On Improving Hotspot Detection Through Synthetic Pattern-Based Database Enhancement

Gaurav Rajavendra Reddy, *Member, IEEE*,
Constantinos Xanthopoulos, *Member, IEEE*,
Yiorgos Makris, *Senior Member, IEEE*

***Abstract*—Design hotspots are layout patterns which may cause defects due to complex design and process interactions. Several machine learning and pattern matching-based methods have been proposed to identify and correct them early during design stages. However, almost all of them suffer from high false-alarm rates, mainly because they are oblivious to the root causes of hotspots. In this work, we seek to address this limitation by using a novel database enhancement approach through synthetic pattern generation based on carefully crafted design of experiments. We evaluate the effectiveness of the proposed method using industry-standard tools and designs and demonstrate more than 3X reduction in classification error in comparison to the state-of-the-art.**

I. Introduction

Photolithography continues to remain at the heart of Integrated Circuit (IC) fabrication. In the latest nodes, it has become extremely challenging due to complex design and process interactions. 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 pass Design Rule Checks (DRCs) and comply with Design For Manufacturability Guidelines (DFMGs), 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 (i.e., a set of polygons surrounding the hotspot area) which causes complex interactions of light during the lithography process. Discovering hotspots in later stages of fabrication, especially after mask production, may result in significant financial losses to the foundry. Therefore, there is a great incentive to identify and correct them early in the design stage itself.

Hotspot detection has been a topic of high interest in the past decade. Authors of [1] proposed Pattern Matching (PM) techniques, wherein a new design is compared to a database of previously seen hotspots and potential hotspot areas are flagged. While these techniques are helpful in quickly analyzing large layouts and identifying known hotspots, they also cause large numbers of false alarms. To address this issue, Machine Learning (ML) based methods were proposed. These methods essentially "learn" (are trained) from a known database and use the trained model to make predictions on new patterns. In the past decade, we have witnessed several variants of ML-based hotspot detection flows, wherein, the usage of Support Vector Machines (SVMs), Artificial Neural Networks (ANNs), multiple/meta classifiers [2], imbalance-aware learning [3], hybrid PM-ML solutions [4], and more recently, deep learning [5] was proposed. Most of these methods, however, still suffer from high false-alarm rates when exposed to Hard-To-Classify (HTC) patterns. This is mainly because these techniques are oblivious to the root causes of hotspots and ignore the fine nm-level differences between similar-looking patterns, which play a significant role in making a pattern a hotspot or a non-hotspot.

In this work, we take a novel approach to improving hotspot detection by increasing the information-theoretic content of the training dataset. We call this process "database enhancement" and it involves *synthetic pattern generation* and *design of experiments*. Combined, these procedures enable a machine learning entity to effectively learn the "root cause features" of hotspots. They are also "method agnostic", as they can be used with any of the previously proposed hotspot detection methods to improve their performance.

G. R. Reddy, C. Xanthopoulos, and Y. Makris are with the Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: gaurav.reddy@utdallas.edu; constantinos.xanthopoulos@utdallas.edu; yiorgos.makris@utdallas.edu).

II. Proposed Methodology

The proposed hotspot detection flow is shown in Figure 1. A set of known hotspots and non-hotspots gathered from prior experience form the initial database. Design of Experiments (DOEs) is then performed to *increase the information-theoretic content* of the initial database. As a 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 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, it transforms it into patterns and feature vectors, and predictions are made using the trained classifier. Patterns classified as hotspots are subjected to further investigation, flagged as areas of interest for inline inspections, and used to drive design changes.

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 2(a) shows one such hotspot and Figures 2(b-f) show some of its synthetic variants. A time-efficient method for varying these features relies on perpendicularly moving the edges of one or more polygons in each snippet by a randomly sampled distance. This approach allows to quickly generate multiple patterns whose variance can be 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 (Probability Density Function (PDF)), 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. Parameters p and d can be varied based on Process Design Kit (PDK)/domain-specific information. For instance, higher p values can be used in Extreme Ultra-Violet (EUV) lithography-based processes because of their relaxed design rules in comparison to Deep Ultra-violet (DUV) lithography [6]. Similarly, the parameter d may depend on the manufacturing grid size, permitted widths/spaces, etc. The pattern generation procedure is detailed in Algorithm 1.

