Machine Learning based Lithographic Hotspot Detection with Critical-Feature Extraction and Classification

Duo Ding, Xiang Wu, Joydeep Ghosh and David Z. Pan
ECE Dept. Univ. of Texas at Austin, Austin, TX 78712
{ ding, dpan }@cerc.utexas.edu, { ghosh, xwu }@ece.utexas.edu

ABSTRACT

In this paper, we present a fast and accurate lithographic hotspot detection flow with a novel MLK *(Machine Learning Kernel)*, based on *critical feature* extraction and classification. In our flow, layout binary image patterns are decomposed/analyzed and critical lithographic hotspot related features are defined and employed for low noise MLK supervised training. Combining novel *critical feature* extraction and MLK supervised training procedure, our proposed hotspot detection flow achieves over 90% detection accuracy on average and much smaller false alarms (10% of actual hotspots) compared with some previous work [9,13], without CPU time overhead.

1. INTRODUCTION AND PREVIOUS WORK

With the rapid advances of semiconductor process technology [4, 5, 16], the minimum feature size of modern IC is becoming smaller and smaller than the actual lithographic wavelength. In order to bridge such a wide gap between design demands and manufacturing limitations, various popular *manufacturability aware design* techniques have been proposed and applied towards high fabrication yield and resilience for designs with scaling-down feature sizes [3,6,7,12, 14,19]. Successful *design for manufacturability* (DFM) techniques ensure high fabrication yield by incorporating manufacturability aware models into design stage to avoid potentially problematic patterns (usually referred to as process hotspots).

A typical DFM flow follows an incremental procedure, with an embedded simulator for hotspot detection. On one hand, approaches that employ lithographic simulations [10, 14] are precise yet costly to run due to the complex calculations involved; on the other hand, approaches that utilize pattern/graph matching techniques [9,17,18] suffer from high false alarms (upto over 60 times false alarms than actual hotspots [9]) or other issues since hotspot patterns are hard to apply - too many patterns lead to high over-estimate rate and too few patterns result in low accuracy rate.

In the meantime, there are works that start incorporating modern data mining methods towards fast and accurate hotspot detection tasks. A neural network judgment based detection flow was proposed in [13], where hotspot image patterns were used for training an artificial neural network (ANN) kernel. Also in [11], data mining algorithms are developed for hotspot pattern clustering. While these early attempts have shown promising potentials for data mining applications, there are still limitations to overcome, such as high ANN training noise and low hotspot detection accuracy, etc.

To better address issues above, we first introduce the concept of critical hotspot feature as a compact representation of the original pixel based design layouts. Compared with 2D pixel images used in [11, 13], critical-feature reduces the dimension of the original samples and filters out detection noise data. With an embedded Machine Learning Kernel (MLK) [8,15] trained with the extracted critical feature, our flow demonstrates high speed with good detection accuracy and small false alarm rate (10% of actual hotspots).

Key novelties of our proposed flow includes,

- Novel *critical feature* extractions are proposed for MLK training noise reduction, detection speed-up and accuracy enhancement.
- Supervised training / learning procedures are employed for MLK establishment and special algorithm is proposed for further accuracy improvement.
- Fast kernel re-training ensures scalability to new technology process, design rules and Optical Proximity Correction / Double Patterning set-ups.

The rest of the paper is organized as follows. Section 2 introduces preliminaries and describes each step of our proposed detection flow in detail. Testing benchmarks and simulation results are shown and discussed in Section 3, followed by conclusion in Section 4.

