D. degree in computer science from the University of California at San Diego (UCSD), La Jolla, in 2006.

He was with Ammocore Technology, Inc., Santa Clara, CA, in 2002. He was with Synopsys, Inc., Mountain View, CA, in 2004. He is currently a Senior Software Engineer with Blaze DFM, Inc., San Diego, CA. He is the author of more than 20 papers on VLSI physical design, DNA array design, and IC manufacturing cost minimization.