As expected, the above-mentioned procedure results in a plethora of patterns, many of which might not 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2021.3049285, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

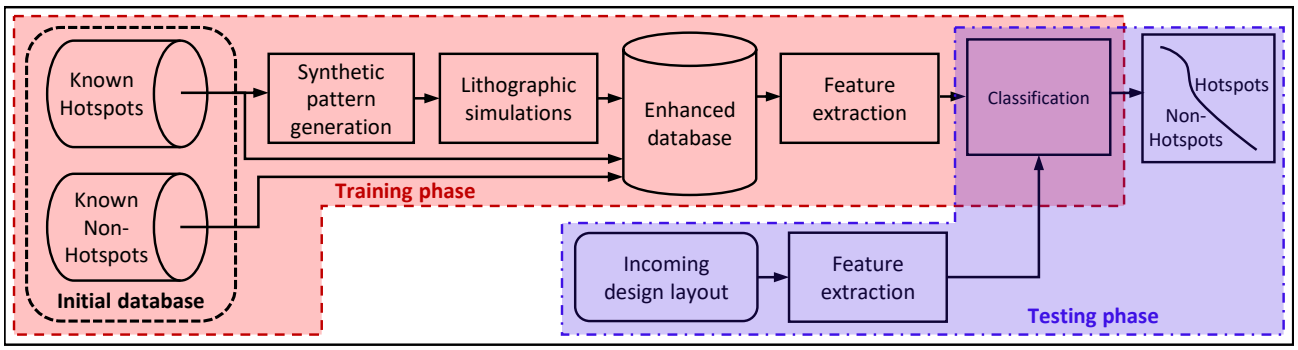

Fig. 1. The proposed machine learning-based Hotspot detection flow

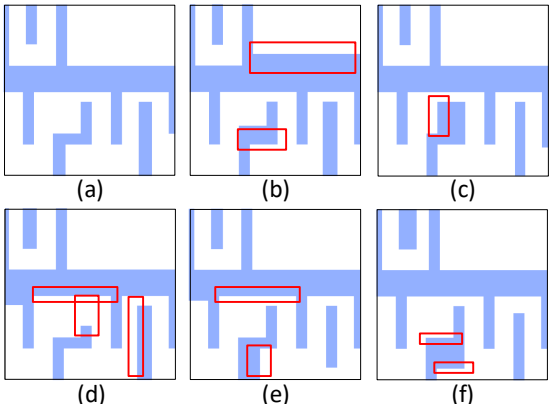

Fig. 2. (a) A hotspot pattern, (b-f) Synthetic patterns generated from pattern (a). Red markers indicate the subtle differences from pattern (a)

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 (or validated through test-silicon-based experiments [7]) to ascertain the ground truth about them. 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 this work, we implement a slightly varied version of the Fragment Transform (FT) [8] method, which we call Fragment Transform Plus (FTP). It is a sophisticated Feature Extraction (FE) method, wherein an entire layout is subjected to fragmentation using OPC tools and transformed into a large set of fragments, as shown in Figure 3(a). Such an abstraction makes this FE method flip-, mirror- and rotation-invariant. Post-fragmentation, every fragment is uniquely identified and analyzed individually to make hotspot/non-hotspot decisions. To obtain the context of a fragment, a metric called Radius Of Influence (ROI) is used. The ROI is determined by the lithography tools and the wavelength of light used for patterning. As shown in Figure 3(a), if a circle with radius ROI is drawn by centering on the fragment under consideration, it is assumed that this circle encloses all the fragments that play a role in causing a hotspot at its center. The fragment under consideration is called the *primary fragment*. Its parallel neighbors on either side, along a line perpendicular to its surface, are called *secondary fragments*. The lateral neighbors of both primary and secondary fragments are called *tertiary fragments*. The primary fragment, along with its secondary and tertiary fragments, together can be regarded as a "pattern". To accurately capture the characteristics of a pattern, the following set

```
def GenerateSyntheticPatterns(KnownHotspot):
    Input: A Known Hotspot, Synthetic pattern count, distance PDF,
           Edge PDF
    Result: Synthetic variants of the Hotspot
1   for i in range(SynPatCount):
2       HotspotPolys = All polygons in the original hotspot pattern
3       for polygon in HotspotPolys:
            /* Sample the no. of edges to be varied
               */
4           EdgeCount = Sample from Edge PDF
5           for j in range(EdgeCount):
6               while EdgeAttempts ≤ MaxEdges:
                    /* Randomly select an edge      */
7                   edge = GetRandomEdge(polygon)
8                   while DistAttempts < FixedCount:
9                       dist = Sample from distance PDF
10                      polygon = polygon.MoveEdge(edge, dist)
                        /* Perform checks to avoid
                           simple DRC errors        */
11                      MinimalDRC(ModifiedPattern)
12                      if MinimalDRC == Pass:
13                          go to line 5
14                      else:
15                          polygon = UnmodifiedPolygon
16                          DistAttempts+ = 1
17                          try a different dist value (go to line 8)
18                  EdgeAttempts+ = 1
19                  try a different edge (go to line 6)
        /* All polygons with/without updates,
           together form the modified pattern   */
20      SyntheticPattern = All Polygons (including modifications)
    /* Return patterns with variations          */
21  return SyntheticPatterns
```

