2018 IEEE 36th VLSI Test Symposium (VTS)

Enhanced Hotspot Detection Through Synthetic Pattern Generation and Design of Experiments

Gaurav Rajavendra Reddy, Constantinos Xanthopoulos and Yiorgos Makris
gaurav.reddy@utdallas.edu, constantinos.xanthopoulos@utdallas.edu, yiorgos.makris@utdallas.edu
Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson, TX 75080, USA

***Abstract*—Continuous technology scaling and the introduction of advanced technology nodes in Integrated Circuit (IC) fabrication is constantly exposing new manufacturability issues. Design hotspots are one of such problems, which are a result of complex design and process interactions. These hotspots are known to vary from design to design and foundries expect such hotspots to be predicted early and corrected in the design stage itself, as opposed to a process fix for every hotspot, which would be intractable. Various efforts have been made in the past to address this issue by using a known database of hotspots as a source of information. Most of those works use either Machine Learning (ML) or Pattern Matching (PM) techniques to identify and predict hotspots in new incoming designs. Almost all of those methods suffer from high false-alarm rates, mainly because (i) they are oblivious to the root causes of hotspots, and (ii) a large hotspot database to learn from is generally not available. In this work, we try to address these limitations by using novel hotspot Design of Experiments (DOEs) and synthetic pattern generation approaches. We analyze the effectiveness of the proposed method against the state-of-the-art on a 45nm process, using industry standard tools and designs.**

I. Introduction

Continued technology scaling and the introduction of every advanced technology node in Integrated Circuit (IC) fabrication brings in new challenges for foundries. Lithography is a major challenge during technology development. As shown in Figure 1, in early technology nodes, the wavelength of light used in lithography was much smaller than the features being printed. In the latest nodes, this has reversed and lithography has become extremely challenging due to complex interactions between designs and sophisticated unit processes. To mitigate some of the lithography-related issues and ensure reliable manufacturing, various Resolution Enhancement Techniques (RETs) such as Optical Proximity Correction (OPC), Multi-patterning, Phase-shifted masks etc., are used. Despite employing RETs, certain areas in the design (layout), which are Design Rule Check (DRC) clean and Design For Manufacturability Guidelines (DFMGs) compliant, show abnormal and unexplained variation, causing parametric or hard defects. Such areas are termed as 'Hotspots' (popularly known as 'Lithographic hotspots' or 'Design weak-points'). The cause of hotspots is mostly attributed to their neighborhood (a set of polygons surrounding the hotspot area) which causes complex interactions of light during the lithography process. Since, hotspots vary from design to design, identifying their root causes and finding a fix for all such hotspots through process changes is extremely difficult, time consuming and expensive. Thus, in most cases, foundries create a database of known hotspots and restrict their presence in incoming customer designs. A hotspot database is usually populated through Failure Analysis (FA), inline inspections, lithographic simulations using well-calibrated lithographic models, etc. [1]. If a design pattern turns out to be a hotspot in later stages of fabrication, especially, after mask production, it may result in huge financial losses to the foundry. Hence, there is a great incentive to identify hotspots early and correct them in the design stage itself.

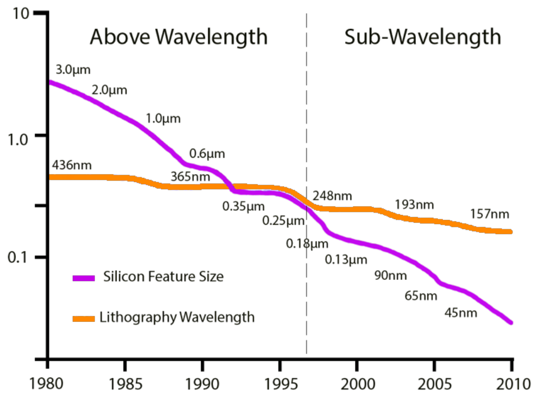

Fig. 1. Changes in lithography with silicon feature sizes [2]

Many researchers have suggested pattern matching and machine learning-based techniques to identify and predict hotspots in new incoming designs. Unlike previous works [3], [4], where the focus has been on using more and more powerful machine learning tools, we take a novel approach to improving hotspot detection by increasing the information-theoretic content of the training data that these methods use. We call this process 'Database enhancement' and it involves two procedures: (i) 'Synthetic pattern generation' and, (ii) 'Design of experiments': Combined, these procedures enable a machine learning entity to effectively learn the 'root cause features' of hotspots. These procedures are also 'method agnostic', as they can be used with any of the previously proposed hotspot detection methods to improve their performance.