2. PROPOSED FLOW AND ALGORITHMS

2.1 A Motivational Example

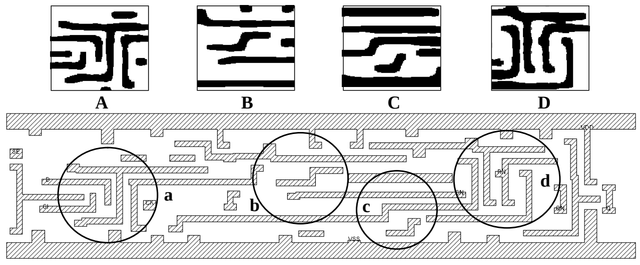
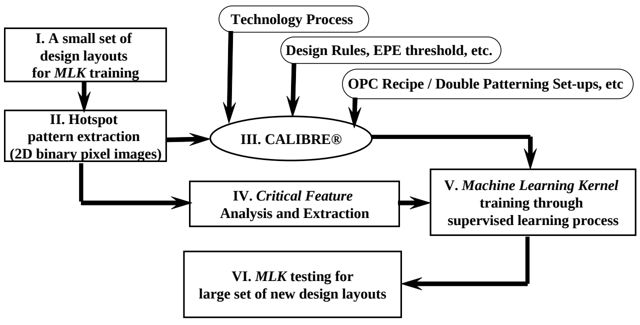


Figure 1: a,b,c and d are 4 local samples from a certain 45nm design [2]; A,B,C and D are the corresponding litho-simulated print-images of a,b,c and d, after OPC.

The rationale of our proposed hotspot detection flow can be illustrated from Fig. 1, where a, b, c, and d are metal1 layout patterns for a certain 45nm logic cell design; correspondingly, A, B, C, and D are the litho-simulated print-image of a, b, c, and d after OPC, representing post fabrication layouts of the 4 local patterns in the original design. From A, B, C and D, CALIBRE [1] shows that a and d are highly susceptible design patterns to the fabrication process, pattern c is slightly better and pattern b contributes the least towards generating hotspots in post fabrication stage. With A-D, we can re-design a, c and d area in the original layout to avoid post fabrication circuit defects, such as shorts, opens or other issues that high variability brings. However, in order to get A-D, there are complex integrals and convolutions involved in lithographic simulation, which is very expensive in terms of both run-time and computational resources, especially when invoked repeatedly for DFM closure.

In this paper, a machine learning kernel is established through mining the correlations between a small set (a few hundred) of design layouts such as a to d and their corresponding lithographic simulated images as in A to D. Afterwards, the kernel is used to predict post fabrication hotspot patterns without invoking lithographic simulations, leading to very fast detection speed. Unlike some previous work [11, 13] which focus on pixel based layouts, we propose a set of *critical features* as a compact representation of the original images. Algorithms for extracting these features are fast and hotspot capture rate is proven high (plus 90% on average) by a large set of simulation benchmarks.

2.2 Machine Learning based Detection Flow



I. A small set of design layouts for *MLK* training
II. Hotspot pattern extraction (2D binary pixel images)
Technology Process
Design Rules, EPE threshold, etc.
OPC Recipe / Double Patterning Set-ups, etc
III. CALIBRE®
IV. *Critical Feature* Analysis and Extraction
V. *Machine Learning Kernel* training through supervised learning process
VI. *MLK* testing for large set of new design layouts

Figure 2: **Overall proposed flow with stages *I-VI***

The overall flow is shown in Fig. 2, consisting of 6 stages. In *Stage I*, a small set Ω of design layouts are established as initial raw data set of *MLK* training, followed by *Stage II*, where a collection set Δ of typical binary pattern images are sampled from the design layout set Ω. In *Stage III*, CALIBRE simulation set-ups (such as technology process, design rules, OPC, EPE, etc) are loaded and lithographic simulations are performed on set Δ at a one-time cost, identified post-OPC hotspot patterns are stored in set Θ according to EPE threshold. *Stage IV* performs *critical feature* analysis and generates the critical metrics vector, as training and classification input vector to the *MLK*, which is an essential step for low noise data-training/classification and high hotspot detection accuracy. *Stage V* imports the critical metrics vector from *Stage IV* and performs supervised *MLK* training and validation based on set Θ from *Stage III*, resulting in an optimized highly non-linear *MLK*. The established *MLK* is tested on large sets of new design layouts in *Stage VI* with the same setups as in *Stage III*.