Algorithm 1: Synthetic pattern generation

of parameters are measured for every fragment within a pattern[1]:

$$fragment\_parameters = [len,\ ext\_space,\ int\_space,\ C\_corn,\ AC\_corn,\ F0\_offset] \quad (1)$$

where: len: length of a fragment; ext_space: distance to the externally opposite fragment; int_space: distance to the internally opposite fragment; C_corn: corner information (convex/concave/no_corner) from the clockwise end of the fragment; AC_corn: corner information from the anti-clockwise end of the fragment; $F0\_offset$: offset of secondary fragments w.r.t. the location of the primary fragment.

The fragment parameters for a secondary fragment of a sample pattern are depicted in Figure 3(b). To generate the complete feature vector, the parameters from all fragments within a given ROI are concatenated together. Given that the same ROI can contain different

[1] $F0\_offset$ is not measured for primary and tertiary fragments. ext_space is not measured for some secondary and tertiary fragments which are along the periphery of a pattern. Such features are omitted because they are either redundant or they are unnecessary to accurately capture the information within a pattern.

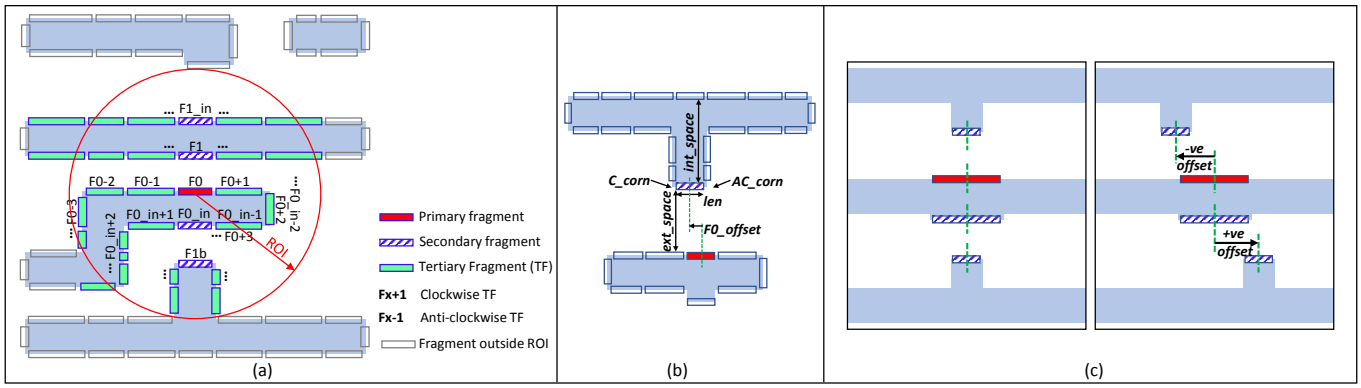

Fig. 3. Feature extraction (a) Key components of Fragment Transform Plus (FTP), (b) Fragment parameter measurements corresponding to a secondary fragment (c) Patterns illustrating the difference between FT and FTP

number of fragments in different instances, similar to [8], the ROI is abstracted as *neighboring fragment depth* in order to ensure that all patterns result in the same number of features/dimensions. Essentially, a fixed number of perpendicularly opposite fragments, as well as lateral fragments, are considered as the neighboring fragments of the primary fragment. The depth value is chosen such that all fragments within the ROI are included in the pattern.

