Summer 7-21-2017

Lithography Hotspot Detection

Jea Woo Park
Portland State University

Recommended Citation
Park, Jea Woo, "Lithography Hotspot Detection" (2017). *Dissertations and Theses.* Paper 3781.
https://doi.org/10.15760/etd.5665

Lithography Hotspot Detection

by

Jea Woo Park

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Electrical and Computer Engineering

Dissertation Committee:
Xiaoyu Song, Chair
Douglas V. Hall
Fu Li
Jingke Li

Portland State University
2017

Abstract

The lithography process for chip manufacturing has been playing a critical role in keeping Moor's law alive. Even though the wavelength used for the process is bigger than actual device feature size, which makes it difficult to transfer layout patterns from the mask to wafer, lithographers have developed a various technique such as Resolution Enhancement Techniques (RETs), Multi-patterning, and Optical Proximity Correction (OPC) to overcome the sub-wavelength lithography gap.

However, as feature size in chip design scales down further to a point where manufacturing constraints must be applied to early design phase before generating physical design layout. Design for Manufacturing (DFM) is not optional anymore these days. In terms of the lithography process, circuit designer should consider making their design as litho-friendly as possible.

Lithography hotspot is a place where it is susceptible to have fatal pinching (open circuit) or bridging (short circuit) error due to poor printability of certain patterns in a design layout. To avoid undesirable patterns in layout, it is mandatory to find hotspots in early design stage.

One way to find hotspots is to run lithography simulation on a layout. However, lithography simulation is too computationally expensive for full-chip design. Therefore, there have been suggestions such as pattern matching and machine learning (ML) technique for an alternative and practical hotspot detection method. Pattern matching is fast and accurate. Large hotspot pattern library is utilized to find hotspots. Its drawback is that it can not detect hotspots that are unseen before. On contrast, ML is effective to find previously unseen hotspots, but it may produce false positives.

This research presents a novel geometric pattern matching methodology using edge driven dissected rectangles and litho award machine learning for hotspot detection.

1. Edge Driven Dissected Rectangles (EDDR) based pattern matching

   EDDR pattern matching employs member concept inside a pattern bounding box. Unlike the previous pattern matching, the idea proposed in this thesis uses simple Design Rule Check (DRC) operations to create member rectangles for pattern matching. Our approach shows significant speedup against a state-of-art commercial pattern matching tool as well as other methods. Due to its simple DRC edge operation rules, it is flexible for fuzzy pattern match and partial pattern match, which enable us to check previously unseen hotspots as well as the exact pattern match.

2. Litho-aware Machine Learning

   A new methodology for machine learning (ML)-based hotspot detection harnesses lithography information to build SVM (Support Vector Machine) during its learning process. Unlike the previous research that uses only geometric information or requires a post-OPC (Optical Proximity Correction) mask, our method utilizes detailed optical information but bypasses post-OPC mask by sampling latent image intensity and use those points to train an SVM model. Our lithography-aware machine learning guides learning process using actual lithography information combined with lithography domain knowledge. While the previous works for SVM modeling to identify hotspots have used only geometric related information, which is not directly relevant to the lithographic process, our SVM model was trained with lithographic

information which has a direct impact on causing pinching or bridging hotspots. Furthermore, rather than creating a monolithic SVM trying to cover all hotspot patterns, we utilized lithography domain knowledge and separated hotspot types such as HB(Horizontal Bridging), VB(Vertical Bridging), HP(Horizontal Pinching), and VP(Vertical Pinching) for our SVM model. Out results demonstrated high accuracy and low false alarm, and faster runtime compared with methods that require a post-OPC mask. We also showed the importance of lithography domain knowledge to train ML for hotspot detection.

Dedication

This dissertation is dedicated to my wife, Jeongmin, and my mom for all love and support.

Acknowledgments

First of all, I thank my advisor, Dr. Xiaoyu Song. He has been in full support for pursuing my Ph.D., providing ideas and encouragement. I appreciate all his contribution in time and effort to make my journey to Ph.D. exciting and productive. It was his guidance and excellent scientific advice through the course of this research that enabled me to achieve my Ph.D.

Special thanks go to my co-workers, Robert Todd, Navin Srivastava, Andres Torres, Edmund Pierzchala, and Sam Quinn. Their tremendous help with finding right solutions to the problems I tried to address in my thesis will be deeply appreciated. They also helped me correct my poor English. I am grateful that I work with them.

Most of all, I thank my beloved wife, Jeongmin, for her love, firm support, and giving birth to my daughter, Stella. Without joyous time with my family, I would have given up my pursuit to Ph.D.

I also thank my mother, Wonja Ahn. She is my hero. She has always been the best friend and the strongest supporter for me. Whenever I think of her sacrifice for my education, I feel tears in my heart. Thank you, mom. It was a long journey, and I made it. I love you.

Table of Contents

List of Tables

List of Figures

1

Introduction

1.1 Motivation

Hotspot is a term to describe a location in layout designs where it is prone to have a pinching (open circuit) or bridging (short circuit) error during lithography process for IC (Integrated Chip) manufacturing. In the process, layout patterns composing of a circuit layout are transferred to wafer from a photomask. Figure 1.1 briefly depicts the lithography process. Among those steps, it is the exposure step that is most important to avoid pinching or bridging errors. Figure 1.2 illustrates the exposure tool showing a light source, an illumination, and patterns on photomask to wafer through the projection lens to print a circuitry on a wafer. Note patterns on the mask are different from the original layout. They are modified through Optical Proximity Correction (OPC) to improve fidelity and printability on the wafer.

Since the wavelength of the light source is usually larger than the minimum size of actual patterns on the photomask, geometric shapes on the mask become distorted due to optical proximity effect. This is called sub-wavelength lithography gap in the industry [20]. In other words, what you see on the mask is not what you see on your wafer. In order to tackle this issue and manufacture working ICs, lithographers have invented various Resolution Enhancement Techniques (RETs) such as Optical Proximity Correction (OPC) [12], Off-Axis Illumination [59], Double or Multiple Patterning (DP or MP) [55,57,70,82], Phase-shift mask [62,85], immersion lithography [13,15,65], and sub-resolution assist features [44,78]. (More

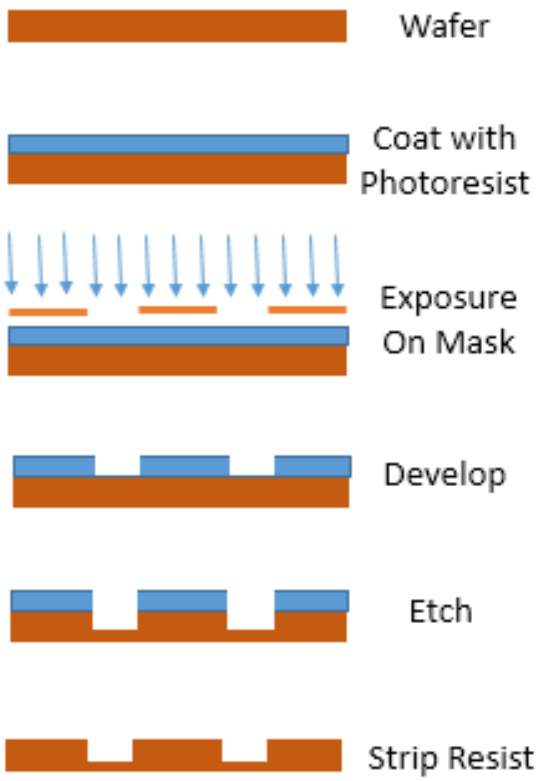
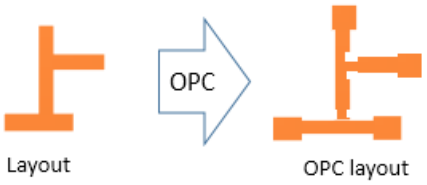


Figure 1.1: Lithography steps

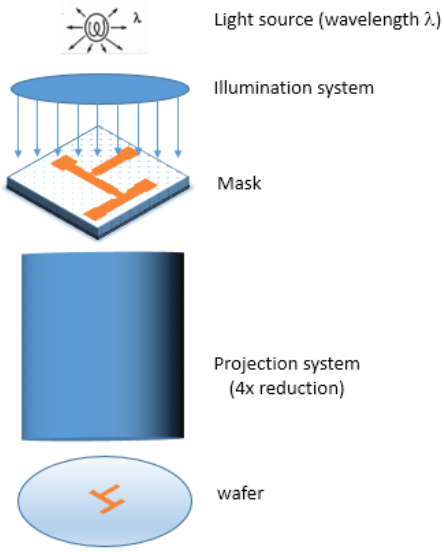
information about RET is at 1.2.)

However, as Moore's law [46] has driven features to smaller dimensions, semiconductor manufacturing process often run into lithography hotspots even with these advanced lithographic techniques. Hotspots are fatal errors that cost a tremendous effort, money, and time if they are found in the lithography process. It is crucial to identify hotspots in an early design stage, ideally at the physical layout generation phase or even at before routing.

In conventional design flow, the industry used to apply lithography simulation on physical layouts [19, 29, 30] to achieve the goal to identify hotspots before manufacturing. This is accurate, but it is computationally expensive at full-chip scale, which limits its practical application.



(a) Optical Proximity Correction (OPC)



(b) Lithography exposure system

Figure 1.2: Optical Proximity Correction and Lithography exposure system

Recently, there have been attempts to avoid lithography simulation for hotspot detection mainly based on pattern matching (PM) and machine learning (ML). The pattern matching-based approach [2,8,11,18,23,63,79,83,86] uses a pattern library that contains a set of known hotspots that are constructed first. Pattern matching then scans through a layout to match the testing layout with the hotspots in the library. This approach is fast and accurate to identify known hotspots. However, it has a limitation to find previously unseen hotspots. In contrast, ML-based hotspot detection [9,43,49,50,53,83] is strong to identify previously unseen hotspots when it is well trained, and the characterization vector is relevant to the problem, but just as pattern matching approaches have their strength and weakness, ML also has limitations. False alarms are inevitable, and therefore, it is critical to create an optimal model and develop methods to reduce the false alarm rate.

This thesis tackles the problems of both pattern matching and ML-based approaches, and it presents a novel methodology and workflow for hotspot

detection. The pattern matching solution presented in this thesis is faster and more accurate than previously developed pattern matching solutions. Besides, it is more flexible in detecting previously unseen hotspots. Our solution for ML-based hotspot detection is unique in terms of its innovative approach to utilizing lithography information and domain knowledge for guiding the ML process and training SVM (Support Vector Machine) models. Unlike other ML-based hotspot detection approaches that use only geometric information or require a post-OPC layout, our solution uses detailed optical information of lithography by sampling latent image intensity to train an SVM model. With this novel idea, we remarkably reduced false alarm rate while achieving high accuracy.

1.2 Resolution Enhancement Technique

1.2.1 Off-axis Illumination

Off-axis illumination [59] is used to enhance resolution limit imposed by Rayleigh equation 1.1 by guiding the zero and the first diffraction light into the projection lens. Figure 1.3 shows illumination types: Conventional and off-axis (Annular, Dipole, Quadrupole)

$$R = \kappa \cdot \frac{\lambda}{NA} \tag{1.1}$$

NA: Numerical Aperture, λ: wavelength, κ: process constant

To transfer patterns on a photomask to a wafer, two diffractions of light, at least, must be captured by the projection lens. In the case of Conventional illumination, the zero order diffraction goes into the projection lens along the axis while the zero order of Off-axis illumination enters off the axis as the name suggests.