2.3 Critical Feature Analysis and Extraction

In this paper, *critical hotspot feature* is a metric extracted from the original design layout pattern as a representation mapping of a set of parameters most critical and sensitive to the presence of hotspot patterns. An effective *critical feature* should remain the same under arbitrary 2D transformations such as shifting, rotation and mirroring, etc. A generalized set of *critical feature* includes all pixel based image analysis transforms and representations, such as discrete fourier transform, Hough transform, distance transform and representations of particular patterns of interest.

In this paper, we proposed 3 major features, namely *Bounded Rectangle*, *T-shape metal* and *L-shape metal* features. As a compact representation of the original pixel image pattern, these features capture the relative geometry relations in between metal tracks effectively and lead to satisfactory detection accuracy. Unlike other 2D pixel based transforms, the proposed features are computationally easy to extract, thus contribute to a fast detection flow. Although more features can be added to enrich the existing critical feature metric, they can also slow down the detection process. Thorough simulations demonstrate that the proposed 3 critical features are generic enough to cover up to over 90% of hotspots with small false alarms (less than 10%).

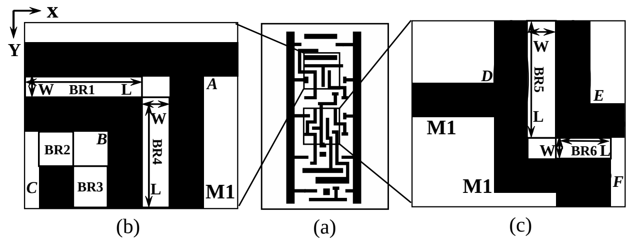


Figure 3: **(a) certain 45nm cell layout; (b)(c) Two sampled pattern examples for critical-feature extraction procedure.**

2.3.1 Bounded Rectangle Feature

The first *critical feature* we propose is the *Bounded Rectangle* feature, as illustrated by the rectangles in between of metal wires in Fig. 3(b)(c), *BR1* to *BR6*. Fig. 3(a) is a certain design layout, with two sampled patterns (b)(c), from which critical features are to be extracted. *BR* feature records the relative geometrical positioning between adjacent metal tracks through metal interval representation. Each *BR* is expressed with a 5 parameter vector *(W, L, X, Y, D)*, where L denotes the length of *BR* along the metal edges confining itself; W denotes the width of *BR* along the direction perpendicular to L; *(X, Y)* is the coordinates of the upper-left corner of *BR*; D is set to 0 if W is along X direction, to 1 if W is along Y direction. For example in Fig. 3(b), *BR1.D = BR2.D = 1; BR3.D = BR4.D = 0.*, in Fig. 3(c), *BR5.D = 0; BR6.D = 1.*

2.3.2 T-shape and L-shape Features

The other two *critical features* we propose are the *T-shape* metal feature and *L-shape* metal feature, as illustrated in Fig. 3(b)(c), region A to F. *T-shape* and *L-shape* features signify the number of *T-shape* and *L-shape* metal wires extending along both sides of *BRs*, together with corresponding wire width and jog width information. For example in Fig. 3(b), area A is *T-shaped* metal for *BR1/BR4*, area B is *L-shaped* metal for *BR1/BR2/BR3/BR4*, area C is neither *T-shape* nor *L-shape* for *BR2/BR3*. In Fig. 3(c), area D is *T-shape* metal track for *BR5*, area E and F are *L-shape* metal tracks for the right side of *BR5* and both sides of *BR6*.

Combining these features, we obtain the critical information of both adjacent metal tracks and the intervals in-between them for a certain pixel based layout. These features proposed are proven to have high detection efficiency meanwhile maintaining a low noise figure compared with [13]. Through executing proposed *critical feature* extraction, a feature metric vector is derived for each bounded metal track region in every image from set Δ, in the form of *(W, L, X, Y, D, Tf, Lf)*. A sorted collection of these vectors forms a metric for each layout image sample, such metric is the final input for MLK supervised training in the following *Stage V*. The purpose of the training is to iteratively establish a knowledge kernel, which can be applied later to predict hotspot patterns for new design layouts.

2.4 MLK Supervised Training

As an important classification technique in data mining, Artificial Neural Network (ANN) originated from imitating human brain neuron networks and human learning activities [15]. A small size ANN with 5 to 10 hidden layer neurons can be trained to establish highly nonlinear models with the highly interconnected manner of neurons in the network.