Differences between FT and FTP: To minimize the information loss during the FE procedure, as well as the total number of resultant features, we made the following changes to the original FT method:
(1) A new fragment parameter $F0\_offset$ is added: Although not apparent in [8], the FT method fails to accurately capture the spatial arrangement of fragments which are located slightly farther from the primary fragment. For instance, the previously proposed FT method produces the same feature vector for both patterns shown in Figure 3(c), even though they are slightly different from each other. As noted by the authors of [3], even minor nm-level variation could mean the difference between a pattern becoming a hotspot or a non-hotspot. Therefore, it is necessary to accurately capture such differences between patterns while performing FE. To avoid such *transformation loss*, we introduce the new fragment parameter called $F0\_offset$. $F0\_offset$ is the offset of the center of a secondary fragment w.r.t. the center of the primary fragment, along the axis of orientation of the primary fragment. $F0\_offset$ is measured for all secondary fragments and it can be either positive or negative. Offset of the secondary fragment in the anticlockwise direction of the primary fragment is considered as negative, whereas offset in the clockwise direction is considered positive.
(2) Fragment orientation is omitted: Fragment-based FE methods are preferred to be mirror-, flip-, and rotation-invariant. Such orientation-invariance assists in generating smaller training datasets. The previously proposed FT method, however, includes *fragment orientation* as one of its features which makes it orientation-specific. Therefore, to make FTP truly orientation invariant, we omit the *fragment orientation* parameter[2].
(3) Both clockwise and anticlockwise corners of a fragment are considered: As per [8], it is unclear whether the FT method records information related to both corners of a fragment. In FTP, however, both clockwise and anticlockwise corner information is considered.

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 with a Radial Basis Function (RBF) kernel is used.

[2]Orientation-invariant FE must be used only on metal layers which are patterned using symmetric illumination shapes in scanner optics. In cases of asymmetric illumination, the orientation feature must be included as part of the feature vector.

III. 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 ML-based hotspot detection flow shown in Figure 1. 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 the proposed approach.

A. Experimental Setup

To generate baseline designs for our analysis, we obtained several Register-Transfer-Level (RTL) codes from OpenCores, synthesized, placed and routed them using the Nangate Open Cell Library (OCL), which is based on a 45nm PDK [9]. All layouts were subjected to full DRC and were found to be DRC clean. The Metal1 (M1) layer of these layouts was subjected to computation-intensive full-chip lithographic simulations using the Calibre Litho-Friendly-Design (LFD) tool-kit and the litho models provided in the PDK. These simulations ascertained the ground truth by identifying all the hotspots in the layouts. All layouts were then converted to patterns and feature vectors using the FTP method described in Section II-B. To implement FTP, we used Calibre OPCpro for fragmentation and Calibre Standard Verification Rule Format (SVRF) technology for feature extraction. An ROI of $500nm$ was considered. The ROI value was abstracted as a neighboring fragment depth value of 4. After filtering out the redundant features, which are inherently created by the FTP method, the resulting dataset consisted of 519 features.

Non-Enhanced Training Dataset: In reality, to train hotspot detection models, foundries gather patterns from the first few designs manufactured in a given technology node. This dataset is usually significantly smaller in comparison to the large number of patterns which will be tested using the trained models over the lifetime of the node. To replicate such a scenario, we randomly sample two layouts and obtain a small dataset containing a total of only 100,000 patterns, which we use as our non-enhanced training dataset.

Enhanced Training Dataset: The non-enhanced dataset generated in the previous step contains 1932 hotspots. For every one of these hotspots, we used our method to generate 500 synthetic variants. Of them, an average of approximately 484 passed DRC and, among

TABLE I
TRAINING AND TESTING DATASETS

Layout	Size	Pattern Count	HT#	NHT#
NON-ENHANCED TRAINING DATASET				
wb_conmax	205 µm × 205 µm	50,000	1,236	48,764
Ethernet	360 µm × 360 µm	50,000	696	49,304
Total	Not applicable	100,000	1,932	98,068
ENHANCED TRAINING DATASET				
Non-Enhanced Dataset	Not applicable	100,000	1,932	98,068
Synthetic patterns	Not applicable	386,136	192,680	193,456
Total	Not applicable	486,136	194,612	291,524
TESTING DATASET				
SPI	66 µm x 66 µm	326,394	5,768	320,626
TV80	96 µm x 96 µm	789,253	16,736	772,517
AES (Encrypt)	160 µm x 160 µm	1,999,419	47,700	1,951,719
HTC patterns	Not applicable	384,736	191,345	193,391
Total	Not applicable	3,499,802	261,549	3,238,253

them, about 200 were randomly chosen for training. Litho simulations were performed on all DRC-clean synthetic patterns in order to obtain the ground truth (i.e., whether they are hotspots or non-hotspots). The synthetic patterns along with the patterns in the non-enhanced dataset, together form the "Enhanced Training Dataset".