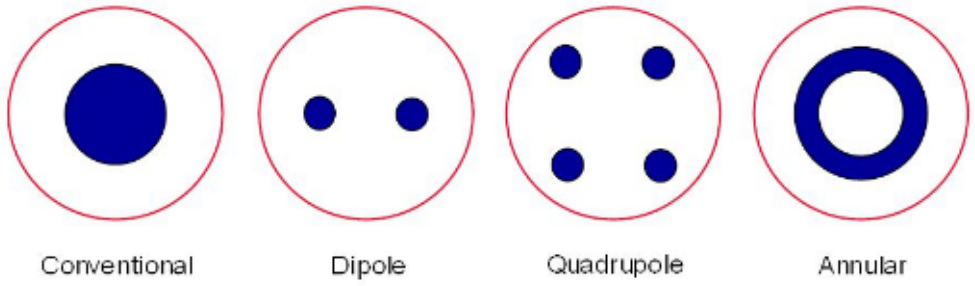


Figure 1.3: Type of illuminations. Dipole, Quadrupole, and Annular are off-axis illuminations.

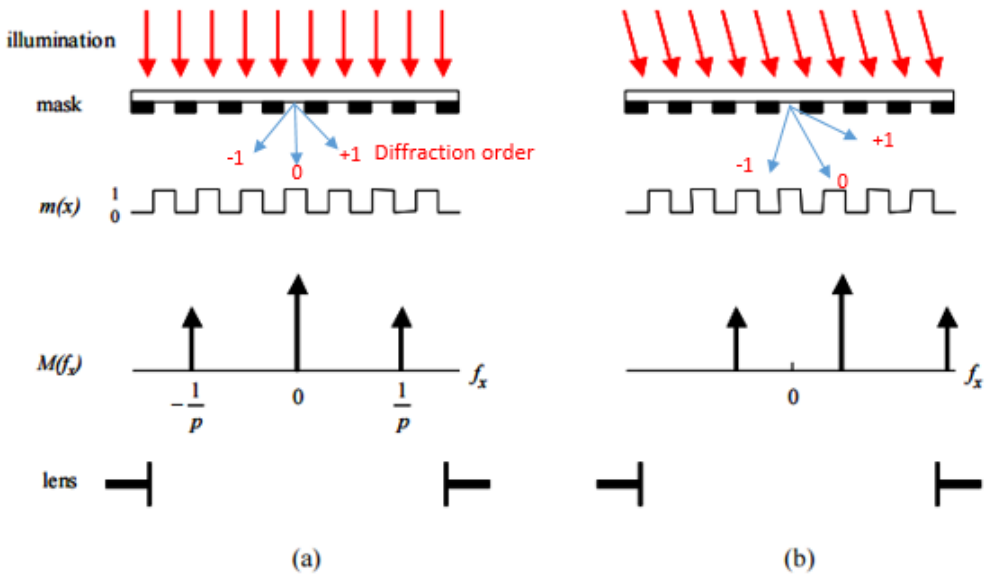


Figure 1.4: Diffraction order entering lens from [39], p is pitch between lines. (a) Conventional; (b) Off-axis;

Off-axis illuminations can form images even when feature size on the photomask is too small, and the first order failed to be captured by the lens in Conventional illumination. As seen at Figure 1.4, Conventional illumination's first order can be missed into the lens when the diffraction angle is significant, which means the feature size on the mask is too small compared to the source light wavelength. However, in the case of Off-axis illumination, one of the two first order can go into the lens to form patterns on the wafer.

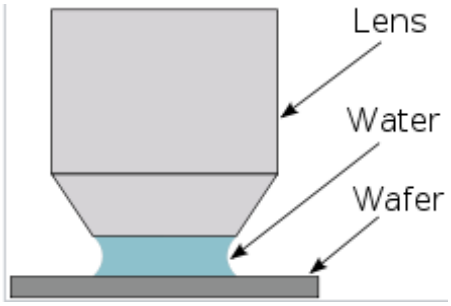


Figure 1.5: Immersion Lithography. Figure from [75]

1.2.2 Immersion Lithography

Immersion lithography [13, 15, 65] uses a liquid as the refractive material between the projection lens and the wafer surface. This fluid's refractive index is greater than one so that it enlarges the numerical aperture which is the size of the maximum refraction angle multiplied by the refractive index. As seen at the equation 1.1, larger NA means better resolution. 1.5 depicts immersion lithography.

1.2.3 Phase Shift Mask

Phase Shift Mask [62, 85] is a photomask that uses a phase shifter to generate phase differences making constructive and destructive interference work together improving image resolution. Figure 1.6 explains how it works.

1.2.4 Sub-resolution Assist Feature

Sub-resolution Assist Feature (SRAF) [44, 78] is unprintable features placed next to printable features on the wafer, usually isolated features, to improve the depth of focus on them. 1.7 illustrates SRAF.

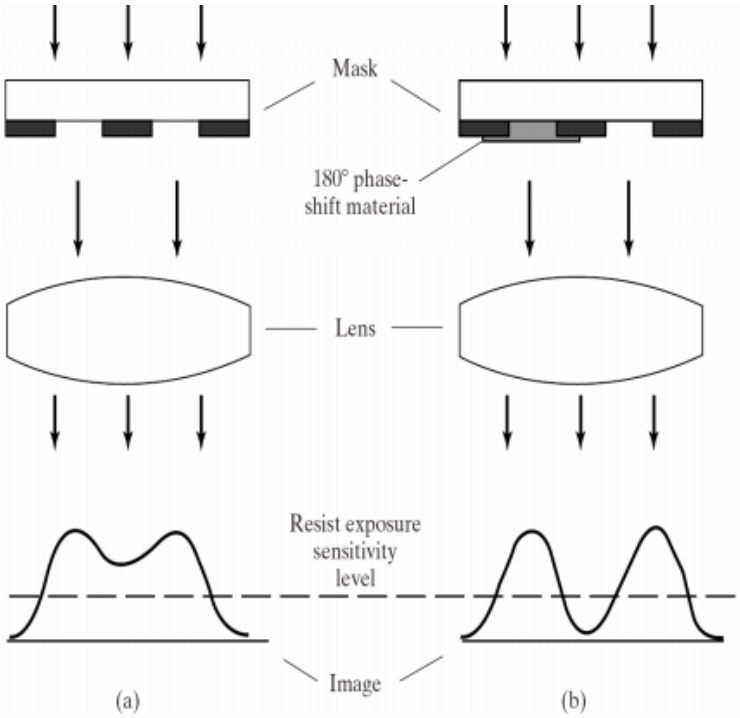


Figure 1.6: (a) Mask without shifter, (b) Phase Shift Mask; figure from [27]

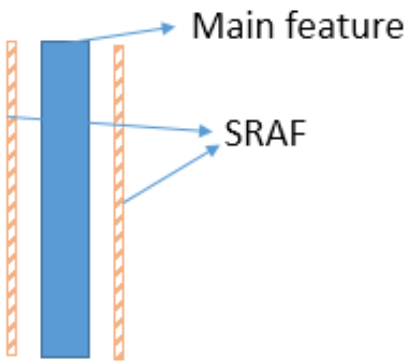


Figure 1.7: Sub-resolution assist feature: SRAF disappears on wafer, improving depth of focus of the main feature.

1.2.5 Multi-patterning Lithography

Multi-patterning [55, 57, 70, 82] is a technique to overcome lithographic limitations dictated by Rayleigh equation 1.1 in chip manufacturing process. Lithography

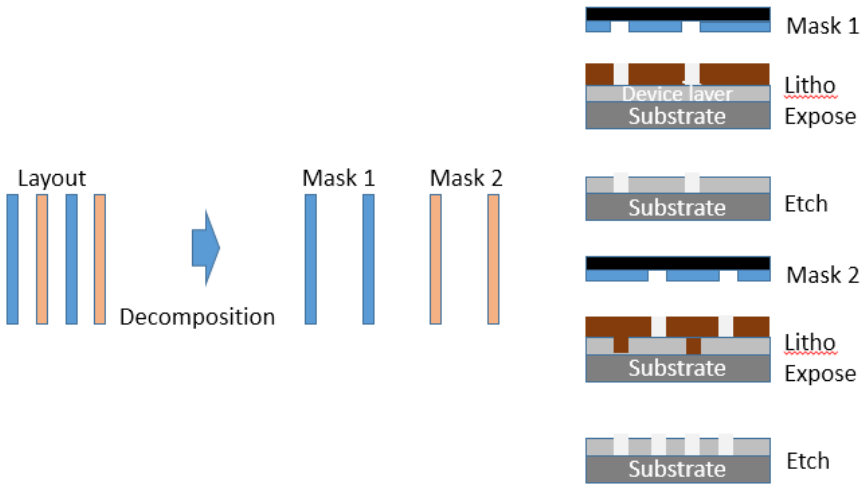


Figure 1.8: Litho-Etch-Litho-Etch Multi-patterning

using ArF of 193nm wavelength reached its physical limit of 40 nm half-pitch. Multi-patterning technique allows chip manufacturing companies to image IC designs of 20 nm and below.

There are two mainstream approaches of multi-patterning: LELE (Litho-Etch-Litho-Etch) and SADP (Self-aligned double patterning). For LELE, a physical layout is decomposed to two layouts, and Litho-Etch is done for each one which is combined to define a single layer. Figure 1.8 shows LELE. For SADP, a spacer is formed along the lines, and the lines are removed leaving only spacers along the lines. Then, those spacers are used to form final lines, which means the number of lines double the original lines. Figure 1.9 depicts this process.

1.3 Problem Definition

1.3.1 Pattern Matching

Current pattern matching methods lacks flexibility in identifying a hotspot that is not registered in a database, namely hotspot pattern library. They are fast

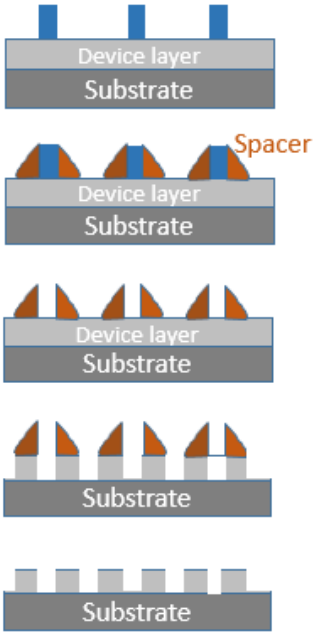


Figure 1.9: Self-algined Double Patterning

and accurate to detect hotspots only when those are in the library. In other words, pattern matching is working perfect for an exact match, but it has limited flexibility to describe a hotspot pattern in a fuzzy way (Relaxing the description such that it can detect not only exactly matching hotspots but also fuzzy matching for previously unseen hotspots). There have been several fuzzy pattern matching ideas introduced to attack this issue. Hybrid approaches [40, 42, 51, 83] have been tried, where they tried to use a combination of pattern matching and ML-based hotspot detection. But, as mentioned in [64], such hybrid models are 10 to 100 times slower than pattern matching. They also do not fully address the false alarm issue inherent with a ML-based approaches.