In our flow, we take the extracted critical features as inputs and feed into the ANN network for supervised training, which is essentially parameter optimizations for all the neurons in the network, through iterative coefficient updating. After the training procedure, in Section 3 the established ANN kernel will be directly applied onto various benchmarks in 45nm technology for hotsot detection tasks.

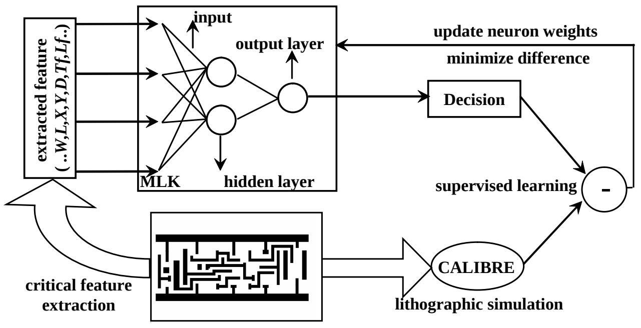


Figure 4: An illustration of the MLK (Machine Learning Kernel) supervised training

As shown in Fig. 4, supervised training for ANN results in a weight vector on all the links of the neuron network (circles in the MLK box) to minimize the MSE (mean square error) between network prediction results (binary decision) and the given supervising target (CALIBRE simulation result for the same layout from which critical features are extracted). As an iterative process, it employs various kernel functions for weight update towards MSE minimization.

The objective of the iterative training process can be written as follow,

$$Minimize \qquad \frac{1}{M}\sum_{p=1}^{M}(A^p_{output} - T^p_{calibre})^2$$

where M is the total number of samples in the supervised training set, A_p is the ANN prediction output for pth sample pattern at certain iteration and T_p is the training target of the sample, which is pre-set by lithographic simulations. For each iteration, all training samples are employed and prediction is made based on the most updated set of weights. Minimization stoping criteria is the satisfaction of certain pre-defined training error target. For effective weight update, various network architectures and update functions have been proposed, a general form of the update is as follows,

$$x_{k+1} = x_k - \alpha_k g_k$$

where x_k is the weight vector for the network links and x_{k+1} is the updated vector after 1 iteration and α_k is the learning rate vector for x_k. g_k is the update function which we can choose from a wide range of packages in MATLAB, such as resilient backpropagation and conjugal gradient, etc.

2.5 Detection Enhancement for Large Layout

For large area design layouts, we propose an algorithm to further increase detection accuracy, which we refer to as proximity detection. *detection_effort* in Algorithm 1 is defined as follows,

Algorithm 1 PDA (Proximity Detection Algorithm)

```
1: //Generate locations for Sampled-Pattern-for-Testings
2: Generate SPT seed (detection_effort, distribution)
3: Populate each seed locally (proximity_effort)
4: //Invoke ANN kernel with multi-threaded technique
5: for each SPT($i$) do
6:     MLK(SPT($i$).feature)
7: end for
8: hotspot voting process within each local proximity
```

$$e = \frac{\sum^{i}_{effective\_area} SPT(i)}{whole\_design\_area}$$

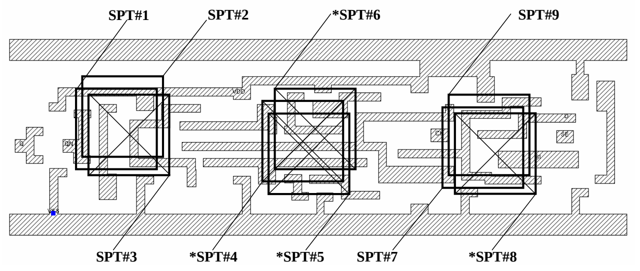


Figure 5: An example for cell level hotspot detection using Algorithm 1. (e set to < 0.1 for observation convenience)

where e is the ratio of total effective areas covered by SPTs (sampled patterns for testings) in Algorithm 1 versus original design layout area. Fig. 5 shows an illustration of Algorithm 1, where we examine a 45nm cell design layout. With Algorithm 1, false detection on $SPT3$ and $SPT8$ are detected and discarded through a voting process among their respective proximity SPTs before final hotspot decisions are made for the particular region. Advantages of combining our proposed flow with Algorithm 1 is that it further improves detection accuracy using proximate pattern correlations. Since our ANN (artificial neural network) kernel is already established, extra prediction time caused by the SPT area overhead is very little, especially with multithread techniques.

