High Performance Lithographic Hotspot Detection using Hierarchically Refined Machine Learning

Duo Ding, Andres J. Torres*, Fedor G. Pikus* and David Z. Pan
ECE Dept. Univ. of Texas at Austin, Austin, TX 78712
*Mentor Graphics Corporation, 8005 S.W. Boeckman Road, Wilsonville, OR 97070
ding@cerc.utexas.edu, torres_andres@mentor.org, fedor_pikus@mentor.com, dpan@cerc.utexas.edu

ABSTRACT

Under real and continuously improving manufacturing conditions, lithography hotspot detection faces several key challenges. First, real hotspots become less but harder to fix at post-layout stages; second, false alarm rate must be kept low to avoid excessive and expensive post-processing hotspot removal; third, full chip physical verification and optimization require fast turn-around time. To address these issues, we propose a high performance lithographic hotspot detection flow with ultra-fast speed and high fidelity. It consists of a novel set of hotspot signature definitions and a hierarchically refined detection flow with powerful machine learning kernels, ANN (artificial neural network) and SVM (support vector machine). We have implemented our algorithm with industry-strength engine under real manufacturing conditions in 45*nm* process, and showed that it significantly outperforms previous state-of-the-art algorithms in hotspot detection false alarm rate (2.4X to 2300X reduction) and simulation run-time (5X to 237X reduction), meanwhile archiving similar or slightly better hotspot detection accuracies. Such high performance lithographic hotspot detection under real manufacturing conditions is especially suitable for guiding lithography friendly physical design.

1. INTRODUCTION

With continuous shrinking of semiconductor process technology nodes, the minimum feature size of modern IC is much smaller than the lithographic wavelength [1]. In order to bridge the wide gap between design demands and manufacturing limitations of the current mainstream 193nm lithography, various DFM techniques [2–5] have been proposed to improve product yield and avoid potentially problematic patterns (i.e., process hotspots). However, for 45*nm* node and below, hotspot patterns still exist even after design rule checking (DRC) and various resolution enhancement techniques (RET) such as optical proximity correction (OPC), sub-resolution assist feature insertions/layout re-targeting, etc.

Therefore, fast and high fidelity hotspot detection engines can play an essential role to enhance physical verification/DRC (Design Rule Checking), and to develop process aware physical design tools. Conventional approaches that employ lithographic simulations [6, 7] are accurate but very costly to run; on the other hand, approaches that utilize pattern/graph matching techniques [8–10] are fast but they rely on a set of pre-defined hotspot patterns. However, general hotspot patterns are hard to define/model in a deterministic manner - too many patterns lead to high overestimate rate and too few patterns result in low hotspot coverage; pattern enumeration becomes even more problematic as technology processes become more mature and resolution enhancement techniques improve, thus the definition of hotspots are highly dependent on the continuously improving manufacturing conditions.

In recent years, there are emerging works that start incorporating modern data mining methods for fast and accurate hotspot detection. A neural network judgment based detection flow was proposed in [11], where 2D hotspot image patterns were directly used to train an artificial neural network (ANN) kernel. In [12], data mining algorithms are developed for hotspot pattern (2D images) clustering. While these early attempts have shown promising potential for data mining based hotspot detection, there are still limitations yet to overcome, such as high training noise and low hotspot detection fidelity.

Later in [13], a support vector machine (SVM) based hotspot detection method is utilized through performing 2D distance transform and histogram extraction on pixel based layout images. Similarly in [14], SVM is employed for hotspot detection through extraction and classification of certain special layout density related metrics. As an improvement over [11, 12], [13, 14] demonstrate higher detection accuracy and lower classification noise, due to the high fidelity metrics introduced. However, potential issues with such approaches lie in run time and detection coverage, since 2D transforms and density extractions can be expensive to perform, meanwhile detection windows for the layout images are hard to anchor for full chip level detections. In practice, these windows are slid or scanned across the entire layout with certain amount of overlap between each other. As an inevitable result, detection performance becomes a trade-off between run-time and sliding window coverage. In [15], *critical hotspot signature* is proposed and extracted through certain special edge-based metrics. Although such edge-based extractions operate much faster compared with [13, 14], its chip level application still faces similar issues such as scanning window coverage, etc.

Moreover, very few studies exist that deal with the detection challenges under the real manufacturing conditions which put stricter requirements to the detection engine to detect a small number of real hotspots with low false alarm rate. In order to be practically employed in modern IC physical design, a successful hotspot detection engine must demonstrate superior speed compared to full lithography simulation ($>$ 100 CPUs running in the order of days) and DRC (tens of CPUs running for a few hours), as well as comparable performance to meet the real design and manufacturing requirements. Unfortunately under such situations, the hotspot evaluation models in [11–15] suffer from severe performance degradation, since real hotspot detection under real manufacturing conditions requires more than a straightforward model, but rather, multi-level detection models with hierarchies of performance refinement.