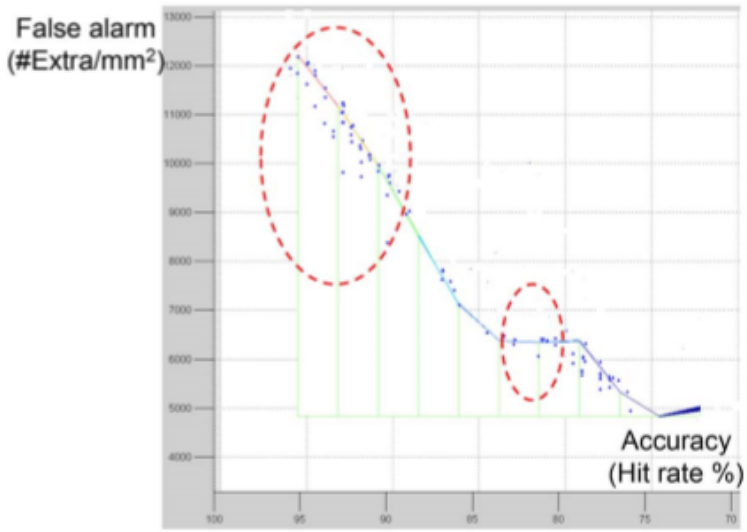


Figure 1.10: Fig.15 of [9] Exponential increase of false alarm

1.3.2 Machine Learning

Machine learning is a subject being studied in computer science to give computers ability to learn without being explicitly programmed such that it can predict or change a behavior of the program when exposed to new data [38]. ML-based approaches for hotspot detection have typically used a supervised learning model, e.g., artificial neural network (ANN) [50] or support vector model (SVM) [43]. Recent research involves hierarchical learning [49], data clustering [53], fuzzy cluster growing [83], and topological classification and critical feature extraction [9]. All of these ML-based hotspot detection methods suffer from false positives when they try to achieve higher accuracy of real hotspot detection. Figure 1.10 of [9] shows their experimental data to point out the tradeoff between accuracy and false alarms. Even though [9]'s critical feature extraction method is one of best ML-based approaches, they were not able to adequately address the false alarm issue. Their experimental data showed that it is exponentially difficult to suppress false

alarms as accuracy levels go up.

1.4 Proposed Approach

1.4.1 Edge Driven Dissected Rectangle-based Pattern Matching

Our solution to the lack of flexibility in pattern matching is to create a framework where a simple and flexible pattern description is adopted for fuzzy pattern matching. This framework is not only flexible enough to detect previously unseen hotspots but also accurate enough to identify exact patterns in the pattern library. Among many pattern matching methods [2, 8, 11, 18, 23, 79], our approach focuses on DRC (Design Rule Check) [76] based pattern matching [11, 18, 23] because it has shown better accuracy and faster runtime. However, unlike other DRC-based pattern matching algorithms, we propose novel algorithms for pattern matching which dissects patterns into rectangles based on polygon edges. We call this solution EDDR PM (Edge Driven Dissected Rectangle Based Pattern Match). Our solution utilizes simple DRC edge length rules and DRC width/space check rules to create rectangles (Figure 1.11) for hotspot pattern descriptions. Formal definitions of those DRC operations are defined in Appendix A. With the idea of EDDR PM we also solve the problem with DRC-based pattern matching needing such a high number of DRC rules when describing complicated hotspot patterns. As shown at Figure 1.12, this description is a set of member rectangles created by edge length, width, or space between edges of a hotspot pattern. Note that the yellow box in the figure is a bounding box of the pattern. Along with this bounding box, each member rectangle's center point can create a vector space 3.8 which will be explained in detail in Section 3.1. The flexibility we add to our pattern matching solution comes from the fact that we can skip a certain member

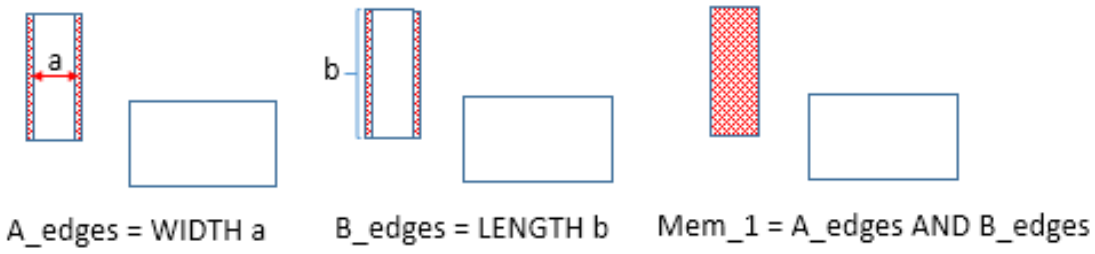
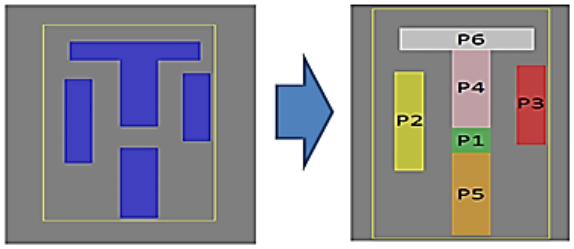


Figure 1.11: DRC length rule check, width rule check, and rectangle creation



P1	P2	P3
Len_1 == 0.15	Len_1 == 0.605	Len_1 == 0.485
Len_2 == 0.15	Len_2 == 0.605	Len_2 == 0.485
Space == 0.16	Width == 0.11	Width == 0.11

P4	P5	P6
Len_1 == 0.48	Len_1 == 0.505	Len_1 == 0.13
Len_2 == 0.48	Len_2 == 0.505	Len_2 == 0.13
Width == 0.15	Width == 0.15	Width == 0.535

Figure 1.12: Pattern and dissected rectangles as member

rectangle creation in the pattern description. For example, if we don't create P3 in Figure 1.12 and only care about the other members, we can detect not only exact matching hotspots but also fuzzy matching ones as well. Previous work for DRC-based pattern matching solutions such as [10] have tried the concept of "don't care region" to do the fuzzy matching for previously unseen hotspot detection. However, the process to take care of "don't care region" is complex and requires a lot of extra work. Our solution does not have this issue because "don't care region" is naturally embedded in our EDDR PM.

This approach has at least three advantages over other solutions. First, it is

faster than other state-of-the-art pattern matching tools. Second, it is intuitive and straightforward for pattern matching engineers to understand and describe patterns. Third, it scales well for parallel computation. We also show how to improve pattern matching runtime using vector space created by an origin rectangle and other reference rectangles inside a pattern bounding box. By adopting the vector concept, we iterate only once or twice when detecting different pattern orientations. Other pattern matching techniques usually iterate eight times (4 rotations x 2 mirrored images) 3.10 to detect all of the eight different orientations. Our method eliminates these unnecessary iterations.

1.4.2 Litho-aware Machine Learning for Hotspot Detection

Previously proposed ML-based hotspot detection methods tried to use only geometric information or required a pose-OPC layout. These methods fail to solve the false alarm issue while maintaining high accuracy. As a fundamentally different approach, we propose to use lithography information and lithographic related domain knowledge for machine learning.

As explained at [14], prior knowledge (Domain knowledge) plays a crucial role to have machine learning trained as accurate as possible. In addition to training data, we have to select features which are paricularly informative to the training. In this thesis, we use aerial image intensity information produced by the same illumination as used in the chip manufacturing process. We also select features based on hotspot type such as bridging and pinching. Furthermore, if the illumination is asymmetric, those two hotspot types split into four types: Horizontal bridging, vertical bridging, horizontal pinching, and vertical pinching. Therefore, we create four SVM kernels that are trained with aerial image intensity information. Each kernel will help us

to find hotspots in the design. It is important to note that the simulation of these intensity points is obtained by using the drawn (design target) structures, thus eliminating the need of a post-opc mask.

1.5 Contributions

1.5.1 Pattern Matching solution

The key contributions of our work on EDDR PM are summarized as follows.

1. We developed a fast pattern matching method that shows significant speedups against state-of-art commercial pattern matching tools and other methods.

2. We presented an easiness and flexibility of our method for fuzzy pattern match and partial pattern matching.

3. We introduced a practical idea of adopting vector space concept to reduce the number of iterations for the pattern matching.

1.5.2 Machine Learning solution

Our test experiment result against 2012 CAD contest at ICCAD [64] shows almost 100% accuracy and low false alarm rate. In fact, it shows our approach outperforms all other previous works by a significant margin as shown at Table 4.2. Our contributions are briefly summarized as follows:

1. We proposed a robust and accurate SVM kernel training method utilizing real lithography illumination information and domain knowledge of lithographic failure types.

2. We presented experiment results demonstrating our method is able to achieve high accuracy with low false alarm rates, which means it overcomes the biggest drawback of ML-based hotspot detection methods.

I have published the following papers:

1. Journal

   (a) Jea Park, Robert Todd, Xiaoyu. Song, *Geometric Pattern Match Using Edge Driven Dissected Rectangles and Vector Space* in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2016, Vol: 35, Issue: 12, pp: 2046-2055, DOI: 10.1109/TCAD.2016.2535908

   (b) Jea Park, Andres Torres, Xiaoyu Song, *Litho-Aware Machine Learning for Hotspot Detection* in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), 2017 (Note: Revision decision on May 10, 2017. Submitted first revision on May 22, 2017)

2. Conference and Poster sessions

   Jea Park, Robert Todd, Xiaoyu Song, *Geometric Pattern Match Using Edge Driven Dissected Rectangles and Vector Space*, Poster presentation in Work-in-Progress session at IEEE Design Automation Conference (DAC), June 2014.

1.6 Organization of the Dissertation

The rest of the dissertation is organized in several chapters.

Chapter 2 reviews related work on hotspot pattern detection.

Chapter 3 describes the fundamentals of EDDR PM such as DRC operations, member rectangle creation, and vector space. We also show how EDDR PM works

for exact pattern matching as well as fuzzy pattern matching. Experimental result of EDDR PM against previous works is provided in this chapter.

Chapter 4 explains our idea of ML-based hotspot detection. This chapter describes how lithography information is incorporated into SVM models in detail. Experimental result of our Litho-aware Machine Learning based hotspot detection against previous works is provided in this chapter.

Chapter 5 concludes this thesis by summarizing key results and contributions.

Chapter 6 discusses future works related to this thesis.

2

Related Work

As hotspot detection has become a hot topic in the industry, many researchers have been trying to come up with fast and accurate solutions [2,3,8–11,18,23,25, 28,40,42,43,49–51,79,83,86]. For example, they have proposed ML-based hotspot detection methods [9, 43, 48–50, 71], hybrid methods using both ML and pattern matching technologies [3, 40, 42, 51, 83], explicit model-based methods [28, 79, 86], string-based pattern matching [2,25], and DRC-based pattern matching [10,11,18, 23].

1. Machine learning-based detection:

   In this approach, hotspot patterns are extracted for training an artificial neural network or SVM model. This model is then used to predict potential hotspots in layout designs. Therefore, it is essential to create an accurate learning model to avoid false alarms. It requires a long training time and other complex techniques to reduce false alarms. In fact, false alarms are inevitable in this approach even though the false alarm rates may be relatively low. Our litho-aware machine learning based hotspot detection method address this issue.

2. DRC-based pattern matching:

   This method uses DRC (Design Rule Check) rules to identify patterns. First, the input pattern is converted into some DRC rules, whose output is then analyzed to obtain pattern matches. This approach does not suffer

from the false alarms, therefore it can be used for the exact match. However, DRC tends to generate a large number of complex DRC rules causing high computational cost as the numbers of patterns are increased. Our EDDR PM proposal addresses this shortcoming.

3. Explicit model-based pattern matching:

   In this approach, an explicit model is created to compare target patterns to patterns in the layout. For example, Kahng [79] proposed to create a dual graph to represent patterns and to use it for filtering out all non-matching patterns. But, this approach also suffers from false alarms due to its inherent modeling error. Compared to Machine-learning-based pattern matching, it is more accurate and more efficient.