The rest of the paper is organized as the following: The State-Of-The-Art (SOTA) and its limitations are explained in detail in section II. The proposed methodology is presented in section III. Experimental results are discussed in section IV and conclusions are drawn in section V.

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 02,2020 at 22:28:06 UTC from IEEE Xplore. Restrictions apply.

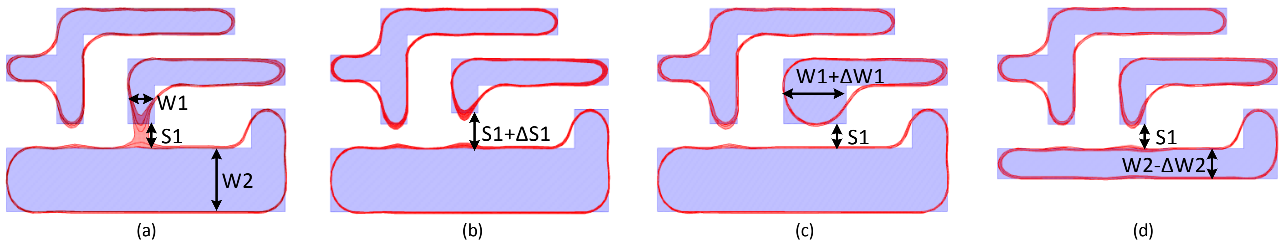

Fig. 2. (a) A hotspot pattern, (b-d) variants of pattern 'a' which are non-hotspots

II. The State-Of-The-Art And Its Limitations

Hotspot detection has been a topic of high interest in the past few years. Authors of [5], [6] have used pattern matching techniques, wherein a new design is compared to a database of previously seen hotspots and potential hotspot areas in the design are flagged. While these techniques are helpful in identifying known hotspots, they often fail in predicting unknown (never-seen-before) hotspots. To address this issue, machine learning techniques using Support Vector Machines (SVMs) [4], Artificial Neural Networks (ANNs) [7], multiple/meta classifiers [3] etc. were proposed. They essentially 'learn' (are trained) from a known database and use the trained model to make a prediction on new patterns. Over time, various flavors of these techniques have been proposed which have provided better accuracy, faster run-times etc., [8], [9]. However, most of them still suffer from high false alarm-rates, mainly because (i) these techniques are focused on learning the major differences between hotspots and non-hotspots, rather than their root cause features, and (ii) the lack of large hotspot databases limits their learning capabilities. Often, only a small number of known hotspots are used for learning [4], [5]. The 'Database enhancement' approach proposed in this work specifically addresses these issues.

A. False-Alarms In The State-Of-The-Art

The state-of-the-art machine learning-based hotspot detection techniques suffer from high false-alarm rates [3]–[5], [9]. The source of these false-alarms is illustrated using the following example. Figure 2 shows four patterns with their contours (Process Variability (PV) bands) obtained from litho simulations. Among them, pattern (a) is a hotspot due to a short between two of its polygons. Patterns (b-d) are very similar to pattern (a), but their subtle differences from pattern (a) makes them non-hotspots.

Case 1 - Let us assume that a machine learning based classifier is being trained to detect hotspots and among the patterns shown in Figure 2, only pattern (a) is part of its training dataset. During testing, if pattern (b) is presented to the classifier, it tends to classify it as a hotspot due to its close similarity to pattern (a). But, in reality, it is not a hotspot due to the increased space $S1 + \Delta S1$. The classifier made this error because it had failed to recognize $S1$ as a root cause feature of this pattern.