Testing Dataset: To mimic the real-life scenario wherein new layouts are tested against pre-trained hotspot detection models, we evaluate the effectiveness of our method using three complete layouts which were never used during training. All patterns from these three layouts, together, comprise our testing dataset.

Further details about all the datasets are shown in Table I.

Hard-To-Classify Test Patterns: The authors of [10] have performed an interesting study, wherein they compare the layout patterns used during Technology Development (TD) against the patterns found in product designs. They discovered that, while some patterns in the product designs were topologically similar to the patterns seen during TD, the product designs had many more dimensional variations of those patterns. In a separate study, authors of [3] note that even minor variations in the widths and spaces between polygons of a pattern can mean the difference between a pattern becoming a hotspot or a non-hotspot. If we extend the results of these studies to hotspot detection, we can envision a scenario wherein a hotspot detection model is trained using a certain hotspot but tested with many more variations of that same hotspot, which could be either hotspots or non-hotspots. Such non-hotspot patterns, which closely resemble the known hotspot, are the real source of false alarms as they lie in close proximity to the training hotspots in the hyper-dimensional space and are truly hard-to-classify.

Upon detailed analysis, we found that the test layouts (i.e., the designs SPI, TV80 and AES, shown in Table I), do not contain such HTC patterns within them. Therefore, to mimic a scenario witnessed by an actual foundry, we further expanded the testing dataset by adding approximately 200 synthetic patterns – which were never used during training – for each known hotspot. These act as HTC patterns in the testing dataset and assist in determining whether the trained model is truly robust in preventing false alarms. Details about these patterns are shown in Table I. In the rest of the paper, test patterns other than the HTC patterns (i.e., patterns from SPI, TV80 and AES designs), are referred to as Easy-To-Classify (ETC) patterns.

To further demonstrate the importance of including HTC patterns in the testing dataset and to contrast their distribution against the ETC patterns, we perform Principal Component Analysis (PCA) on the training dataset and project both the ETC and the HTC patterns onto the same space. For the sake of brevity, we plot only 10 randomly

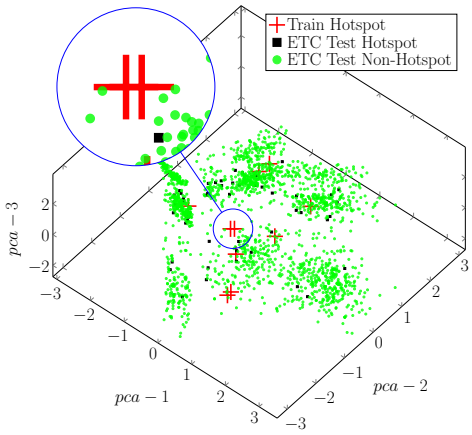

Fig. 4. Distribution of ETC test patterns w.r.t. training hotspots

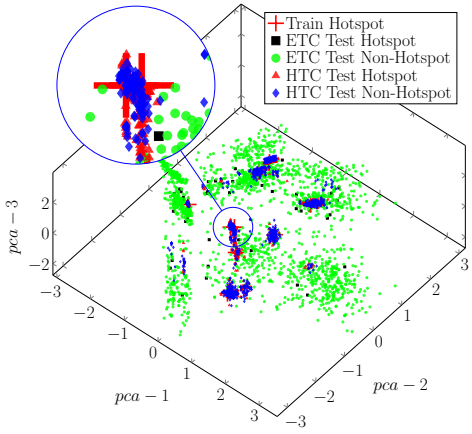

Fig. 5. Distribution of ETC and HTC test patterns w.r.t. training hotspots

sampled hotspots from the training dataset, their corresponding HTC test patterns (2000 in total), and the same number of randomly sampled ETC test patterns. Figure 4 shows the distribution of just the ETC test patterns w.r.t. to the training hotspots. Figure 5 is similar to Figure 4 but it also includes the HTC test patterns. By contrasting the two figures, we observe that HTC test non-hotspots are located in much closer proximity to the training hotspots, when compared to the proximity of ETC test non-hotspots to training hotspots. This makes HTC test non-hotspots more prone to be misclassified as hotspots and, therefore, their presence in the testing dataset is vital for accurate evaluation of the effectiveness of hotspot detection methods.