To better address the issues and challenges above, we propose for the first time a hierarchically refined machine learn-

This work is supported in part by NSF, SRC, Fujitsu, IBM and Intel.
Asia and South Pacific Design Automation Conference (ASPDAC) 2011, Jan. 25–28, 2011, Pacifico Yokohama, Yokohama, Japan.

978-1-4244-7516-2/11/$26.00 ©2011 IEEE

ing framework for high performance hotspot detection flow, providing full layout, feature-centric analysis without being penalized in run-time or coverage by sliding window or raster scanning related techniques. Under such a hierarchically refined data mining framework, we define novel classification feature metrics in a fragment-based manner and implement the framework in a leading industrial geometry processing engine via a shared object library [16]. The proposed flow is implemented with enhanced ANN and SVM kernels on large industry layouts under real manufacturing conditions, demonstrating very promising performance in detection accuracy enhancement, false-alarm suppression and CPU time reduction.

2. OUR CONTRIBUTIONS

In face of the aforementioned challenges, this paper proposes a novel flow for high performance hotspot detection under real manufacturing conditions. Our main contributions are summarized as follows:

- We define novel hotspot feature-centric characterization and powerful machine learning kernels for high fidelity detection under real manufacturing conditions.
- We introduce a hierarchically refined machine learning based flow for ultra-low false-alarm hotspot detection.
- We propose special hotspot signature measurements for ultra-fast, full layout detection without sliding window or raster scanning techniques.
- We perform thorough qualification using real industry examples for a 45nm metal1 process under real manufacturing conditions.

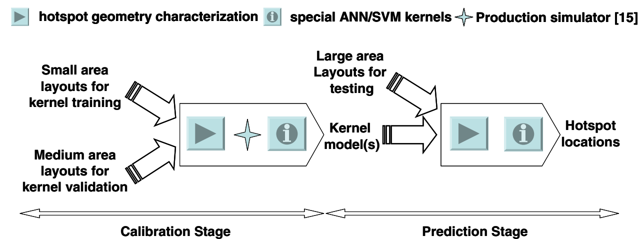

Figure 1: A top level overview of the proposed flow

Before going to further details, we illustrate a simplified top level overview of the proposed flow in Fig. 1, where *calibration stage* involves training and validations of multiple kernel models, which are then employed in the *prediction stage* for fast searching of potential defective pattern locations in new design layouts. In the *calibration stage*, input layouts first go through a novel hotspot geometry characterization/extraction step such that the sample locations are converted to compact 1D vectors; with these data vectors, special machine learning kernel models are trained and validated under the supervision of the production simulator [16]. In the *prediction stage*, new layouts are characterized/extracted as 1D vectors then applied to the established kernel models for high performance hotspot detection under a special integrative flow. The resulted models from *calibration stage* preserve the information of both the hotspot/nonhotspot features and the manufacturing conditions, therefore the *prediction stage* can be carried out onto large quantity of layouts with high efficiency once the *calibration stage* has been carried out *a priori*.

The rest of the paper is organized as follows, in Section 3, we introduce our proposed hotspot signature characterization for high detection coverage and speed, followed by descriptions of our special machine learning kernels for enhanced accuracy in Section 4. In Section 5 we describe an integrative flow with hierarchically refined classification techniques for ultra-low false-alarm hotspot detection under current real manufacturing conditions. Simulation results on various placed and routed industry layouts are assessed and analyzed in Section 6. Section 7 concludes the paper.

3. FEATURE-CENTRIC LAYOUT CHARACTERIZATION

Hotspot feature metrics (or hotspot signatures/models) refer to a set of special measurements which contribute strongly to successful decision making processes for hotspot detections. The procedure to extract feature metrics from a design layout is referred to as layout context characterization. Unlike special restricted design rules, layout measurements and characterizations do not decide whether a certain pattern is defective or not, but leave the decision making process to recursively trained and validated kernels (engines) using machine learning techniques. In face of the aforementioned challenges under real manufacturing conditions, a properly defined set of feature metrics plays the **key** role for a successful classification flow. Unlike previous studies utilizing 2D transforms or density based calculations or sliding window based techniques, we propose novel metrics and special data structures for layout characterization with significant run-time reduction and satisfactory accuracy.