3. SIMULATION AND TESTING

Benchmarks are taken and built based on NANGATE 45nm cell library [2]. Testing and simulation are performed on Intel Xeon 2.4GHz Linux with 4G RAM.

70 CALIBRE identified hotspot patterns and 64 non-hotspot patterns participated in the ANN supervised training, with window size less than 1.5 micron by 1.5 micron. ANN hidden layer neuron is set to 5, learning rate to 0.05, epochs to 30, training error target to 0.01, kernel function is finalized to Resilient Backward Propagation with sigmoid function. Entire training process took less than 200 seconds to finish.

Upon completion of the supervised training, we apply the ANN kernel to following benchmarks for cross-validation and testing:

- B1: Small Area Patterns. B1 consists of 35 small area design patterns from [2], which the ANN kernel has never seen during training and validation. As a comparison baseline, CALIBRE litho-simulation was carried out and 17 hotspot patterns are identified, together with 18 non-hotspot patterns.

Table 1: Performance of the proposed *critical-feature* based hotspot detection with *Machine Learning Kernel.*

Benchmarks	Testing target	HotSpot Detection Rate	Non-HotSpot Detection Rate	HotSpot Detection -False-positive Rate	HotSpot Detection -False-negative Rate
B1 [a]	Best Perf	100%	98%	2%	0%
(MLK kernel;	Average Perf	94%	85%	15%	6%
without PDA[b])	Worst Perf	80%	54%	56%	20%
	XNOR2_X2	100%	100%	0%	0%
B2 [a]	DFF_X2	100%	100%	0%	0%
(using PDA[b])	SDFFRS_X2	100%	100%	0%	0%
	SDFF_X2	100%	100%	0%	0%
B3 [a](with PDA[b]):	Accuracy Rate	90%	—	10% false alarms	10%

[a] Hotspot detection EPE threshold is set to 15nm in 45nm technology process.
[b] PDA: Proximity Detection Algorithm

- B2: 45nm Cell Level Layout Patterns. B2 consists of 5 large area cell level design layouts from [2], which did not participate in the supervised training step of the ANN kernel.
- B3: Chip Level Design Layout. B3 is taken from a fully placed and routed 45nm FFT kernel, with 644 modules and 48027 cell instances placed on 380 by 350 micron die.

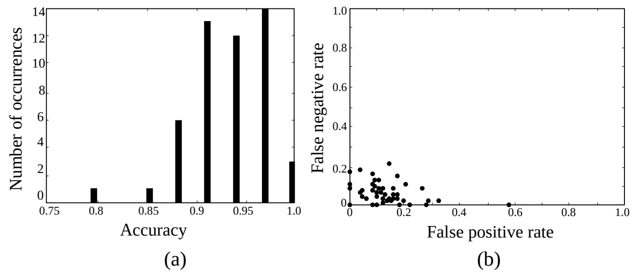


Figure 6: (a) Histogram of B1 hotspot detection accuracies with 50 independent experiments; (b) *False positive rate* versus *false negative rate*

From Fig. 6(a)(b), we fully evaluate the established ANN kernel with 50 independent cross-validations on B1. As a result, our flow demonstrates above 90% of accuracy with over 85% confidence, with the best detection accuracy 100% and worst 80% (only one occurrence every 50 testings). In Fig. 6(b), we plotted another two crucial parameters for performance evaluation, namely *false positive rate* and *false negative rate*. *False positive rate* is the percentage of actual non-hotspots detected as hotspots (hotspot false alarms); *False negative rate* refers to the percentage of actual hotspots detected as non-hotspots, which is to be kept minimal.