4. String-based pattern matching:

   It applies string matching techniques to pattern matching. Frist, a grid is created. Each grid point is converted to a layout matrix where 1 is assigned when it overlaps with a geometry, otherwise 0. Then, points are encoded into strings for pattern matching. This method does not suffer from false alarms, but it is not suitable for cutting-edge designs because manufacturing grid sizes are getting smaller and it increases computation time exponentially.

5. Hybrid detection:

   The idea of hybrid hotspot detection is to combine machine learning and pattern matching together for hotspot detection. The combination of two complement each other trying to minimize their weakness and boost their strength for better hotspot detection result. This approach, however, as

reported in [64], is ten time slowerthan pattern matching approaches and still shows high false alarm rates.

2.1 Machine Learning-based detection

1. Artificial Neural Network using bitmap of litho simulation contour

   Nagase [48] adopted Artificial Neural Network (ANN) [32] which is trying to mimic a human brain for learning. ANN can approximate unknown functions with a large number of inputs. It creates a neural network which has highly interconnected neurons (processing elements) that process information and passes it to the flow of information inside the network. A link between neurons is associated with weight. ANN learns by altering the weight values through test samples. If the network generated undesired output from the samples, it alters the weights. The authors trained their ANN using bitmaps from lithography simulation contour images on post-OPC patterns. Their success rate to find hotspots in testing was about 42% to 90%. They tried only four types of patterns, which is far fewer types than real hotspot patterns in reality.

2. ANN using critical hotspot features

   Ding [50] reported that their method to extract critical hotspot features and use them for ANN training was more accurate and faster than 2D pixel image-based models. They proposed three major features such as Bounded Rectangle, T-shape metal, and L-shape metal shown at Figure 2.1. The number of features is dramatically smaller than training on 2D pixel image such as bitmap, which is a major factor for their improved runtime. Their

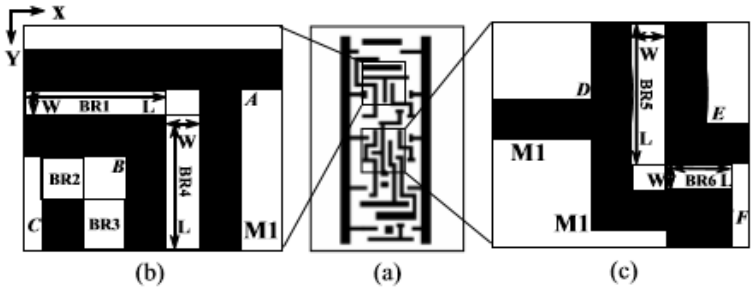


Figure 2.1: critical features. Figure from [50] (a) certain 45nm cell layout; (b)(c) Two sampled pattern examples for critical feature extraction procedure; Each BR is expressed with a 5 parameter vector (W, L, X, Y, D), where L denotes the length of BR along the metal edges containing itself; W denotes the width of BR along the direction perpendicular to L; (X, Y) is the coordinates of the upper-left corner of BR; D is set to 0 if W is along X direction, to 1 if W is along Y direction. Area A is T-shaped metal for BR1/BR4, area B is L-shaped metal for BR1/BR2/BR3/BR4, area C is neither T-shape nor L-shape for BR2/BR3.

accuracy ranged from 80% to 90% with a 10% false alarm rate.

3. SVM using 2D distance transform and image histogram

   Drmanac [71] tried to build SVM models using 2D distance transform and histogram extraction on pixelized layout images. They first do raster scanning on a layout and produce the portable bitmap (PBM). Then, they transform the image to a grayscale format named portable gray map (PGM) where each pixel is now an integer from 0 to 255 representing gray scale level. They used a distance transform technique for this transform which is widely being used in image processing shown at Figure 2.2. With this, they create an image histogram in a raster window, which has 256 bins on the x-axis and display number of pixels per bin on the y-axis. A simple example of this process is shown at Figure 2.3 This image histogram information is used for training their SVM which computes the similarity between image histograms. Their result showed it achieved about 90% accuracy which is

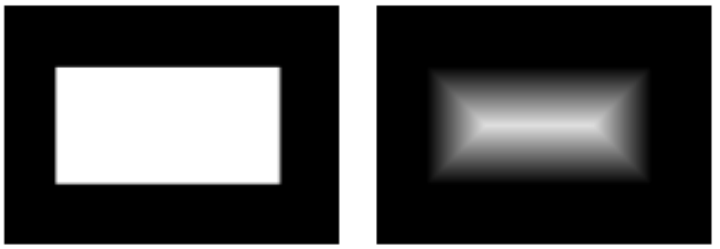


Figure 2.2: Distance transform from PBM to PGM of [71] (a) Portable bitmap (PBM); (b) Portable gray map (PGM)

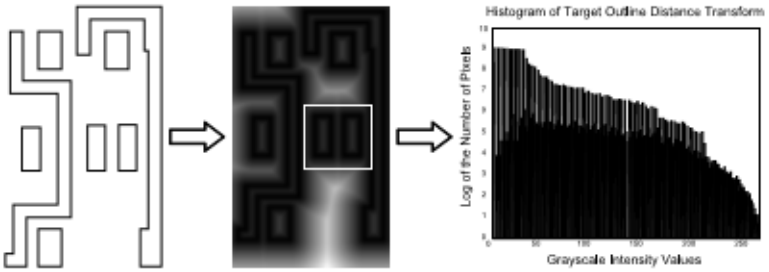


Figure 2.3: Image histogram creation process. Figure from [71].

not satisfactory for industry standard. It also showed their runtime is faster than direct lithography simulation in finding hotspots (variability prediction in their paper). However, it is still too slow for full chip level application and it is not surprising if you consider a huge number of pixels they have to deal with.

4. Multi-level method

Ding and Torres [49] proposed a fragment-based classification feature metrics. Fragmentation is a process to break edges of geometry into smaller pieces for OPC. The authors defined hotspot signature based on fragment information, which is shown at Figure 2.4, such as convex corner, concave corner, external distance between fragments, internal distance

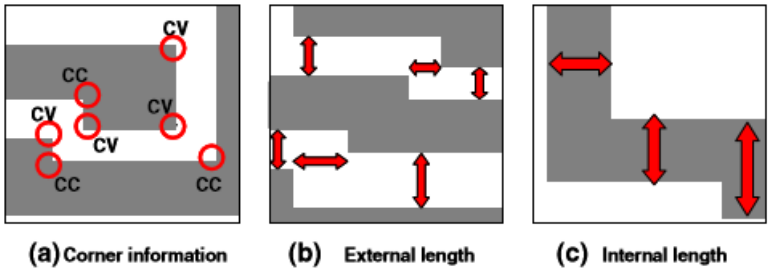


Figure 2.4: [49]'s features to train their hotspot detection model. Figure from [49].

between fragments, and etc. (NOTE: External distance is the distance when fragments are located on different polygon. Internal distance is when fragments are located on the same polygon.) By performing this step, their feature-centric layout characterization avoids expensive operations for characterization of hotspots such as 2D distance transform and density extraction. They fed this data to train SVM or ANN in a multi-level manner where they create ANN or SVM at each level with a different threshold to detect hotspots until the false alarm rate is under their target. They called this flow "hierarchically refined machine learning in multi-level". The main reason of multi-level machine learning was to lower false alarms. Their experimental result showed 89% accuracy with false alarm rate ranging from 130 to 7,500 per mm^2. (Note: False alarm rate is measured by false alarm count per mm^2)

5. Layout density-based feature metric with two-level SVM

   Wuu [43] extended their previous work [41] which used layout density for pattern representation to encode features for SVM training. They chopped a layout in pixels of 35 by 35 nm to generated density information per pixel. This density-based pattern representation is shown at Figure 2.5. Along with this density information for SVM models, they proposed a two-level

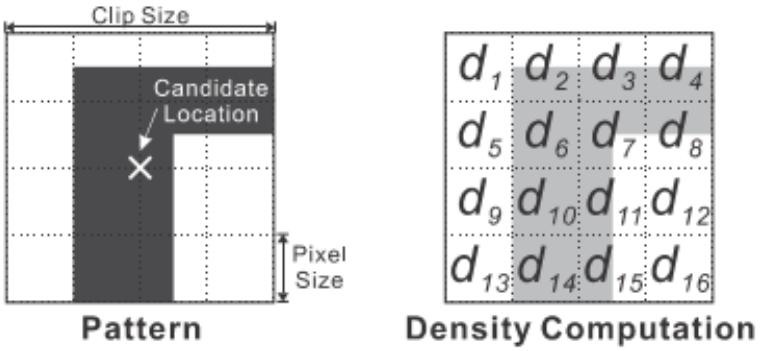


Figure 2.5: Density-based pattern representation. Figure from [43].

approach to lower false alarm rates. At the first level, they trained an SVM using hotspot and non-hotspot samples. Then, the first classifier runs on non-hotspot samples to gather the samples that were wrongly predicted as hotspots. SVM at their second stage, which becomes the second classifier, is trained using hotspot samples and those samples that were produced as hotspots on the first stage. It is a similar attempt with [49] to reduce false alarms. They also tried to use small sample pattern clips for their level-1 classifier to filter out the majority of non-hotspots while training data for the level-2 classifier was larger including the peripheral pattern density information. This method yielded accuracy 84% on average with a reasonable false alarm rate. But their testing layout is only about 700 by 700 um^2. It may be worse for full chip hotspot detection.

6. SVM using critical features extracted from topologically classified samples

   Yu's approach [9] to build an SVM is made of two steps. At the first step, they classify training samples into clusters based on similarity of topologies of their core regions. They introduced two-level topological classification. String-based classification [2] was applied first and then

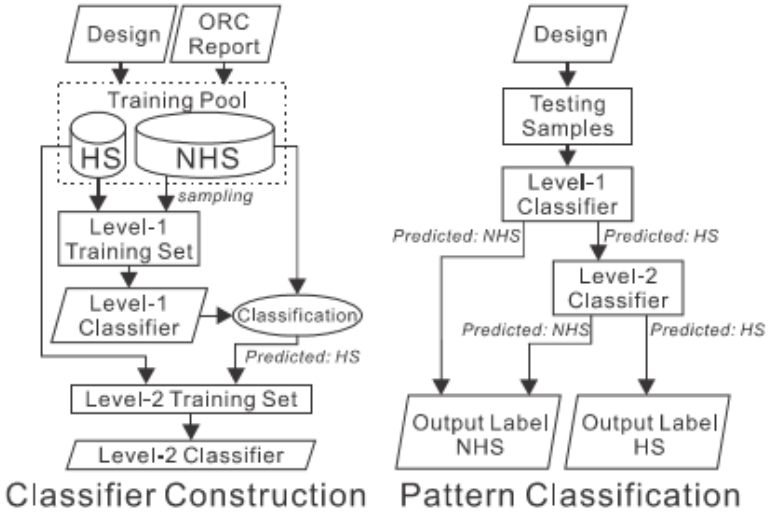


Figure 2.6: [43]'s two-level approach for training model and testing flow. Figure from [43].

density-based classification, which is the same way as layout density-based feature metric was done. The output from the first string-base classification is used to refine the clusters. Figure 2.7 shows an example of this two-level topological classification. Moreover, they defined critical features using their previous work [11] to train their SVM. They extracted critical feature information from each cluster that was fed to an SVM for training so that they could have multiple SVMs. They also created a feedback SVM to suppress false alarms. Their method shows about 90% accuracy with a relatively small false alarm rate.