B. Experimental Analysis

We use an SVM with an RBF kernel as a two-class classifier. We used the grid search method with 3-fold cross validation and swept through 546 hyper-parameter combinations for both non-enhanced classifiers (SOTA) and enhanced classifiers (proposed) to choose their optimal hyper-parameters, as detailed in [11]. They were trained using their corresponding training datasets shown in Table I, and tested using the common testing dataset. The formulas for various metrics used in our analysis are the same as in [12]. The test results for ETC patterns and HTC patterns are shown in Table II. We observe that with average hotspot hit rates of about 89% and false positive rates less than 2%, both the SOTA and the proposed method perform similarly well on ETC test patterns. In case of HTC patterns, however, the SOTA shows very high rate of false positives. The proposed method, on the other hand, reduces false positives by about 69% (% change from 40.42% to 12.41%) in comparison to the SOTA. We also obtained the Matthews Correlation Coefficient (MCC) which is an effective indicator of the quality of two-class classification, especially

TABLE II
TEST RESULTS FROM NON-ENHANCED AND ENHANCED CLASSIFIERS

Test Layout	SOTA (Non-enhanced)						This work (Enhanced)					
	HT hit rate (%)	NHT hit rate (%)	FP rate (%)	FN rate (%)	Total Err. rate (%)	MCC	HT hit rate (%)	NHT hit rate (%)	FP rate (%)	FN rate (%)	Total Err. rate (%)	MCC
TEST RESULTS FOR ETC PATTERNS (WITH DIMENSIONALITY REDUCTION)												
SPI	91.71	98.47	1.50	0.15	1.65	0.68	92.27	99.10	0.88	0.14	1.02	0.77
TV80	86.97	97.92	2.04	0.28	2.31	0.63	88.61	98.01	1.95	0.24	2.18	0.65
AES	87.27	98.03	1.92	0.30	2.22	0.66	88.30	98.02	1.93	0.28	2.21	0.67
Average	88.65	98.14	1.82	0.24	2.06	0.66	89.73	98.38	1.59	0.22	1.80	0.70
TEST RESULTS FOR HTC PATTERNS (WITH DIMENSIONALITY REDUCTION)												
HTC Patterns	**97.78**	**19.58**	**40.42**	**1.10**	**41.52**	**0.28**	**99.27**	**75.31**	**12.41**	**0.36**	**12.77**	**0.77**
TEST RESULTS FOR ETC PATTERNS (WITHOUT DIMENSIONALITY REDUCTION)												
SPI	91.47	98.56	1.41	0.15	1.56	0.69	92.29	99.16	0.82	0.14	0.96	0.78
TV80	86.21	98.04	1.92	0.29	2.21	0.64	88.56	98.10	1.86	0.24	2.10	0.66
AES	86.50	98.15	1.81	0.32	2.13	0.67	88.04	98.11	1.85	0.29	2.13	0.67
Average	88.06	98.25	1.71	0.25	1.97	0.67	89.63	98.46	1.51	0.22	1.73	0.70
TEST RESULTS FOR HTC PATTERNS (WITHOUT DIMENSIONALITY REDUCTION)												
HTC Patterns	**97.64**	**20.14**	**40.14**	**1.18**	**41.32**	**0.28**	**99.27**	**75.85**	**12.14**	**0.36**	**12.50**	**0.77**

while handling imbalanced datasets. The MCC value ranges from -1 to +1, indicating the quality of prediction from random to perfect, respectively. The MCC value for the enhanced classifier is about 0.7 for predictions on ETC patterns and about 0.77 for predictions on HTC patterns, thereby, confirming its effectiveness.

The first half of the results shown in Table II were obtained using dimensionality-reduced datasets. Specifically, PCA was used for dimensionality reduction and only the first 250 principal components were used for training and testing. To verify whether PCA introduces any additional error into the analysis, we repeated the experiments without PCA (i.e., using all 519 features). The results are shown in the second half of Table II. By comparing the corresponding results with and without dimensionality reduction, we observe that the difference across all metrics is less than 0.6%, indicating that dimensionality reduction does not introduce any significant error into the analysis. Supplemental experimental analyses can be found in [11].