Case 2 - Let us assume that the classifier's training dataset includes both patterns (a) and (b). In this case, the classifier easily recognizes that the constrained space $S1$ makes this pattern a hotspot and a relaxed space $S1 + \Delta S1$ would make it a non-hotspot. Then, if pattern (c), which is very similar to patterns (a) & (b) (also having a constrained space $S1$), is presented to the classifier, the classifier tends to call it a hotspot. But in reality, it is not a hotspot, because of the increased width $W1 + \Delta W1$. Here, the classifier predicted incorrectly because, during training, it had only recognized $S1$ as a root cause feature, but not $W1$. Similarly, the feature $W2$ is also a root cause feature.

From the above example, it becomes evident that, unless otherwise trained with many variants of a known hotspot, the ML entity assumes that all polygons in a pattern equally contribute towards making it a hotspot and fails to learn the root cause features. Without such learning, it remains oblivious to the subtle variations in similar-looking patterns and tends to misclassify them, creating large amounts of false-alarms. Hence, enhancing the database with sufficient variants of known hotspots becomes imperative towards empowering an ML entity to learn effectively.

III. Proposed Methodology

The proposed hotspot detection flow is shown in Figure 3. A high-level description is provided below and its major blocks are explained in detail in the next sub-sections. This flow is typically implemented at the foundry side and executed prior to mask fabrication; yet parts of it can be potentially incorporated into the Product Design Kits (PDKs) and transferred to the customer, in order to reduce design debug cycles.

A set of known hotspots and non-hotspots gathered from prior experience form the initial database. Design of Experiments (DOEs) is performed to increase the information-theoretic content of the initial database. As part of these experiments, synthetic variants (patterns) of known hotspots are generated and subjected to process simulations (litho/litho-etch) to determine which of the patterns are hotspots. Synthetic patterns, along with the initial database, form the enhanced database. Patterns in the enhanced database are converted into numerical feature vectors. Feature vectors are, then, subjected to dimensionality reduction and a machine learning-based classifier (i.e., an SVM) is trained using the dimensionality-reduced feature vectors. The trained model is, then, stored to evaluate future incoming designs. When a foundry receives a new design from its customers, the design is decomposed into smaller patterns and predictions are made on these patterns using the trained classifier. Patterns classified as hotspots are subjected to further investigation, flagged as areas of interest for inline inspections, and drive design fixes if warranted.

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 02,2020 at 22:28:06 UTC from IEEE Xplore. Restrictions apply.