2.2 Drc-based pattern match

1. Topological graph

   Pikus [18] constructed all lengths of geometry edges and distances between polygons in a pattern to create a graph based on this information.

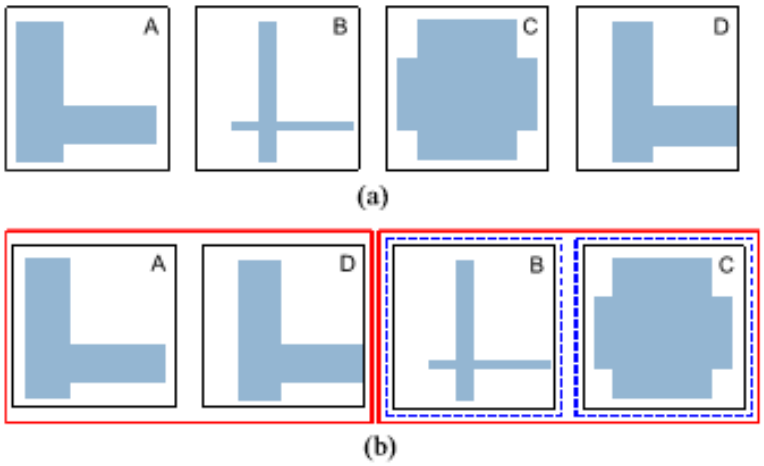


Figure 2.7: Two-level topological classification. (a) four core regions of hotspots, (b) {A,D} and {B,C} classification from the string-based classification. {A,D}, {B}, and {C} final classification from density-based classification. Figure from [9].

During matching phase, they created a search graph using the topological information created by DRC on a layout. Therefore, matching is a process to find the topological graph of hotspot patterns inside the search graph. This approach suffers from slow runtime because their topological representation of a pattern is not an efficient and compact representation. If they have a complex pattern to describe, they need more rules and their DRC rules explods. This approach is also not capable of detecting previously unseen hotspots. Figure 2.8 shows a pattern description example. As seen in this example, the pattern information is represented by DRC topological operations such as edge lengths and distance between edges. The distance between two edges can have a range value of X, W, and Z polygon case, while the V and Y polygons would still be the same distance away from W.

2. Hash table of corner or edge recorded by DRC

   Gennari [23] proposed hashing technique to speed up DRC-based pattern

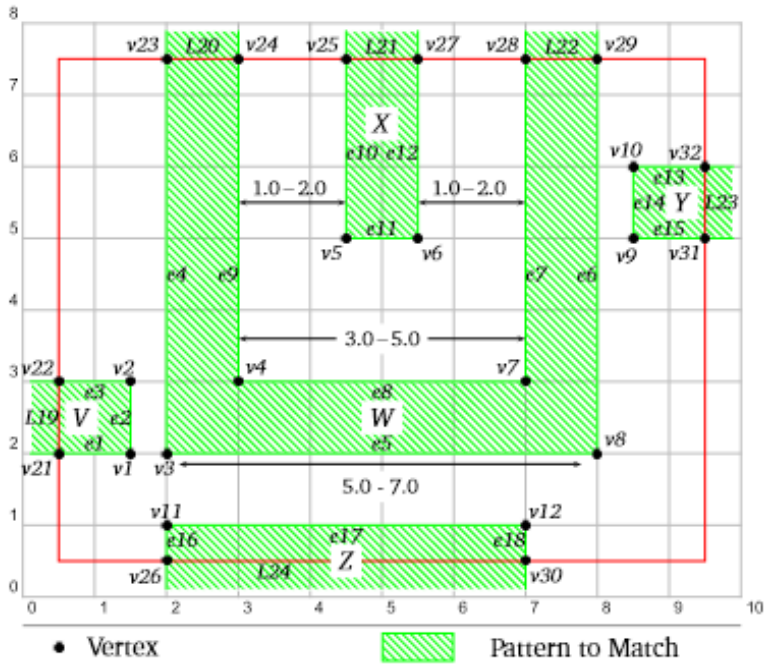


Figure 2.8: Example pattern description of [18]

matching. They identify corners or edges in a pattern by DRC and create hash values based on information around the corners or edges to represent pattern configuration. This hash table is used to match patterns in a layout where the DRC engine reports corners or edges of polygons in the layout and compute hash values. If the hash value is the same in the hash table, it proclaims matched. This idea needs a sophisticated hash function to avoid hash collisions. More hash collision means more matching time required. This hashing idea for pattern match cannot handle previously unseen hotspot as well like above topological graph approach.

3. Critical design rule extraction

   Yu [11] tried to extract critical feature design rules that are most relevant to hotspot descriptions. Rather than defining all the rules that are

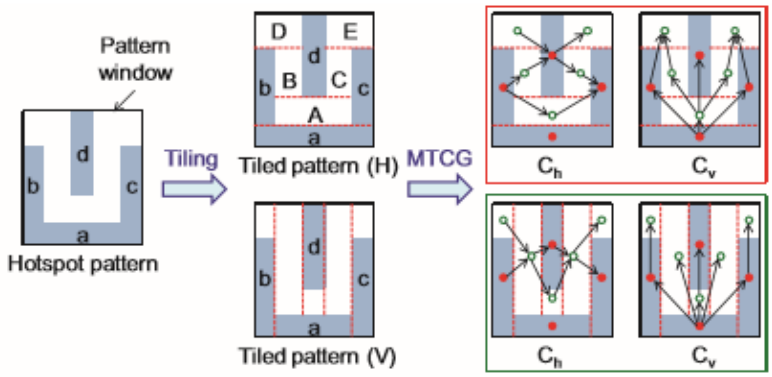


Figure 2.9: Modified Transitive Closuer Graph of [11]

necessary to describe a pattern, they reduce the number of rules to five most common hotspot rules. To do that, they adopted TCG (Transitive Closure Graph) from Lin [4] and modified it as MTCG (Modified TCG) shown as Figure 2.9 to describe pattern's topology. They first construct MTCG for a hotspot pattern and extract critical design rules from it. They perform these critical rules on a layout to create MTCGs and compare hotspot MTCG to the created MTCGs for an exact match. They claimed their approach outperformed other DRC-based pattern matching methods such as [18, 23]. They extended their idea for fuzzy pattern match by adding "don't care region" concept to their original MTCG which is shown by their next paper [10]. Since [10] has some capability for the fuzzy pattern match, it is somewhat possible to identify previously unseen hotspots. But, it works only in a limited way and not sufficiently flexible because polygons inside a pattern are described as exact match and they have some freedom in only "don't care region" areas during the matching process.

2.3 String-based pattern match

1. String-based pattern match

Yao [2] described an idea to use string matching for the pattern matching. They divided a pattern into rectangles with additional specifications encoded by strings, which they called it a "range pattern" since each rectangle can have range value. Range values include width, length, space, optimal width, optimal length, and optimal space range value. They even allow linear combinations of those range values to be encoded in the rectangles. Therefore, their range pattern can be used as fuzzy pattern matching tool to handle previously unseen hotspots. Regarding some detail about their approach, they divide a layout into 2D pixels to represent them as a 2D matrix. As shown at Figure 2.10, if a rectangle overlaps a grid location, it is 1. Otherwise 0. For a simple exact match, meaning there is no range value encoded in the range pattern, the matrix comparison is sufficient to final exact match. For fuzzy matching applications, the matrix representation of all possible hotspot patterns that can be generated by a general range pattern is too big and it is too computationally expensive for matching. So, they proposed new representation called cutting-slice representation which slices range patterns in regions and put constraints derived from the range pattern value on the regions. This method is accurate and flexible to handle previously unseen hotspots. But the process is complicated resulting in slow performance, and it suffers more when a smaller pixels are required as design nodes are getting smaller and smaller.

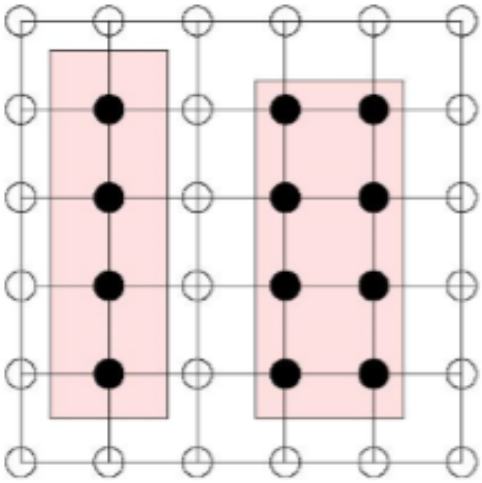


Figure 2.10: Layout representation as layout matrix. Figure from [2]

2.4 Hybrid detection

1. Hybrid flow using hierarchical clustering and pattern matching

   Ma [3] took a hybrid approach combining unsupervised machine learning for clustering and pattern matching. The hierarchical clustering algorithm was adopted to classify hotspots into clusters. This is because we don't know how many clusters are in the training hotspot data set before clustering. Hierarchical clustering allows choosing the number of clusters (k) after its training. They picked the best k using C-index [60] and Point-biserial correlation coefficient [45]. Their distance metric for distinguishing clusters is based on geometric similarity as described with the equation of 2.1. Figure 2.11 is an example of the equation.

$$\rho(\theta_1, \theta_2) = \left[\iint_{\theta_1 \neq \theta_2} dA \right]^{\frac{1}{2}} \tag{2.1}$$

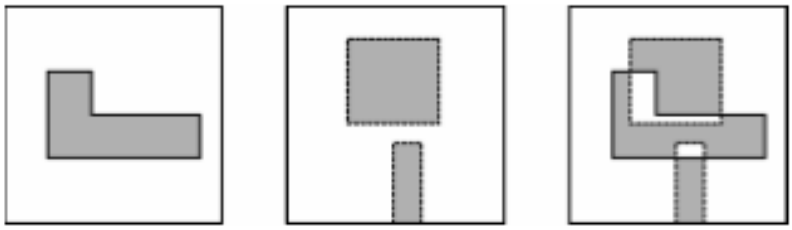


Figure 2.11: Visualization of the distance metric of [3]. Two patterns and, on the right, the area where the two differs. Figure from [3]

Their flow is as follows: First, they gather all the hotspots on a training layout from a lithographic simulation. And then, using the distance metric, they perform the hierarchical clustering algorithm. They choose appropriate k to generate clusters. Once the hotspot clusters are created, each cluster is analyzed to produce a representative pattern of the cluster which is then fed into pattern matching tool to identify hotspots on testing layouts. Because hotspots in one cluster share similar geometric shapes, they claimed that all hotspots in the same cluster may be fixed by a common fixing solution, which is mentioned as their "future work". They did not carry out the pattern matching part they described in their paper so that we don't know the accuracy and false alarm rate of their approach. The complexity of training time is O(n$\hat{2}$). Table 1 of their paper showed it took about 30 minutes for training with 13,200 hotspot samples. Considering it is common in the industry to deal with millions of hotspots, this runtime is not acceptable.