For B1, our average *false positive rate* is 15%, and *false negative rate* is 6%, as shown in Fig. 6(b). Detailed summary is in Table 1. Since B2/B3 are large area design layouts, we employ the proposed proximity detection algorithm for accuracy enhancement. e is set to <1.0 for B2 and between 1 to 3 for B3. e larger than 3 is considered as high effort, meaning the total SPTs area in Algorithm 1 is more than 3 times of original layout area. e should be set higher for denser design layouts.

As a result, hotspot detection accuracy for B2 reached 100%, with *false positive rate* 0%. B3 is evaluated with the same manner and results show plus 90% of hotspot detection accuracy with very small (10% of total hotspots) *false positive rate* (detection false alarms). Details can be found in Table 1.

4. CONCLUSION

As crucial assistance for lithographic simulation, fast and accurate non-lithographic detection flows start playing more and more important roles in modern DFM, where lithographic simulations are costly to run. In this paper, we presented a *critical feature* extraction/classification based hotspot detection flow utilizing modern machine learning (artificial neural network) technique. With *critical feature* representation, learning/detection noise for the training procedure is effectively reduced without run-time overhead when compared with [13]. Experimental results demonstrate small detection false alarm rate (10% of actual hotspots) and an average of plus 90% detection accuracy, with best achieved accuracy 100%.

5. REFERENCES

[1] CALIBRE is a Mentor Graphics lithography simulation tool.
[2] http://www.nangate.com, nangate 45nm cell library.
[3] http://www.tela-inc.com,1d restrict design rule.
[4] International technology roadmap for semiconductors. 2007.
[5] M. Bohr. Silicon technology leadership and the new scaling paradigm. In *INTEL Developer Forum*, April 2007.
[6] T.-C. Chen et al. Predictive formulae for opc with applications to lithography-friendly routing. In *Proc. Design Automation Conf.*, June 2008.
[7] M. Cho, K. Yuan, Y. Ban, and D. Z. Pan. ELIAD: Efficient lithography aware detailed router with compact printability prediction. In *Proc. Design Automation Conf.*, June 2008.
[8] J. Ghosh. Adaptive and neural methods for image segmentation. In A. C. Bovik, editor, *Handbook of Image and Video Processing*, pages 401–414. Academic Press, 2000.
[9] A. B. Khang et al. Fast dual graph based hotspot detection. In *Proc. of SPIE*, volume 6349, 2006.
[10] J. Kim et al. Hotspot detection on post-opc layout using full chip simulation based berification tool: A case study with aerial image simulation. In *Proc. of SPIE*, volume 5256, 2003.
[11] N. Ma et al. Automatic hotspot classification using pattern-based clustering. In *Proc. of SPIE*, volume 6925, 2007.
[12] J. Mitra, P. Yu, and D. Z. Pan. RADAR: RET-aware detailed routing using fast lithography simulation. In *Proc. Design Automation Conf.*, June 2005.
[13] N. Nagase et al. Study of hotspot detection using neural network judgement. In *Proc. of SPIE*, volume 6607, 2007.
[14] E. Roseboon et al. Automated full-chip hotspot detection and removal flow for interconnect layers of cell-based designs. In *Proc. of SPIE*, volume 6521, 2007.
[15] P.-N. Tan, M. Steinbach, and V. Kumar. Classification: Alternative techniques. In *Introduction to Data Mining*, pages 207–315. Addison-Wesley, 2006.
[16] TSMC. Design for manufacturability. In *TSMC Technology Symposium, Austin Texas*, May 2008.
[17] J. Xu et al. Accurate detection for process hotspots with vias and incomplete specification. In *Proc. Int. Conf. on Computer Aided Design*, 2007.
[18] H. Yao et al. Efficient process hotspot detection using rang pattern matching. In *Proc. Int. Conf. on Computer Aided Design*, 2006.
[19] P. Yu and D. Z. Pan. TIP-OPC: A new topological invariant paradigm for pixed based optical proximity correction. In *Proc. Int. Conf. on Computer Aided Design*, 2007.