C. Applicability to Newer Technology Nodes

Owing to their extremely complex fabrication processes, newer technology nodes – such as $10\,\mathrm{nm}$ and $7\,\mathrm{nm}$ – introduce a large number of design constraints. Therefore, for our method to remain effective in synthetically enriching the information-theoretic content of hotspot databases in these technologies, we must ensure that, despite these constraints, it continues to generate DRC-clean patterns[3]. To this end, we implemented and evaluated it using an industry-standard, Extreme UltraViolet Lithography (EUV)-based 7nm PDK [6]. Specifically, we first captured 1000 patterns from a full-chip design and used them as a proxy for the initial hotspot database. For every pattern in this dataset, we generated 200 synthetic variants using the methodology described in Section II-A. We then subjected them to a full DRC and found that approximately 41.97% were DRC clean. This result demonstrates that the pattern variations (jogs, widths, spaces, etc.) which are carefully introduced by our method lead to legal patterns despite the more complex design constraints. While this percentage is lower than the 96% DRC pass rate of synthetic patterns in $45\,\mathrm{nm}$ technology, it is not a major impediment because synthetic pattern generation and the corresponding lithography simulations are highly parallelizable and, more importantly, one-time procedures. Therefore, with the understanding that slightly higher computational resources may be required due the increased complexity of the fabrication process, the proposed method remains highly applicable to newer technology nodes.

[3]Based on the results reported herein, we conjecture that, given sufficient DRC-clean synthetic variants of known hotspots, the ability of SOTA ML-based hotspot detection methods to learn the root cause is significantly improved in any technology. Regrettably, due to the lack of publicly available lithography models, we cannot apply and evaluate our entire flow in these newer technologies. For the same reason, we can also not evaluate our flow on the popular ICCAD'12 dataset or its recent derivative ICCAD'19 dataset.

IV. CONCLUSION

We proposed a novel database enhancement approach for ML-based hotspot detection and experimentally demonstrated more than 3X reduction in prediction error in comparison to the state-of-the-art.

ACKNOWLEDGMENTS

This research was partially supported by the Semiconductor Research Corporation (SRC) through task 2709.001.

REFERENCES

[1] H. Yao *et al.*, "Efficient process-hotspot detection using range pattern matching," in *IEEE/ACM International Conference on Computer-aided Design (ICCAD)*, 2006, pp. 625–632.
[2] D. Ding *et al.*, "EPIC: Efficient prediction of IC manufacturing hotspots with a unified meta-classification formulation," in *Asia and South Pacific Design Automation Conference (ASPDAC)*, 2012, pp. 263–270.
[3] H. Yang *et al.*, "Imbalance aware lithography hotspot detection: a deep learning approach," in *SPIE Design-Process-Technology Co-optimization for Manufacturability*, vol. 10148, 2017, p. 1014807.
[4] K. Madkour *et al.*, "Hotspot detection using machine learning," in *IEEE Int'l Symposium on Quality Electronic Design*, 2016, pp. 405–409.
[5] Y. Chen *et al.*, "Semi-supervised hotspot detection with self-paced multi-task learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2019.
[6] L. T. Clark *et al.*, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016.
[7] G. R. Reddy *et al.*, "Pattern matching rule ranking through design of experiments and silicon validation," in *ASM International Symposium for Test and Failure Analysis (ISTFA)*, 2018, pp. 443–448.
[8] D. Ding *et al.*, "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.
[9] FreePDK45. https://www.eda.ncsu.edu/wiki/FreePDK. [Online; accessed 1-Nov-2018].
[10] V. Dai *et al.*, "Optimization of complex high-dimensional layout configurations for IC physical designs using graph search, data analytics, and machine learning," in *SPIE Design-Process-Technology Co-optimization for Manufacturability*, vol. 10148, 2017, p. 1014808.
[11] G. R. Reddy *et al.*, "On improving hotspot detection through synthetic pattern-based database enhancement," *arXiv*, vol. 2007.05879, 2020.
[12] G. R. Reddy *et al.*, "Enhanced hotspot detection through synthetic pattern generation and design of experiments," in *IEEE VLSI Test Symposium (VTS)*, 2018, pp. 1–6.