2. Flow to combine SVM and Pattern matching

   [42] tried to use both SVM and pattern match to improve accuracy and lower false alarm rates. The authors used their previous work [43] for training SVM which is two-level density-based machine learning [43]. They used

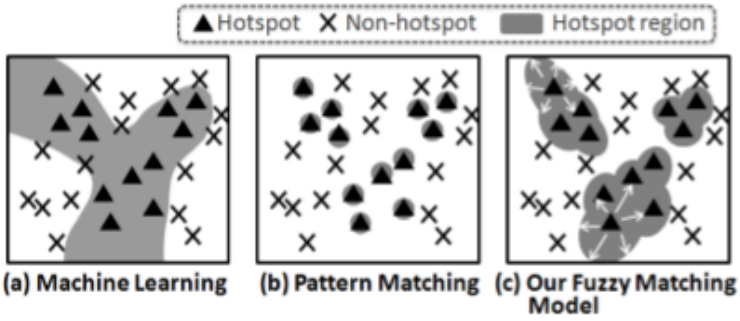


Figure 2.12: 2D-space example of hotspot region decision. Figure from [83]

commercial industry pattern matching tools in their hybrid hotspot detection flow. The motivation for their hybrid flow is simple. Pattern matching tool can perform exact match resulting in no miss on known hotspots while machine learning can miss some known hotspots, but can identify previously unseen hotspots. Therefore, Combining these two methods in a hybrid flow may show some benefit. However, their result was not impressive. It was because their machine learning model still produced high false alarm rates. Their model was based on a density-based encoding method which did not overcome the issue presented in their previous work [43].

3. Fuzzy pattern match

   Lin [83] presented a fuzzy matching model which is constructed by density-based SVM [43], hotspot grouping, and fuzzy region growing process which is illustrated in Figure (c) of 2.12. The fuzzy region of a hotpot is determined by the fuzzy distance which is calculated by expanding a hotspot point until it reaches non-hotspot points. The group distance, fuzzy distance, and fuzzy region are trained for their fuzzy model. They generate hotspot candidates in a similar way of pattern matching with a hash table [23]. They first select representative polygons in a hotspot pattern. The represented

polygons are defined as the polygons with the most vertices which are close to the center of the hotspot. These representation are used to create a hash pattern library. Then, during the testing stage of their model on a layout, they scan their testing layout to find the represented polygon of each known hotspot. If they find it, they apply their fuzzy model to decide whether it is a hotspot or not. Their experimental result showed 72.41% accuracy with the false alarm of 1,907 per mm^2.

3

Geometric Pattern Match Using Edge Driven Dissected Rectangles

3.1 Change from Traditional Design Rules to Pattern Match

There was a shift of physical verification paradigm at 45 nm process node and below, which was propelled by design complexity and manufacturing issues, particularly lithography hotspots. [61] explains well this change as follows.

"Human beings are visual people. From the earliest moments of our life, visual patterns are the dominant way we learn about our world. Throughout our lifespan, we react more strongly to visual stimuli than any other. Even when we speak different languages, we can communicate basic ideas via pictographs with perfect understanding.

IC layouts are visual in nature - any engineer who looks at a layout can instantly recognize transistors and wires and vias - yet we have always defined them with an esoteric textual scripting language. We define layout features by describing in text how wide and tall and long they are. We enhance these definitions by specifying the distances allowed (or not allowed) between features. This text-based, one-dimensional approach worked well enough for a fairly long time, but words have finally begun to fail us.

At today's nanometer nodes, especially at 45 nm and below, we're no longer defining relatively simple, one-dimensional length and width types of measurements. Lithography and manufacturing limitations combined with performance requirements expand the radius of influence within a design layout so that we now find ourselves trying to describe an increasing set of combined features that

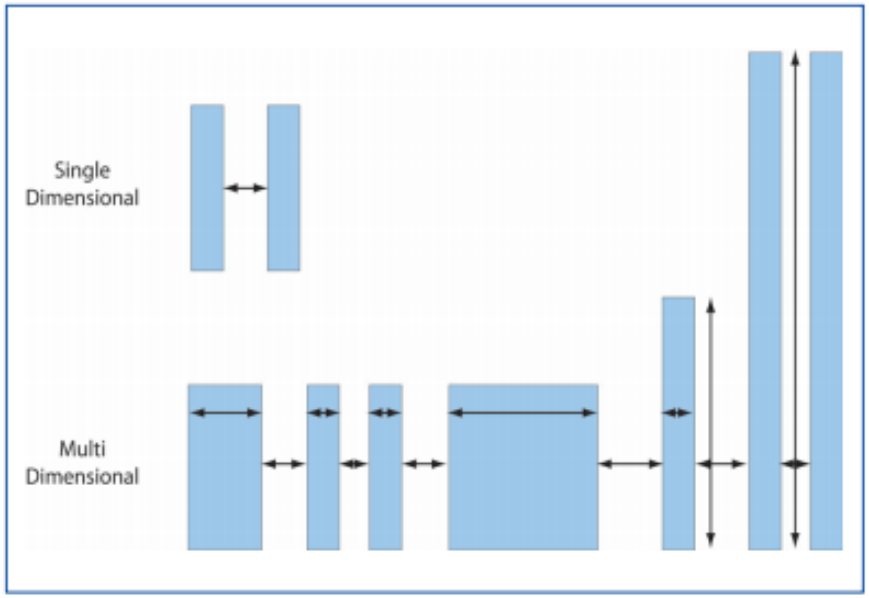


Figure 3.1: Design constraints and influences have spread far beyond simple length-/width measurements at 45 nm and below. Figure from [61]

are all interdependent, and sometimes multi-dimensional. Some configurations are so complex that they simply cannot be accurately (or practically) described with existing scripting languages.

Figure 3.1 illustrates how the focus of design rules has changed from a simple length-width type of measurement to a complex, interdependent, multidimensional set of variables. Not only are there more measurements in the multi-dimensional case, but all the measurements are interdependent, so the allowable range of any particular dimension depends on the values of many surrounding measurements.

Lithography presents a different set of challenges. Even in the late 1990s, feature sizes were smaller than the wavelength of light commonly used in lithography, and the gap has been growing steadily ever since. Achieving resolution at 45 nm and below has become a challenging puzzle, where systematic variability is heavily impacted by both the wafer manufacturing processes and

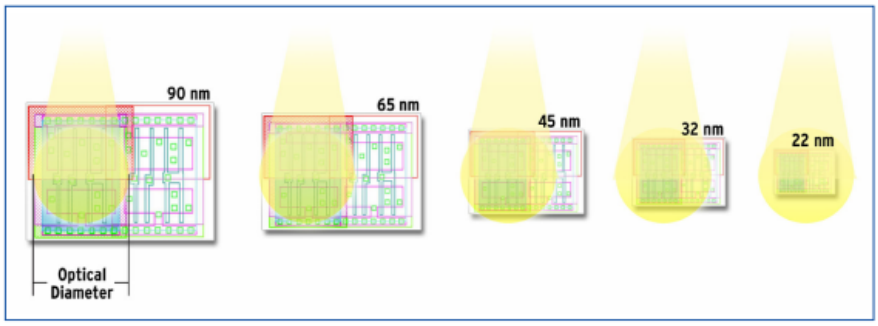


Figure 3.2: Feature size has been continuously shrinking node over node. At 22 nm, an entire IC standard cell design may be smaller than the optical diameter. Figure from [61]

the topological layout features themselves. As geometries shrink relative to the illumination source wavelength (Figure 3.2), the impact of optical effects on the wafer worsens. The constructive and destructive interference of light as it passes through the photomask and the stepper (scanner) optics can easily induce diffraction effects that distort on-chip features, or even make them disappear, rendering the integrated circuit (IC) unusable.

As a consequence, design rules are exploding in number and complexity, making design rule checking (DRC) harder and lengthier. What we have observed across the industry is that the number of physical verification checks is growing at >20% node over node driven primarily by the growth of manufacturing process complexity. More alarming, the number of individual operations required to execute each check is also growing. The total number of operations within a physical verification deck is growing at >30% node over node. Figure 3.3 illustrates these growth patterns.

This runaway growth in both size and complexity has impacts throughout the IC manufacturing flow. Design rule manual developers are spending an inordinate amount of time trying to craft specialized rules that overcome manufacturing limitations and accurately satisfy the requirements of the design. Design teams

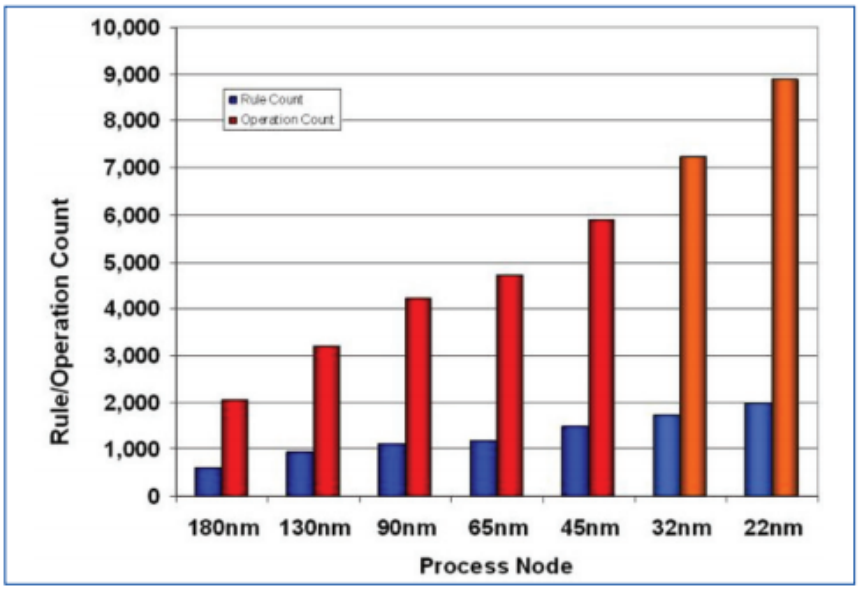


Figure 3.3: Growth in number and complexity of physical verification rules. Figure from [61]

must then spend even more time attempting to interpret these rules in complex rule checks that can contain hundreds of operations. A lot of valuable time and expertise is being used in an attempt to achieve congruence between the original intent of the design and its rendering as a physical implementation that can be profitably manufactured. Design teams are experiencing increased difficulty reaching physical implementation closure, longer physical verification runtimes, and escalating debugging difficulty and timelines.

The majority of physical verification requirements are based on one simple concept: certain combinations of geometric shapes cannot be successfully manufactured with a given process. Problematic topological configurations are identified through manufacturing process simulation, failure analysis, or other verification/validation techniques. Simulations and layout analysis techniques, for example, can identify areas of concern within a particular design - features or

configurations that will likely fail or negatively impact yield during manufacturing due to lithographic variability, planarity variation, or high sensitivity to random defects. Failure analysis, on the other hand, uses post-manufacture silicon testing and yield analysis techniques to identify and isolate systematic defects that appear repetitively across dies and designs.

Historically, these problematic configurations were textually defined in an engineering specification (design rule). This design rule was passed on to someone whose responsibility was to interpret the rule and write a new design rule check (using the physical verification scripting operations) that accurately represented the original pattern and design rule constraints. This design rule check would then be added to the rule decks used for physical verification. In this flow, then, these configurations are twice abstracted by the time the design rule check is implemented. Additionally, as advanced nodes are being implemented, problematic configurations are now being defined well before silicon production, generally by the teams using lithography and optical process simulations.

What we need is some efficient and accurate way to identify known problematic configurations in the physical design so they can be removed or improved before they cause failures in the manufacturing flow."

3.2 Applications of pattern match

To avoid yield limiting patterns in a design implementation, designers run DRC (Design Rule Check) or/and DFM (Design for Manufacturing) rules on their design during physical verification. Usually, those rules have been implemented in a text-based script. However, as their design becomes more complex along with the continuous shrinking of technology node, the text-based DRC and DFM rules have

been too lengthy and complicated. Sometimes, it was almost impossible to write rules to describe problematic patterns in order to eliminate them in their design since more and more rules are two-dimensional. In other words, design rules are getting increasingly complex with each new process node, and a yield limiting pattern might require hundreds of lines of traditional text-based script to express.