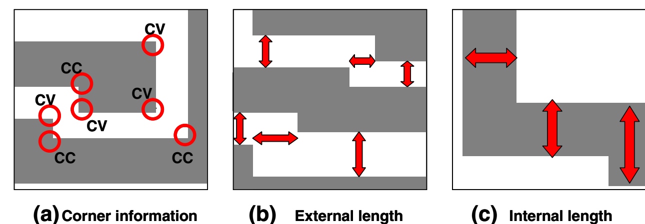

Figure 2: 3 major types of hotspot feature measurements

3.1 Hotspot Signature Measurement

Before introducing our special hotspot signature metrics, we first define several types of fast layout measurement data structures and several fundamental operators, which scale to cover the entire layout without applying sliding window or density based techniques. Different types of feature measurements are illustrated in Fig. 2, namely (a) corner information (convex or concave), (b) distance to an externally facing polygon edge and (c) distance to an internally facing polygon edge. In a fragmented design layout where the fragments (polygon edges) are indexed numerically, these feature measurements are optimized to be processed and stored with high efficiency in [16] throughout the entire layout. With a proper combination of these proposed measurements, we can accurately characterize the entire layout at a one-time cost, therefore the context representation of any fragment in the layout becomes an indexing problem that can be processed with very high speed. Also due to such data structures employed, hotspot detection related geometries (such as jog shapes, corners, intra-distances and inter-distances, etc.) are generated for every fragment in the layout with full processing coverage without penalties introduced by traditional sliding window related methods.

To carry out the proposed feature measurements, we introduce 4 types of feature-centric operators constructed under

Table 1: Hotspot signature measurement operators

operators	operation description (features to measure)
$f_{corn}(\cdot)$	corner information: CV(convex) / CC(concave)
$f_{ext}(\cdot)$	external inter fragment distances
$f_{int}(\cdot)$	internal inter fragment distances
$f_{misc}(\cdot)$	miscellaneous information

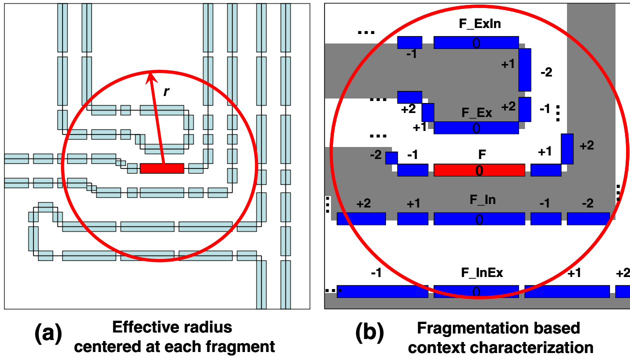

Figure 3: Fragmentation based hotspot signature extraction

the shared object environment of [16], as shown in Table 1. In particular, **Operator** $f_{misc}(\cdot)$ requests extra information regarding $Frag$, such as fragment orientation (X or Y axis) and the length of $Frag$, etc. Consequently, hotspot related geometries of a design layout are represented and indexed in high resolution (per-fragment based) with a combination of the defined operators, leading to a high fidelity quantization process, as we will describe in detail as follows.

3.2 Fragmentation based Context Characterization

To provide full characterization coverage over the entire layout, our hotspot feature definition and extraction are carried out on a per fragment basis. Given a properly fragmented layout and any fragment of interest F, we illustrate the concept of an effective radius r in our proposed context characterization procedure. As shown in Fig. 3(a)(b), r centers at (each) fragment F. By definition, effective radius r covers the neighboring fragments which need to be considered in the context characterization of F. In its empirical nature, r heavily depends on the lithography processes, and it is generally easy to pick in the training and validation procedure given the manufacturing conditions.

In Fig. 3(b), we illustrate our proposed context characterization process: suppose the current fragment of interest is F (colored red in the center of the effective circle), **first** we define several short-hand notations whose indices are used for indexing throughout the fragments lying within the effective region of F. Details please refer to Table 2. Note F can also be denoted as F_0. **Then** we present the characterized feature vector of fragment F in the following formulation as in Eqn.(1)-(2):

$$V_F = \sum_{i}^{\widetilde{F}_i \in \delta_r^F} \{f_{corn}(\widetilde{F}_i) \oplus f_{ext}(\widetilde{F}_i) \oplus f_{int}(\widetilde{F}_i) \oplus f_{misc}(\widetilde{F}_i)\} \quad (1)$$

$$\widetilde{F} = [F, F_Ex, F_In, F_InEx, F_ExIn...] \quad (2)$$

where F is an integer ID number representing certain fragment in the layout, δ_r^F is the effective region of F. Operators $\bigoplus$ and $\sum$ are matrix operations for the generation of a vector V_F. The length of vector V_F is the total number of features M.

In this paper, the parameter vector V_F formed by the context characterization is defined as the **hotspot signature metric**, and the context characterization for each F is also referred as *feature extraction* process. Such a process filters out detection noise and provides a compact vector based data set for MLK (machine learning kernels) to be properly established. Meanwhile, inside [16] environment, Eqn.(1) can be bulk processed with a one-time calculation for all fragments throughout the entire layout by directly invoking the afore-constructed operators, resulting in upto hundreds of times of run-time reduction compared with previous studies. Simulation results and further discussions will be presented in Section 6.