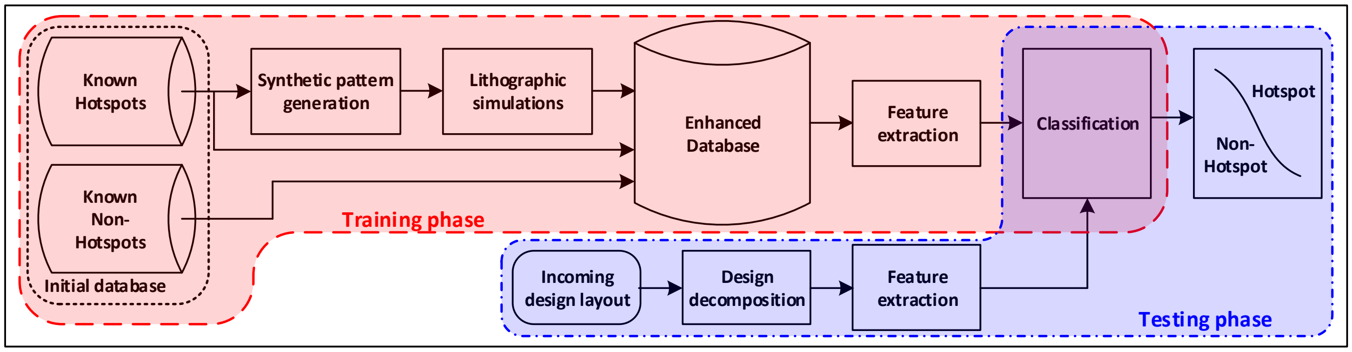

Fig. 3. The proposed machine learning-based Hotspot detection flow

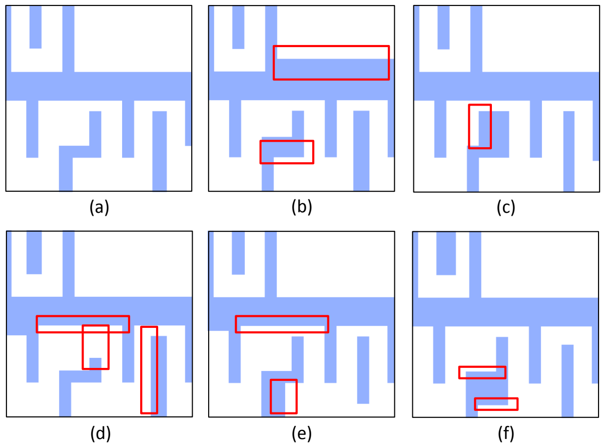

Fig. 4. (a) A hotspot pattern, (b-f) Synthetic patterns generated from pattern ‘a’. Red markers indicate the subtle differences from pattern ‘a’

A. Synthetic Pattern Generation and DOEs

For every hotspot in the initial database, multiple synthetic patterns are generated by changing one or more features at a time. Features such as corner to corner distances, jogs, line-end positions, layer spacing, layer area etc., are varied. Figure 4 (a) shows one such hotspot and Figures 4 (b-f) show some of its synthetic patterns. A time-efficient method for varying these features relies on perpendicularly moving the edges of one or more polygons in each snippet by a random distance. This approach allows to quickly generate multiple patterns whose variance can be easily controlled by two parameters. The first parameter p is the probability of any given edge to move or remain stationary. By increasing this probability, we effectively increase the number of polygons and their edges that are altered in the snippets. The second parameter d is associated with a distribution of distances, which is sampled for every polygon edge selected by the first parameter. The sampled value denotes the distance by which the edge will be displaced. These distance values follow a normal distribution centered at 0. In this way, most synthetic patterns are slight variants of the original pattern, thereby enabling us to learn the root causes effectively. However, the variation between generated patterns can be easily changed by varying the parameter d. Essentially, the parameter d can be thought of as the standard deviation of this distribution. Parameters p and d are varied based on domain knowledge and experimentation.

As expected, the above-mentioned procedure results in a plethora of patterns, many of which might not even pass the DRC. To ensure that valid layout topologies are generated and to make this process run-time efficient, we implemented a minimal DRC engine in Python, which we execute after every pattern is generated. This check ensures that most of the generated patterns are valid. However, since implementing complex design rule checks becomes complicated, all synthetic patterns which pass this minimal DRC check are also subjected to a full DRC using CalibreDRC. Through this approach, we can ensure that the vast majority of the generated patterns are DRC clean and usable. Synthetic patterns are, then, subjected to lithographic simulations to ascertain the ground truth about them. To this end, it is assumed that litho models are well-calibrated to the process, as is often the case in mature processes (with PDKs 1.0 and above). On the other hand, during early technology development, foundries may not have well-calibrated models readily available, but do have access to plenty of test-silicon. In those situations, simulation results from crude models can be used as a guide to direct actual silicon-based experiments.

The number of synthetic patterns necessary to significantly improve the information-theoretic content in the training set depends on the process node, design complexity, layer of interest etc. We have studied this dependency on a 45nm process and a detailed explanation can be found in the experimental results section. In general, these experiments are not run-time intensive, as they work with small layout snippets. Moreover, this is a one-time procedure, hence a large number of synthetic patterns could be generated. Synthetic patterns, along with their litho simulation results, are added into the initial database in order to create the enhanced database/dataset.

B. Feature Extraction

In all proposed machine learning-based hotspot detection schemes, hotspot and non-hotspot patterns are initially obtained in the form of layout snippets and then subjected to feature extraction, whereby the image snippet is transformed into a numerical feature vector which can be used to train/test a machine learning entity. Various feature extraction methods, such as bounded rectangle region-based [7], polygon fragment-based [4], concentric circle sampling-based [9], density transform [5] etc., have been proposed in the past, suited to

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 02,2020 at 22:28:06 UTC from IEEE Xplore. Restrictions apply.