Pattern matching tool enables designers to implement complex design constraints with easy-of-use. It also helps more streamlined communication between designers and manufacturing foundries because pattern matching is a direct visual comparison between patterns rather than a long text description. Major benefits of using a pattern matching based approach are following as described at [17].

1. Reducing time required for rule deck development by simplifying and automating the creation of complex physical verification or design methodology checks that were previously difficult or operationally impossible to create using text-based scripting.

2. Reducing design variability by performing physical verification checks previously difficult or impossible to perform.

3. Simplifying debugging by providing a direct visual comparison between actual geometries, making it much easier to understand and fix violations.

4. Faster updates between manufacturing and design, enabling the quick accurate implementation of recently-identified yield-limiting patterns.

5. Improving consistency and accuracy across flows and between teams by enabling design, manufacturing and test teams to share pattern libraries across multiple tools. Pattern libraries can be created for specific design methodologies, manufacturing processes, or other categorizations.

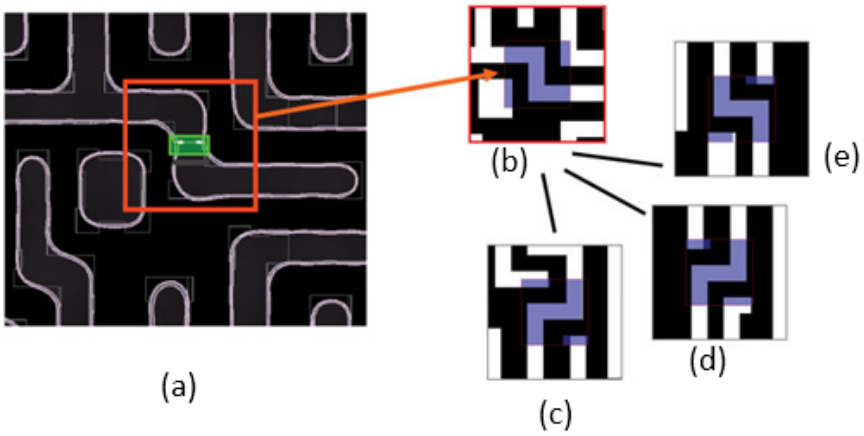


Figure 3.4: (a) Hotspot found at the foundry on wafer, (b) Hotspot pattern registered into a library, (c)(d)(e) patterns found in a design as hotspots by pattern matching tool. Figure from [74]

6. Improving communication between designers and fab/foundry by using actual patterns (rather than text-based abstractions) to create complex checks.

One of the most important applications of pattern match is that it can be used to detect yield limiting patterns (hotspots) that have been identified at manufacturing companies. Once designers have a hotspot pattern library provided by the foundry, they can run pattern matching tool to quickly find problematic patterns in their design. Figure 3.4 shows an example of hotspot registered in the library and several yield limiting patterns found in a design.

3.3 The fundamental idea

The fundamental idea of EDDR PM (Edge Driven Dissected Rectangles Pattern Match) is that any hotspot pattern can be represented by rectangles which are derived by edge lengths, widths, and/or spaces of polygons. These rectangles inside bounding box of a pattern become unique members to represent the pattern.

Efficient and flexible pattern matching is possible with the information about these member rectangles along with vector space created by the members and the bounding box.

The best way to derive member rectangles based on edge lengths, widths, and/or spaces of polygons is to employ geometry processing engine which is known as DRC (Design Rule Check) engine. It is well known and proven that DRC tool can handle polygon geometries and edges of those efficiently. With this industry level confidence, we adopt DRC engine for our EDDR PM.

3.4 Background

3.4.1 Design Rule Checks (DRC)

Design rules [76] are the rules provided by the process engineers to ensure manufacturability of the design layout. Process variations and technical limitations of the photo-lithography techniques make it necessary for each design to be DRC-clean before tape-out. Modern DRC rule sets are complex, but they always include the two most basic rules: width and spacing (Figure 3.5). The width rule prevents pinch-off of narrow shapes by defining a minimum width for any shape. Similarly, the spacing rule prevents bridging by defining a minimum distance allowed between two shapes. These rules can be expressed using constant values or equations/inequalities with variables. Violations of these rules are reported by the DRC tool by providing locations of the edges in violation. Besides these basic rules, there are many other DRC rules like area, ratio, overlap and density constraint rules. As design nodes are getting smaller and smaller, the DRC rules are getting more complicated and modern DRC tools must perform these checks efficiently.

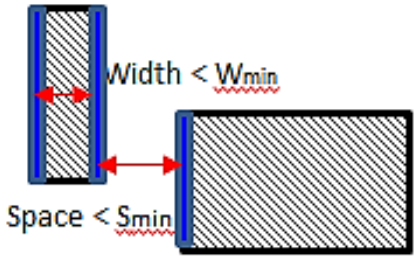


Figure 3.5: Minimum width/space check. Highlighted are edges that violate the width and space constraints

3.4.2 DRC Edge Operation

Edge operation of a DRC tool is a fundamental operation to check edge related rules. Some basic edge operations are LENGTH, WIDTH, SPACE, and ANGLE. The edges operations used in this paper are defined in 3.4.3. These operations can be used to generated edge-driven dissected rectangles as shown in Figure 3.7. Since it is a core operation of DRC tools, edge operation is usually optimized for speed, often by employing parallel computing. Because our proposal directly relies on the DRC edge operation, our solution naturally benefits from these advantages as well.

3.4.3 Formal definition of DRC edge operations

Definition 3.4.1. LENGTH is an edge operation function that takes three inputs (edges of polygons, length constraint, length value) and returns edges that meet the constraint. Length constraint can be any relational operators such as ==, >=, <=, and etc. It can be expressed:

E' = LENGTH (E, RO, value)

* E': edges that meet the length constraint.

* E: edges of polygons.

* RO: relational operator.

Definition 3.4.2. WIDTH is an edge operation function that takes five inputs (edges of polygons, edges of polygons, all edges of layer, width constraint, width value) and measure width between the first input edges and the second input edges facing inward polygons. It returns rectangles using the edges that meet the width constraint in the form of P1 in Fig. 2 (a). Width constraint can be any relational operators such as ==, >=, <=, and etc. It can be expressed:

R = WIDTH (E1, E2, E3, RO, value)

* R: rectangles formed by edges that meet the width constraint.

* E1, E2, E3: edges of polygon. E1 and E2 are subset of E3.

Definition 3.4.3. SPACE is the same edge operation function as WIDTH except that it measures space between the first input edges and the second input edges facing outward polygons. It can be expressed:

R = SPACE (E1, E2, E3, RO, value)

Example 1. P1 in Fig. 2 (a) can be created by LENGTH and WIDTH DRC operation.

// a is length for Len_A. Metal_1 is Metal_1 layer's edges.

// b is length for Len_B. Metal_1 is Metal_1 layer's edges.

// 0.3 is width value.

Len_A = LENGTH (Metal_1, ==, a)

Len_B = LENGTH (Metal_1, ==, b)

P1 = WIDTH (Len_A, Len_B, Metal_1, ==, 0.3)

Definition 3.4.4. ANGLE is an edge operation function that takes three inputs (edges of polygons, angle constraint, angle value) and returns edges that meet the

constraint. Angle constraint can be any relational operators such as ==, >=, <=, and etc. It can be expressed:

E' = ANGLE (E, RO, value)

Example 2. Fig. 15 is one of cases that may have different ways to decompse into member rectangles. If we apply "ANGLE == 0" on Len_1, we get Fig. 15 (b). If "ANGLE == 90" on Len_1 is applied, we get Fig 15 (c). DRC operations below are to create members as Fig. 15 (b).

P1_Len_1 = ANGLE (LENGTH (Metal_1, ==, 2), ==, 0)

P1_Len_2 = LENGTH (Metal_1, ==, 2)

P1 = WIDTH (P1_Len_1, P1_Len_2, Metal_1, ==, 2)

P2_Len_1 = ANGLE (LENGTH (Metal_1, ==, 2), ==, 0)

P2_Len_2 = LENGTH (Metal_1, ==, 1)

P2 = WIDTH (P2_Len_1, P2_Len_2, Metal_1, ==, 0.5)

Definition 3.4.5. OR is an polygon operation function that arbitary number of polygons as inputs (P1,...,Pn) and returns the union of polygon regions. It can be expressed:

P' = OR (P1,...,Pn)

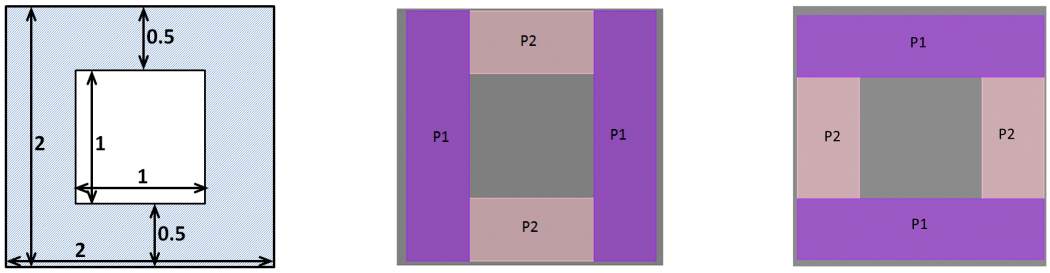
* P': merged polygons.

* P1,..,Pn: polygons.

Definition 3.4.6. NOT is an polygon operation function that takes two inputs (P1,P2) and returns P1 without overlapped region with P2.It can be expressed:

P' = NOT (P1,P2)

* P': P1 polygons without overlapped area with P2.



(a) (b) (c)

Figure 3.6: Example of applying ANGLE (a) pattern that can be decomposed in different ways; (b) ANGLE == 0 on Len_1; (c) ANGLE == 90 on Len_1

* P1,P2: polygons.

3.5 Method of EDDR PM

3.5.1 Pattern Description

The pattern is described by member rectangles inside the pattern bounding box. Each member rectangle is derived by edge operations as explained in Figure 3.7. We can create member rectangles based on edge length and width, or edge length and space. And a member rectangle can be described by this simple format below.

<Name of member>

Len_1 == value (um or any user unit)

Len_2 == value

Width == value or Space == value

Using this information, we can perform DRC operations such as LENGTH, WIDTH, or SPACE to generate member rectangles. Since there are some patterns that may have different ways to decompose, we perform additional DRC operation of ANGLE during member rectangles generation for those patterns to enforce only

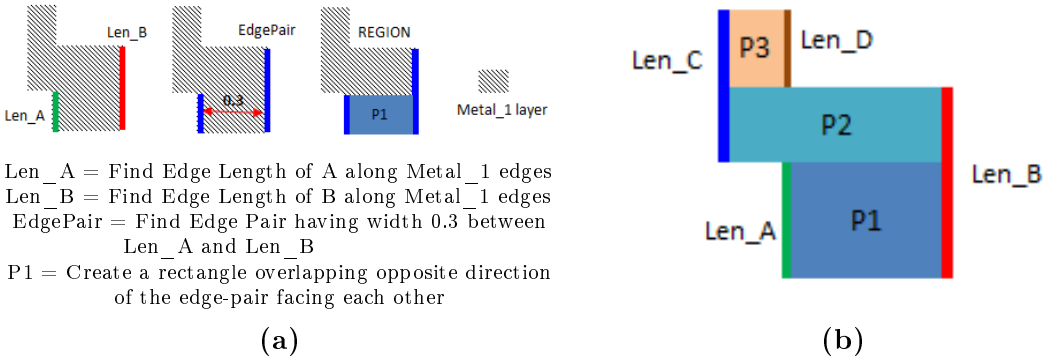


Figure 3.7: Member rectangle creation example using edge-driven dissection. (a): Edge operation example. (b): Example of Edge Driven Dissected Rectangles. P1, P2, and P3 are generated by the process described in (a).

one way of decomposition (see example 2 in 3.4.3). Formal definitions of those DRC operations are defined in 3.4.3.