Table 2: short-hand fragment notations

notation	descriptions of the short-hand notation
F	current fragment of interest (detection anchor point)
F_In	the fragment(s) facing F internally
F_Ex	the fragment(s) facing F externally
F_In_Ex	the fragment(s) facing F_In externally
F_{+i}	the ith neighbor traced from F counter-clockwise
F_{-i}	the ith neighbor traced from F clockwise

4. ROBUST LEARNING KERNELS

For our proposed hotspot detection methodology, MLKs (machine learning kernels) play an essential role unlike previous works. In particular, we employ special MLKs and integrate them in a special manner for detection accuracy/robustness enhancement and false-alarm suppression. In this paper, we modify and enhance 2 most common types of machine learning techniques: ANN (artificial neural network) and SVM (support vector machine) in mainly 2 aspects: first, robustness and accuracy in the weight update process; second, detection threshold $\mathcal{F}(\cdot)$ optimizations for simultaneous $Hhit$ improvement and $Hextra$ suppression.

Generally speaking, typical ANN and SVM perform similarly for binary classifications. Although SVM guarantees global optimum in its formulation if the kernel satisfies Mercer's condition, it is usually prone to data noise and may also result in longer run-time for high dimensional data sets when the number of support vectors becomes large; ANN on the other hand, provides more noise robustness, compact kernel models (neuron weight) and flexible network structures.

With these considerations, we incorporate both classes of kernels into our detection engine with special modifications. For ANN kernels, we modify the resilient backpropagation update method [17] with enhanced robustness and a better parameter trade-off between convergence speed and detection accuracy; we also investigate and propose strategies for detection threshold optimizations. For SVM kernels, we combine a C-type SVM formulation with higher accuracy working set selection based on [18], together with hotspot detection threshold optimizations. We describe our special kernel formulations and implementations briefly as follows, with related symbols and variables summarized in Table 3.

4.1 ANN: Artificial Neural Network Kernel

A typical ANN classifies data by predicting a value for each V_p based on an established set of weights and biases assigned to certain neural network structure. Our ANN kernels are customized with single hidden layer of neurons, with transfer functions denoted as f_{hid}. Inputs V_p to the ANN

Table 3: ANN/SVM kernel related variables

Variables	descriptions
N	total number of input sample vectors
M	feature number per sample vector
V_p	input sample vectors, p=1 to N
V_p^i	the ith element(feature) of V_p, i=1 to M
y_p	hotspot label for V_p in *calibration*, p=1 to N
f_{in}	input transfer function for ANN kernel
f_{hid}	hidden layer transfer functions for ANN kernel
f_{out}	output layer transfer function for ANN kernel
out_p	ANN output prediction value from V_p input
out_{hid}^j	ANN hidden layer jth neuron prediction output
$\overrightarrow{\omega}$	ANN kernel matrix of neuron connection weight
$K(V_i, V_j)$	SVM kernel function between V_i and V_j
α	SVM weight vector for input V_p's
$\mathcal{F}(\cdot)$	threshold function for hotspot decision making
$Est_{\tilde{p}}$	MLK estimation result for a new sample $V_{\tilde{p}}$

kernels are the extracted feature vector samples labeled with values (y_p) indicating hotspot or nonhotspot patterns (these values can be continuous for variability prediction). We use p to represent feature vector index with $p = 1$ to N, V_p^i denotes the ith element of vector V_p, $i = 1$ to M, where M is the total number of features for each sample vector. We use f_{in} and f_{out} to represent input and output layer transfer functions, and index i, j, k to indicate neuron indices in the input, hidden and output layer respectively. In particular, we choose a linear output function, *sigmoid* hidden layer functions and formulate our ANN kernel calibration process in Eqn.(3) to Eqn.(10) as follows:

$$objective : minimize\{\sum_{p=1}^{N} E^p\} \quad w.r.t \quad \omega_{ij}, \omega_{jk} \tag{3}$$

$$E^p = \frac{1}{2}[out_p - y_p]^2 \tag{4}$$

$$out_p = f_{out}\{\sum_j \omega_{jk} \cdot f_{hid}(\sum_i V_p^i \cdot \omega_{ij})\} \tag{5}$$

$$\frac{\partial E^p}{\partial \omega_{jk}} = (out_p - y_p) \cdot f_{hid}\{\sum_i V_p^i \cdot \omega_{ij}\} \tag{6}$$

$$\frac{\partial E^p}{\partial \omega_{ij}} = (out_p - y_p) \cdot \omega_{jk} \cdot V_p^i \cdot (1 + out_{hid}^j)(1 - out_{hid}^j) \tag{7}$$

$$f_{hid} = \frac{2}{(1 + e^{-2x})} - 1, \qquad f_{in} = f_{out} = x \tag{8}$$

$$sign_func(x) = \begin{cases} -1 & \text{x} < 0 \\ 0 & \text{x} = 0 \\ +1 & \text{x} > 0 \end{cases} \tag{9}$$

$$Est_{\tilde{p}} = \digamma\{f_{out}[\sum_j \omega_{jk} \cdot f_{hid}(\sum_i V_{\tilde{p}}^i \cdot \omega_{ij})]\} \tag{10}$$