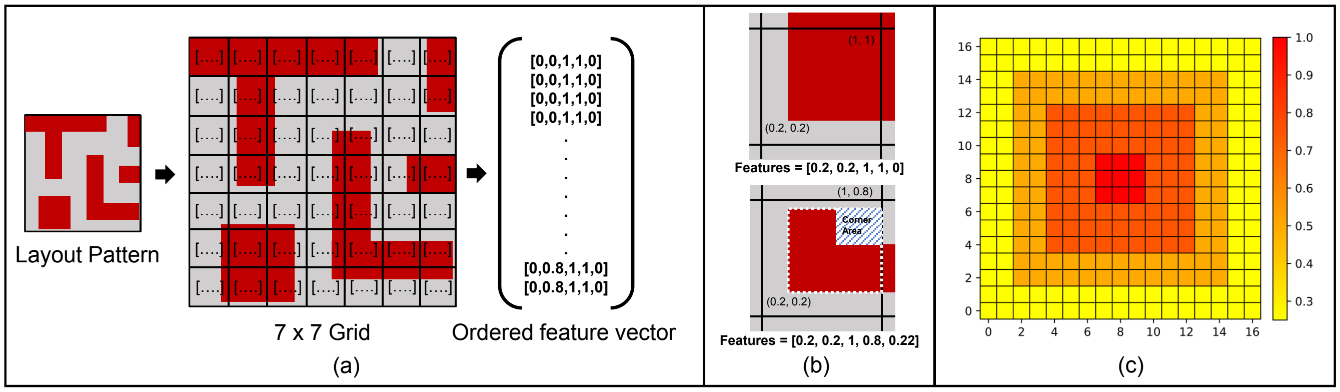

Fig. 5. Feature extraction (a) Grid based co-ordinate transform, (b) Features extracted from within a grid block. Top - features of a polygon without inner-corners, Bottom - features of a polygon with an inner-corner, (c) A plot showing the weights associated with a 17x17 grid used in this work

the detection flow they were used in. While every method has its own drawbacks and there is no clear winner among them, density transformation has been most widely used and works reasonably well. However, it fails to capture the minor variations which are crucial for effective learning.

In this work, we propose a novel feature extraction technique, which we call 'Co-ordinate Transform'. In co-ordinate transformation, as illustrated in Figure 5(a), an $n * n$ grid is overlapped on a pattern, where n is the number of blocks on each dimension of the grid. n is chosen such that each block covers only one, or a part of one polygon in the pattern. n is given by $(pattern\_size/layer\_half\_pitch)$. Polygons present within each block are transformed into five features: {bottom-left x co-ordinate, bottom-left y co-ordinate, top-right x co-ordinate, top-right y co-ordinate, corner info}. While the first four features capture most of the information, they fail to capture the inner corner information. Hence, a fifth feature 'corner info', shown in Figure 5(b), is necessary. This is given by the difference between the $polygon\_bounding\_box\_area$ $(area\ in\ white\ dotted\ line)$ and the $layer\_area$ $(area\ in\ red)$. Features from every block of the grid are extracted and their ordered vector makes the final feature vector. While the five features capture the geometric information of polygons, their ordering captures the location of those polygons within the pattern.

Comparison to density transform: In density transform, different polygons may result in the same density value. For instance, if a polygon with area P_a is present inside a block with area B_a, the density of that block will remain P_a/B_a, irrespective of whether the polygon has a corner, is oriented vertically or horizontally, occupies the top, bottom, left or right portions of the block. To reduce the information loss due to these issues, density transform is often used with very fine grids (having large number of blocks). But such fine grids result in large number of features with very little variation within the features. This increases the ML model complexity and leads to over-fitting. On the other hand, co-ordinate transform is free from such issues as it uses co-ordinates to capture polygon information and uses coarse grids which, in turn, creates fewer features yet with more variation.

Weighted features: As an option, during co-ordinate transformation, weights can be assigned to various blocks of the grid, as shown in Figure 5(c). Typically, minor variations in the central area of patterns have high influence in causing hotspots, while this influence fades as we move towards the periphery. Given enough data, an effective machine learning entity can learn this variation by itself, but, adding domain knowledge like this helps to work with smaller datasets, faster training times etc.

C. Classification

Hotspot detection requires a robust two-class classifier which can learn a separation boundary between hotspots and non-hotspots with maximum margin. In this work, a non-linear SVM is used with a Radial Basis Function (RBF) kernel. As a pre-processing step, dimensionality reduction is performed using Principal Component Analysis (PCA). While retaining most of the variation in the dataset, PCA reduces the number of features into a smaller subset called 'principal components'. Working with principal components aids data visualization, reduces ML model complexity etc. Detailed discussion of SVMs and PCA is out of the scope of this work; For more information, the reader is referred to [10] and [11] respectively.