4.2 SVM: Support Vector Machine Kernel

SVM classifies sample vectors by calculating a (hyperplane) boundary with maximum separation margin in-between of different classes. With such an optimized margin, only the sample vectors forming the boundaries are considered as contributing factors for new sample classifications. These vectors are called support vectors, they are assigned different weights and they perform classification tasks through certain kernel function $K(V_i, V_j)$. For this paper's high fidelity detection flow, we combine a typical 2-class soft-error tolerant SVM kernel, a special working set selection technique using second order information [18] and a detection threshold $\digamma$ optimization procedure in the SVM kernel *calibration process* towards simultaneous accuracy enhancement and false-alarm suppression.

The dual problem of our quadratic formulation of C-type SVM is given as follows in Eqn.(11) to Eqn.(16):

$$objective : minimize\{f(\alpha) = \frac{1}{2}\alpha^T Q\alpha - e^T\alpha\} \quad w.r.t \quad \alpha \tag{11}$$

$$subject \quad to : 0 \le \alpha_i \le C, i = 1, ..., N, \tag{12}$$

$$y^T \cdot \alpha = 0 \tag{13}$$

$$K(V_i, V_j) = exp\{\gamma \cdot \|V_i - V_j\|^2\} \tag{14}$$

$$slope_func(x) = \begin{cases} 0 & \text{x} \le 0 \\ x & 0 < \text{x} < \text{C} \\ C & \text{x} \ge \text{C} \end{cases} \tag{15}$$

$$Est_{\tilde{p}} = \digamma\{\sum_i \alpha_i y_i K(V_{\tilde{p}}, V_i) + bias\} \tag{16}$$

5. HIERARCHICAL MACHINE LEARNING FOR LOW FALSE-ALARM

5.1 Overview

Due to the small number of real hotspots and the highly noisy detection environment under real manufacturing conditions, we hereby propose a hierarchically refined machine learning method to integrate our proposed feature extraction processes and multi-level machine learning kernels. In its nature, such an approach hybrids the strength of ANN (or SVM) kernels and hierarchical classification methods thus contributes to significant detection performance enhancement in terms of run-time, detection accuracy and false alarms, when compared with previous approaches using straight-forward machine learning techniques.

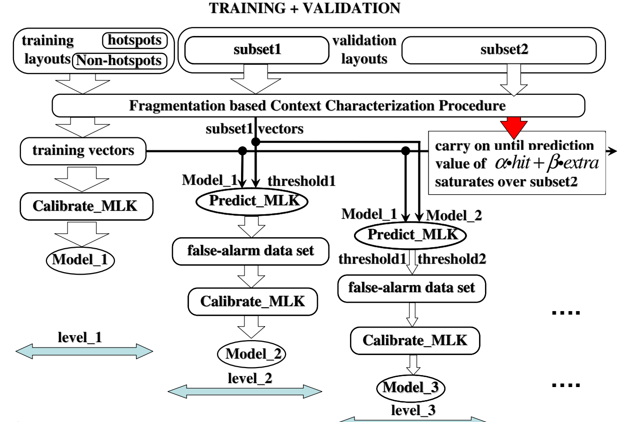

Figure 4: Hierarchically refined training/validation with multiple hotspot metric models and thresholds

As shown in Fig. 4, our proposed training and validation processes take place in a multi-level manner with each level contributing an unique classification model. Different levels of models established in the *calibration stage* are then employed in the *prediction stage* in a similar hierarchically refined manner to help reduce the false alarm rate $Hextra$ without penalizing the hotspot detection rate $Hhit$. Fig. 5 illustrates the hierarchical hotspot predication flow for the testing design layouts. In the following subsections, we describe our proposed hierarchical machine learning in detail.

5.2 Global Training and Detection

Here we refer to the first level training in Fig. 4 as the global training stage, since $Model_1$ is trained with the whole training dataset (on the global scale); similarly in Fig. 5, prediction with only $Model_1$ is defined as global detection, since the whole testing data go through this model. As we will show later in Section 6, under our proposed feature metrics and machine learning kernel implementations, global detection stage alone achieves very satisfactory hotspot detection accuracy, whereas the non-hotspot detection accuracy is not high enough. As we discussed, hotspot and non-hotspot patterns are highly unbalanced under current real manufacturing conditions, resulting in such a huge pool of non-hotspots that even a small fraction of false-alarms leads to excessive post-processing workload. Consequently, much stricter requirements are imposed on the non-hotspot detection accuracy than the hotspot detection accuracy. This is another motivation for our proposed hierarchical refinement.

5.3 Hierarchical Local Refinement

To further suppress false-alarms meanwhile maintaining satisfactory hotspot detection rate, we extend the global stage with extra sub-levels, which we refer as the *local hierarchical refinement*. As illustrated in Fig. 4 and Fig. 5, we employ hierarchical refinements in both *calibration* and *prediction* stages. In *calibration stage*, the refinement flow consists of several key steps: (1) training and validating multiple hotspot metric models using the entire training data set and the false-alarm data sets from a part of the validation data. (2) stopping criteria to decide when to stop adding more hierarchical models. (3) optimizations of the multiple thresholds $\digamma$ associated with the machine learning models. In the *prediction stage*, all the models and thresholds are

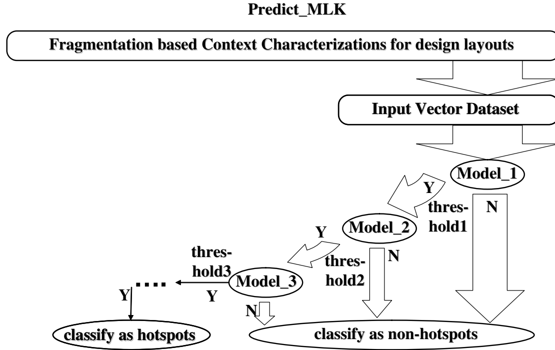

Figure 5: Hierarchically refined prediction with multiple hotspot metric models and thresholds

employed, and hotspots are detected as those patterns that are identified as 'positive' in all hierarchies of models. We describe related key steps in more detail as follows:

Training and validating multiple models: Using the $Model_1$ from the global stage, the *predict* procedure is carried out over the combination set of the training data and a subset of the validation data. A variable $threshold1$ is properly chosen such that a proper set of false-alarm patterns are derived. Note that the patterns within such a set are non-hotspots which are classified as hotspots by $Model_1$, and that under real manufacturing conditions the set is usually large. Subsequently, an extra model $Model_2$ is derived by invoking ANN (or SVM) training process over the false-alarm set. Similarly, such a routine continues by generating another false-alarm set with $Model_1/2$ under $threshold1/2$, leading to a third level model $Model_3$, so on and so forth.

Stopping criteria: To quantify the stopping criteria for the multi-level *calibration process*, we introduce a user defined performance metric Ψ_{perf} in Eqn.(17):

$$\Psi_{perf} = \alpha \cdot Hhit + \beta \cdot Nhit \tag{17}$$

where α and β are user defined weights, $Hhit$ is the hotspot detection accuracy and $Nhit$ is the non-hotspot detection accuracy. Therefore, Ψ_{perf} represents the weighed summation of hotspot and non-hotspot detection accuracies. At every level, Ψ_{perf} is evaluated over $subset2$ of the validation data and the multi-level routine stops when Ψ_{perf} saturates or starts getting worse.

Thresholds optimization : In this paper, we employ a heuristic approach for the threshold optimizations, that is to exhaust the solution space with grid based simulations and select the threshold combinations giving the best Ψ_{perf}. The main reason is two fold: first, *calibration process* is performed only once in an *a priori* manner therefore does not lead to run-time overhead for predictions over testing layouts; second, based on our experimentations in Section 6, Ψ_{perf} saturates or starts to deteriorate after level 3, thus the total solution space is fairly limited. More details of this step can be found in Section 6.

Prediction and testing : testing is carried out at speed with the calibrated models over a new set of layouts using the flow in Fig. 5, consisting of *feature extraction* and hierarchical machine learning classifications. Our proposed refinement detection flow differentiates the pattern streams and guides them through multiple levels of classification models, resulting in satisfactory detection accuracy and ultra-low false-alarms. For more details, please refer to Fig.4, Fig.5.

Table 4: Tested layouts details.

	dimension	hotspot count	non-hotspot count
C1	30 X 30 um^2	4	4.955k
C2	50 X 50 um^2	0	17.37k
C3	200 X 200 um^2	6	293.5k
C4	500 X 500 um^2	38	1779k
C5	1000 X 1000 um^2	137	7175k

Table 5: Simulation results of our proposed method on industry layouts under real manufacturing conditions.

	MLK - ANN				MLK - SVM			
	GD		GD+LR		GD		GD+LR	
	Hhit[a]	Nmis	Hhit	Nmis	Hhit	Nmis	Hhit	Nmis
C1	4	315	3	18	4	251	4	5
C2	-	109	-	11	-	81	-	3
C3	6	493	5	31	6	355	5	7
C4	32	3020	30	195	34	1983	31	38
C5	121	10960	111	485	122	7535	114	135

[a] Hhit is the number of correctly classified hotspots; Nmis is the number of incorrectly classified non-hotspots (false-alarm counts)

6. SIMULATIONS AND EXPERIMENTS

The simulation process involves 3 major steps: **first**, under real manufacturing conditions, ANN and SVM based classification models are trained on a 500 um^2 layout with fully placed and routed metal tracks in $45nm$ technology (in the global stage). **Second**, the validation (multi-level refinement) process (including the brute force threshold optimizations) is then carried out on metal1 layer of a medium area whole chip layout. **Last**, our proposed flow is tested and evaluated on $45nm$ industry layouts under a real set of $45nm$ process manufacturing conditions, in terms of $Hhit$, $Nhit$ and *prediction stage* run-time.