Class balancing: Typically, the number of known hotspots patterns available for training is smaller compared to known non-hotspot patterns. Training with such imbalanced datasets results in a skewed classifier and the classifier tends to favor the dominating class. To avoid this, the minority class is re-sampled (replicated) and the class sizes are equalized. We noted that, by doing so, we do not alter the information-theoretic content of the dataset; we only help evade the skewed learning problem. Handling imbalanced datasets is discussed in detail in [12].

D. Layout Decomposition

When a foundry receives a new design from its customers, in order to perform hotspot detection, the design needs to be decomposed into patterns. As shown in Figure 6(a), patterns have two important attributes: a) *Hotspot region:* The region where actual defects (shorts or opens) are anticipated. b)

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 02,2020 at 22:28:06 UTC from IEEE Xplore. Restrictions apply.

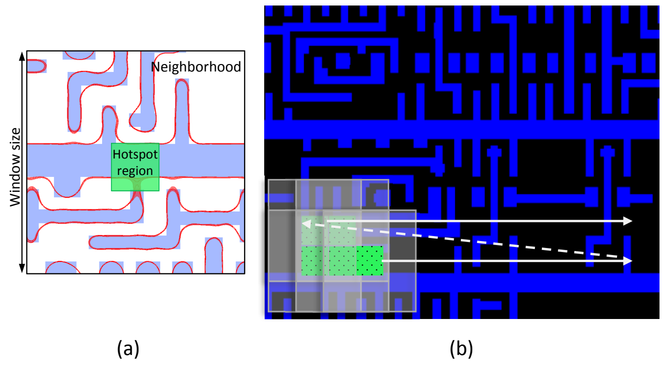

Fig. 6. Layout decomposition into patterns (a) Attributes of a pattern (b) Sliding window procedure

Neighborhood: The area surrounding the hotspot region, which could have possibly caused the defect. The extent (size) of the neighborhood varies from technology node to node and is often determined through experimentation. It is safe to use a larger neighborhood than suspected, because, if the polygons towards the periphery of the pattern are indeed benign, an effective machine learning entity would ignore them by assigning low weights to corresponding features. The bounding box encapsulating the neighborhood is popularly known as a 'Window'. To capture patterns from a layout, a moving window based approach is used, as shown in Figure 6(b). The step size for the window movement in both x and y directions should be carefully selected to ensure that the entire layout is covered and every part of the layout appears within the hotspot region of at least one pattern. Patterns obtained from this step form the test set and are tested for hotspots using the trained classifier.

IV. Experimental Results

The objective of this work is to show that enhancing the training dataset using synthetic patterns indeed increases its information-theoretic content and, in turn, reduces false-alarms. To demonstrate this, we implemented the machine learning-based hotspot detection flow shown in Figure 3. The classifier in this flow is trained with and without an enhanced dataset and tested against a common testing dataset. In the rest of the paper, we refer to the classifier trained with an enhanced dataset as 'enhanced classifier' and to the one trained with the non-enhanced dataset as 'non-enhanced classifier'. The non-enhanced classifier is the State-Of-The-Art (SOTA). The difference between the prediction results of the two classifiers indicates the effectiveness of this approach.

For this analysis, we obtained a Register-Transfer-Level (RTL) code of an Advanced Encryption Standard (AES) encryption core from [13]. To create the design layout, the RTL code was synthesized, placed and routed using the Nangate open cell library [14] which is based on a 45nm PDK [15].

Generating the non-enhanced dataset: From a randomly chosen area of the AES layout, 40,000 patterns are captured on Metal1 (M1) layer using the Calibre Pattern Match tool. Litho simulations are performed using Calibre Litho-Friendly-Design (LFD) tool-kit [16], with the litho models provided in the PDK. Simulation results ascertain the ground-truth for the captured patterns. 50% of this dataset is kept aside as the test set and never used in the training process. The other 50% is used for training and is referred to as 'non-enhanced dataset'[1].