With the data in Table 4, we carry out the first two steps according to Section 5 and derive a three-level detection hierarchy: one global model and two local refinement models. With the models ready, we perform the prediction procedure as in Fig. 5 onto 5 industry layouts under real manufacturing conditions. Details of layouts C1 to C5 are listed in the following Table 4, from which we can further appreciate the challenges that we motivated in Section 2.

In Table 5 we list the simulation results of our proposed detection flow over C1-C5 layouts, where **GD** represents global detection alone and **GD+LR** represents the multi-level machine learning flow with both global detection and local refinement. We observe that although **GD** demonstrates satisfactory hotspot accuracy, it is **LR** that plays a vital role in bringing down the false-alarms. Also in these results, we see that under real manufacturing conditions, hotspots and nonhotspots are highly unbalanced and the detection accuracies for both classes should be kept maximum. Fig. 6 also provides a visualization of whole chip hotspot detection false-alarm counts on benchmark C5 by some currently existing works. From the figure we can see that our proposed method achieves the least amount of false-alarms without hurting the hotspot hit rate, therefore the workloads of the post-detection stages hotspot correction can be kept minimum. In view of the huge total pattern number of C5 from Table 4, we can understand that although a 95% of non-hotspot detection accuracy rate seems good, it has to be pushed to above 99.85% to tackle the real manufacturing conditions for the industry layouts to avoid excessive post-layout corrections.

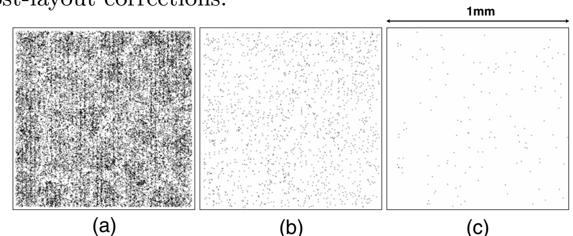

Figure 6: Visualizations of false alarm locations when simulating (a) [15] on C5, (b) [19] on C5, and (c) our method on C5 (barely visible:~100 false alarm spots on $1mm^2$ layout)

Table 6: Result comparison between previous hotspot detection methods and our method

	DAC09 [13]	SPIE09 [19]	ICICDT09 [15]	MLK - ANN GD	MLK - ANN GD+LR	MLK - SVM GD	MLK - SVM GD+LR
Avg hotspot detection accuracy $Hhit$	88%[a]	80%[b]	87%	88%	82%	89%	83%
Avg Nonhotspot detection accuracy	94.523%[a]	99.985%[b]	99.809%	99.847%	99.994%	99.895%	99.998%
Avg false alarm count per mm^2	300K[a]	1.1K[b]	13.5K	10K	0.45K	7.5K	0.13K
Avg CPU run-time[c] per mm^2	356[a]	30[b]	10	1.5	1.5	2.0	2.0
Avg real-time run-time[d] per mm^2	100[a]	8.50[b]	2.80	0.40	0.40	0.52	0.52

[a] Implemented within the same geometry engine framework [16] with slight modifications for compatibility reasons. Calibrated on a 100X100 um^2 region from layout C5 due to run-time constraint.
[b] Implemented based on [19] and tested on the same layouts.
[c] Run-time calibrated in the unit of CPU hour/mm^2 on Linux station with 2.8GHz quad-core processors.
[d] Run-time calibrated in the unit of real time hour/mm^2 on Linux station with 2.8GHz quad-core processors.

In Table 6, we list comparisons of accuracies and run-time between our approach and some existing studies. In the far right-hand columns of Table 6, our proposed methods demonstrate better performance with superior run-time under real manufacturing conditions. With similar or slightly better hotspot detection rate $Hhit$ of 82%-89%, we show hotspot false-alarm reductions ranging from 2.4X (between [19] and MLK-ANN-GD+LR) to 2300X (between [13] and MLK-SVM-GD+LR). Simulation run-time speed-ups range from 5X (between [15] and MLK-SVM) to 237X (between [13] and MLK-ANN), when calibred in CPU hour per layer per mm^2 unit. Within our proposed hierarchical detection flow, modified ANN kernels result in faster runtime than modified SVM kernels while SVM kernels outperform ANN kernels in both hotspot and non-hotspot detection accuracy. We also notice that within our flow, the run-time overhead introduced by local refinement steps are negligible in CPU hour, owing to our ultra-fast hotspot signature characterization operators and data structures.

Based on the experimentations in Table 6, we give a high level performance comparison among existing hotspot detection techniques in Table 7, where we further highlight the key contributions of our hotspot detection flow compared with existing works. It can be seen that our method is most suitable for lithography aware physical design due to its high accuracy, low false alarm, and very fast speed. While the machine learning based algorithms still miss some hotspots, this is acceptable at the physical design stage as a fast and high-fidelity guidance. If needed, those small number of missed patterns can be captured and complemented with pattern matching algorithms to achieve full hotspot detection coverage, without the enumeration issues that general pattern matching methods have.

7. CONCLUSION

To alleviate the huge run-time cost of current lithographic hotspot simulators, in this paper we proposed an ultra-fast and high fidelity hotspot detection flow providing full layout, feature-centric assessment as improvement over sliding window or raster scanning techniques. Under the real manufacturing conditions, we incorporated a novel set of hotspot signature measurement, a hierarchically refined classification methodology and powerful machine learning kernel implementations into an integrative flow. We implemented our algorithm with an industry-strength engine [16] under real manufacturing conditions, and showed that it significantly outperforms previous state-of-the-art algorithms in hotspot detection false alarm rate (2.4X to 2300X reduction) and simulation run-time (5X to 237X reduction), meanwhile archiving similar or slightly better hotspot detection accuracies. The demonstrated high performance makes our approach very suitable for identifying lithographic hotspots and guiding lithography-friendly physical design.

Table 7: Comparisons between existing methods

	[13]	[19]	[15]	ours
kernel	SVM	regression	ANN	ANN,SVM
window-based	yes	no	yes	no
accuracy	high	medium	high	high
false alarm	high	low	high	low
scanning coverage	trade-off	full	trade-off	full
run time	slow[a]	medium	medium[a]	very fast

[a] For scanning window based approaches , run-time can be traded-off between accuracy and coverage.

8. REFERENCES

[1] International Technology Roadmap for Semiconductors. 2009.
[2] J. Mitra, P. Yu, and D. Z. Pan. RADAR: RET-Aware Detailed Routing using Fast Lithography Simulation. In *Proc. Design Automation Conf.*, June 2005.
[3] 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.
[4] T.-C. Chen, G.-W. Liao, and Y.-W. Chang. Predictive Formulae for OPC with Applications to Lithography-Friendly Routing. In *Proc. Design Automation Conf.*, June 2008.
[5] D. Z. Pan, M. Cho, and K. Yuan. Manufacturability Aware Routing in Nanometer VLSI. In *Foundations and Trends in Electronic Design Automation*, 2010.
[6] 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. In *Proc. of SPIE*, 2003.
[7] E. Roseboom, M. Rossman, F.-C. Chang, and P. Hurat. Automated Full-Chip Hotspot Detection and Removal Flow for Interconnect Layers of Cell-Based Designs. In *Proc. of SPIE*, volume 6521, 2007.
[8] A. B. Kahng, C.-H. Park, and X. Xu. Fast Dual Graph based Hotspot Detection. In *Proc. of SPIE*, volume 6349, 2006.
[9] J. Xu, S. Sinha, and C. C. Chiang. Accurate Detection for Process Hotspots with Vias and Incomplete Specification. In *Proc. Int. Conf. on Computer Aided Design*, 2007.
[10] H. Yao, S. Sinha, C. C. Chiang, X. Hong, et al. Efficient Process Hotspot Detection using Range Pattern Matching. In *Proc. Int. Conf. on Computer Aided Design*, 2006.
[11] N. Nagase, K. Suzuki, K. Takahashi, et al. Study of Hotspot Detection using Neural Network Judgement. In *Proc. of SPIE*, volume 6607, 2007.
[12] N. Ma, J. Ghan, S. Mishra, et al. Automatic Hotspot Classification using Pattern-based Clustering. In *Proc. of SPIE*, 2007.
[13] D. G. Drmanac, F. Liu, and L.-C. Wang. Predicting Variability in Nanoscale Lithography Processes. In *Proc. Design Automation Conf.*, San Francisco, CA, 2009.
[14] J.-Y. Wuu, F. G. Pikus, A. J. Torres, et al. Detecting Context Sensitive Hot Spots in Standard Cell Libraries. In *Proc. of SPIE*, 2009.
[15] D. Ding, X. Wu, J. Ghosh, and D. Z. Pan. Machine Learning based Lithographic Hotspot Detection with Critical Feature Extraction and Classification. In *Proc. ICICDT Conf.*, 2009.
[16] CALIBRE Mentor Graphics Corp.
[17] M. Riedmiller and H. Braun. A Direct Adaptive Method for Faster Backpropagation Learning: the RPROP Algorithm. In *IEEE Int. Conf. on Neural Networks*, 1993.
[18] R.-E. Fan, P.-H. Chen, and C.-J. Lin. Working Set Selection Using Second Order Information for Training Support Vector Machines. In *Journal of Machine Learning Research*, 2005.
[19] A. J. Torres, M. Hofmann, and O. Otto. Directional 2D functions as models for fast layout pattern transfer verification. In *Proc. SPIE*, 2009.