TABLE I
Experimental Setup

Parameter	Value
Patterns in the initial dataset	40,000
Patterns used for training	20,000 (50%)
Patterns used for testing	20,000 (50%)
Synthetic patterns used for enhancement (per hotspot in the training set)	200
Training set size after class balancing (non enhanced dataset)	38,426
Training set size after class balancing (enhanced dataset)	246,562
Synthetic patterns added to the test set	≈280 (per hotspot in the training set)
Test set size after class balancing and synthetic pattern addition	238,408
Window size (each side)	$8.5 * layer\_pitch$
Window step size (x and y direction)	$1.5 * layer\_pitch$
Grid size	$17 * 17$
Number of features	1445
Features used after PCA	300

Generating the enhanced dataset: The non-enhanced dataset generated in the previous step contains 787 hotspots. For every one of these hotspots, 500 synthetic variants are generated. Of them, approximately 480 passed DRC and, among them, 200 are used in training. Litho simulations are performed on all DRC-clean synthetic patterns and their ground truth (i.e. hotspot vs. non-hotspot) is obtained. The synthetic patterns along with the patterns in the non-enhanced dataset, together form the 'enhanced dataset'.

Both the enhanced and non-enhanced datasets are generated using co-ordinate transform and have weighted features as explained in section III-B.

About 280 synthetic patterns per hotspot, which are not used in training, are added to the common test set. Note that, these are DRC-clean patterns which could potentially occur in any future incoming designs. As explained in section II-A, these are the type of patterns which are often misclassified due to the subtle variations among them. Hence, it is imperative to have them in the test set, in order to validate whether the trained model is robust enough to classify them correctly.

As explained in section III-C, an SVM with an RBF kernel is used as a two-class classifier. While training a classifier, or any machine learning based entity, some of the model parameters require 'tuning' to learn effectively. In this case, to be fair, we have ensured that, both enhanced and non-enhanced

[1]Some of the prior works [5], [8], [9] have used the pattern database provided by ICCAD '12 CAD contest [17] as a benchmark. While we sought to use the same, that dataset is from 28nm and 32nm technologies for which Litho models are not publicly available. Few other works [3], [4] have used a set of designs whose source is not published. Hence, for this analysis, design layout was generated using open source RTL codes.

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 02,2020 at 22:28:06 UTC from IEEE Xplore. Restrictions apply.

TABLE II
EXPERIMENTAL RESULTS

Metric	SOTA	This work	Formula Used
hotspot hit rate	89.20%	95.70%	$\frac{predicted\_hotspots}{total\_hotspots}$
non-hotspot hit rate	41.50%	74.18%	$\frac{predicted\_non\_hotspots}{total\_non\_hotspots}$
Correct prediction rate	70.60%	87.31%	$\frac{hotspot_{hits}+non\_hotspot_{hits}}{total\_patterns\_tested}$
False positive rate	22.81%	10.06%	$\frac{false\_pos}{total\_patterns\_tested}$
False negative rate	6.59%	2.63%	$\frac{false\_neg}{total\_patterns\_tested}$
Total prediction error	29.40%	12.69%	$\frac{false\_pos+false\_neg}{total\_patterns\_tested}$
Total improvement from this work - reduction in prediction error by 56.8% (% change from 29.40% to 12.69%)			$\frac{tot\_err_{SOTA}-tot\_err_{this\_work}}{tot\_err_{SOTA}}$

classifiers are performing at their best by tuning their model parameters 'C' and 'gamma' using grid-search methods [18].

Training datasets are subjected to class balancing as explained in section III-C. Final training and testing set sizes, and other setup parameters are reported in Table I. The test set is tested by both the enhanced and non-enhanced classifiers. Results are reported in Table II. The results clearly indicate that the enhanced classifier performs better and has reduced the classification error by about 57%.

The number of synthetic patterns to be generated has to be decided by the user. To aid this process, we performed a study where, a non-enhanced dataset of 5,000 patterns was obtained and the number of synthetic patterns used to enhance the dataset was varied. As seen in Figure 7, increasing the number of patterns continuously reduces the classification error. When the error due to the addition of 0 synthetic patterns is taken as baseline, we can observe that, an addition of a mere 40 synthetic patterns (per known hotspot) reduces classification error by about 36% (% change from 29.40% to 18.7%). This testifies the effectiveness and the practicality of this approach.

V. CONCLUSION

We have discussed the problem of lithographic hotspots in advanced technology nodes, analyzed the state-of-the-art in this domain and highlighted that they suffer from high false-alarm rates. To address these issues, we have proposed a novel database enhancement approach which involves synthetic pattern generation and design of experiments. We have implemented the proposed flow using a 45nm PDK and have demonstrated about 57% reduction in classification error in comparison to the state-of-the-art.

ACKNOWLEDGMENT

This research has been partially supported by Semiconductor Research Corporation (SRC) through task 2709.001.

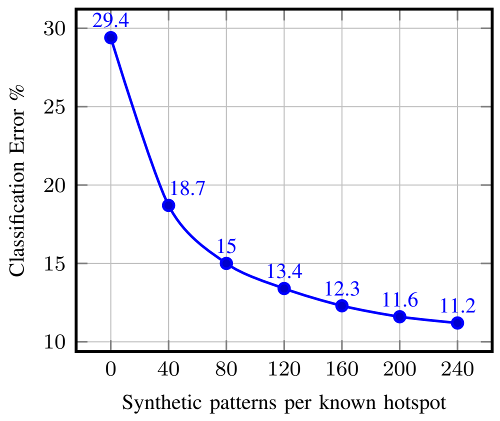

Fig. 7. Variation of prediction error w.r.t number of synthetic patterns used

REFERENCES

[1] V. Dai, L. Capodieci, J. Yang and N. Rodriguez, "Developing DRC plus rules through 2D pattern extraction and clustering techniques," *SPIE Advanced Lithography*, pp. 727517 – 727517–10, 2009.
[2] "Synopsys," http://www.monolithic3d.com/blog/category/3d%20technology/2.
[3] Duo Ding, Bei Yu, J. Ghosh and D. Z. Pan, "EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation," in *17th Asia and South Pacific Design Automation Conference*, 2012, pp. 263–270.
[4] D. Ding, J. A. Torres and D. Z. Pan, "High performance lithography hotspot detection with successively refined pattern identifications and machine learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 11, pp. 1621–1634, 2011.
[5] W. Y. Wen, J. C. Li, S. Y. Lin, J. Y. Chen and S. C. Chang, "A fuzzy-matching model with grid reduction for lithography hotspot detection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 11, pp. 1671–1680, 2014.
[6] H. Yao, S. Sinha, C. Chiang, X. Hong and Y. Cai, "Efficient process-hotspot detection using range pattern matching," in *IEEE/ACM International Conference on Computer-aided Design*, 2006, pp. 625–632.
[7] D. Ding, X. Wu, J. Ghosh and D. Z. Pan, "Machine learning based lithographic hotspot detection with critical-feature extraction and classification," in *IEEE International Conference on IC Design and Technology*, 2009, pp. 219–222.
[8] K. Madkour, S. Mohamed, D. Tantawy and M. Anis, "Hotspot detection using machine learning," in *17th International Symposium on Quality Electronic Design*, 2016, pp. 405–409.
[9] H. Zhang, Bei Yu and E. F. Y. Young, "Enabling online learning in lithography hotspot detection with information-theoretic feature optimization," in *IEEE/ACM International Conference on Computer-Aided Design*, 2016, pp. 1–8.
[10] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag New York, Inc., 1995.
[11] S. Wold, K. Esbensen and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 1, pp. 37 – 52, 1987.
[12] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.
[13] "Opencores," http://opencores.org/.
[14] "Nangate OCL," https://www.nangate.com/?page_id=22.
[15] "FreePDK45," https://www.eda.ncsu.edu/wiki/FreePDK.
[16] "Calibre-LFD," https://www.mentor.com/products/ic_nanometer_design/design-for-manufacturing/calibre-lfd/.
[17] J. A. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *IEEE/ACM International Conference on Computer-Aided Design*, 2012, pp. 349–350.
[18] "GridsearchCV," http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

Authorized licensed use limited to: Univ of Texas at Dallas. Downloaded on August 02,2020 at 22:28:06 UTC from IEEE Xplore. Restrictions apply.