Figure 1.12 is an example of pattern description that has six member rectangles inside pattern bounding box. In this case, Len_1 and Len_2 are the same, but in general, those can be different as explained in Figure 3.7. In this example, we have 6 members. Any one of them can be origin member and one of the remainders can be the first reference member. (Note: We do not need P1 which is derived from space check for this particular pattern match. We can use P4 or any one of the other rectangles as the origin rectangle. We derived P1 in order to demonstrate that we can also use space check for our pattern matching method.)

Using the center point of the origin member and the center point of the first reference member, we can create vector space and other necessary information that is used to perform pattern matching. Figure 3.8 illustrates this. In this example, P1 is the origin member and P2 is the first reference member to form vector space. Besides the vector information, we need to store other information described in Figure 3.8 for pattern match. The information of each member rectangle we store

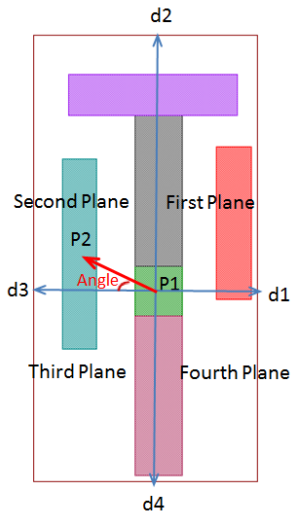


Figure 3.8: Vector information (angle and distance between origin members̀ center to the reference members̀ center point) and other necessary information we store as the pattern description.

for the pattern description is as follows:

1. Vector information (angle and distance between origin and reference)
2. Plane location from origin to reference (one of 4 planes or 4 along-axis)
3. Distance between each center of member rectangle and the bounding box. (d1, d2, d3, and d4)
4. The width and the height
5. A Boolean to indicate whether it is the width or the space rectangle (for example, P1 is Space rectangle.)

This information for each member is stored in PDB (Pattern Description Database) which we will use for pattern match. With this information, we can distinguish 8 different orientations (4 rotations X 2 mirrored images) of any pattern, which eliminates the unnecessary 8 iterations to detect same pattern with different

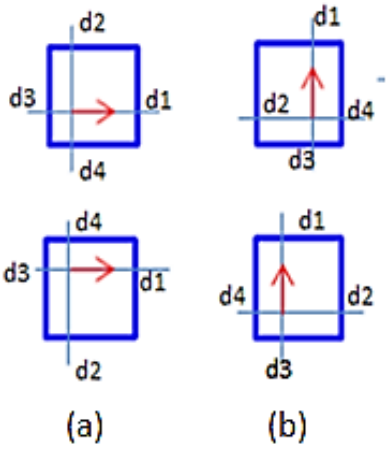


Figure 3.9: Cannot distinguish the top one and the bottom one using the vector information between origin member and the reference member. Both top and bottom have the same angle and same distance along-axis, but they have different d2 and d4. (a): flipping along x-axis case. (b): flipping along y-axis case.

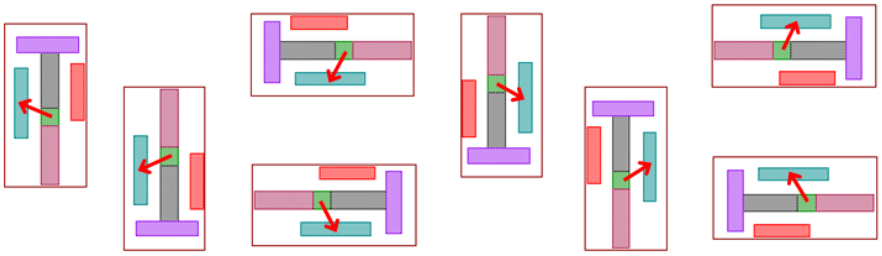


Figure 3.10: Eight different orientations of the same pattern. These are distinguishable using the vector information.

orientations. Figure 3.10 demonstrates this. We also store in PDB Len_1, Len_2, Width or Space value for each member as well as their corresponding relational operator for fuzzy pattern match which will be discussed in 3.14. The only exception is when the vector direction is along-axis or when the angle is 45. In these cases, we have to iterate two times to cover two cases during the pattern matching process. Figure 3.9 explains why we need to iterate two times in the case of along axis.

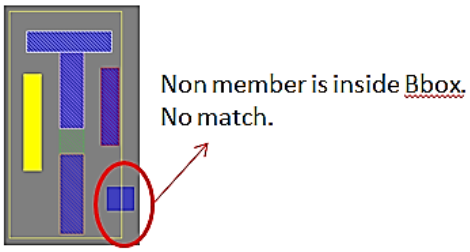


Figure 3.11: When non-member interacts with the bounding box, it is immediately classified as no match.

3.5.2 Pattern Match

With pattern description information explained in 3.5.1, we can run the pattern matching process on a layout by following the simple algorithm (Algorithm 1). It is a brute force algorithm visiting all the origin members one by one in the layout. We can improve this by adopting a bin-search grid algorithm which will be shown in Algorithm 2.

As indicated in the Algorithm 1, we need to take care of non-members inside the pattern bounding box during the pattern matching process. If there is a non-member polygon inside the bounding box, it is immediately classified as no match (Figure 3.11). To do this, we pass non-member polygons to EDDR_PM (Edge Driven Dissected Rectangle Pattern Match), which are created by OR and NOT operations. For example, non-members of Figure 1.12 are:

Non-member = Metal_1 NOT [1] (OR P1 P2 P3 P4 P5 P6)

At step (8) and (18) in the Algorithm 1, it uses scan line based topological check which is not ideal in terms of runtime performance. We can improve it significantly by using a bin-search grid algorithm. Another criterion we examine for an early invalidation of a match for EDDR_PM is passing the total number of

[1] Refer to 3.4.3 for NOT and OR operation

Algorithm 1 EDDR PM (Pattern Match Using Edge Driven Dissected Rectangle)

```
1: procedure EDDR–PM(P1, P2, ...., Pn, nonMem, PDB)
2:     P1 = set of origin members in a layout
3:     P2 = set of the first reference members in a layout
4:     P3..n = set of all the other reference members in a layout
5:     nonMem = set of non-member polygons in a layout
6:     PDB = a pattern description database.
7:     while !empty in P1  do
8:         Find a reference member p2 in P2 by searching the vector distance
9:         between P1's center and P2's center.(PDB has this info.)
10:        if found then
11:            if the found p2 a valid reference member then
12:                Create a bounding box using d1, d2, d3, and d4 determined by
13:                vector info between P1 and the found P2's center.
14:            else
15:                No match. continue to next p1 in P1
16:            if nomMem exists inside the bounding box then
17:                No match. continue to next p1 in P1
18:            Find other members in P3 · · · Pn inside the bounding box.
19:            for each member inside the bounding box do
20:                n = number of each member inside bounding box
21:                m = number of each member described in PDB
22:                if n != m then
23:                    No match. continue to next p1 in P1
24:                if valid member == false then
25:                    No match. continue to next p1 in P1
26:            Matching pattern found at this point.
27:            Output the bounding box to indicate the match.
28:            continue to next p1 in P1
29:        else
30:            No match. continue to next p1 in P1
```

each member to EDDR_PM. If the number does not match inside the bounding box, it is classified as not a match right away (Figure 3.12).

We use information in PDB format as explained in 3.5.1 for valid member check and creating the bounding box. At step (12) in the algorithm, we can

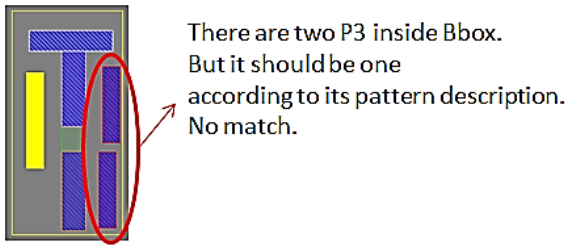


Figure 3.12: Immediate mismatch when the total number of each member inside the bounding box does not match.

determine which orientation we want among the 8 possible ones by calculating the vector (angle and distance) between P1 and the found P2 and referencing the information in PDB. If its plane is at along-axis or its angle is 45 degree, we create two bounding boxes and do the subsequent checks for each bounding box in the algorithm.

Because Algorithm 1 is a brute force search using the topological scan-lines, it is best to have as small number of P1 as possible to reduce runtime. We can do this by adding an extra edge operation related to other members when creating origin members. For example, we can add a space check between the origin member and some of the other reference members for final derivation of the origin members. This additional edge operation to reduce the total number of origin members in a layout does not incur much additional runtime. However, it reduces EDDR_PM runtime significantly.

Even though Algorithm 1 can do a decent job by reducing the number of P1, it is not sufficient to handle a huge number of P1 rectangles presented in the layout. So, we developed another algorithm, Algorithm 2, which utilizes a bin-search grid technique. As presented in 3.13, it achieved significant runtime reduction.

Algorithm 2 EDDR PM (Pattern Match Using Edge Driven Dissected Rectangle)

```
1: procedure EDDR–PM(P1, P2, ...., Pn, nonMem, PDB)
2:     Inputs (P1..n, nonMem, and PDB) are the same as Algorithm 1.
3:     ADD_BIN for each member from P1 to Pn.
4:     LOCATE_BIN for all the origin members of P1 and get bin_counts
5:     for i = 1 → bin_counts for p_i in P1 bins do
6:         LOCATE_BIN a reference member, p2, in P2 bins by searching
7:         the vector distance between P1's center and P2'2 center.(PDB has this
   info.)
8:         if found then
9:             Same process as Algorithm 1 to decide match or no match
10:            .....
11:            LOCATE_BIN for other members inside the bounding box.
12:            .....
13:            Same process as Algorithm 1 to decide match or no match
14:            .....
15:        else
16:            No match. continue
```

3.5.3 Bin-Search Grid

A bin-search grid is efficient for finding objects which interact in a 2D space. It is typically much faster than topological scan-lines which must process all the objects in a single scan. It uses an adaptive structure for rectangle search via binning. The structure starts with a fixed pixel size but will re-grid more finely when the average number of objects in a bin becomes excessive. Usually rectangular extents of geometric objects are used to insert into the grid. The grid is first populated with a series of ADD_BIN methods. ADD_BIN has a rectangle input along with an associated ID for the object. The grid can then be searched with the LOCATE_BIN method. LOCATE_BIN has a search rectangle as input, and returns a set of object IDs that interact with the rectangle.

The bin-search grid has a fixed overall rectangular bounding box, usually layout

extent, which is supplied by the client at initialization time. This rectangle is divided into a 2D grid with an initial default pixel size. ADD_BIN and LOCATE_BIN can then directly calculate the row and column elements of the grid to analyze using the pixel size. ADD_BIN contains heuristics to decrease the pixel size and recalculate the grid when the number of bins in the grid elements becomes large.

LOCATE_BIN analyzes the intersecting grid elements with a search rectangle and builds a list of unique bin IDs that have been previously added. The bin-search grid is efficient when the added bin extents, which are member rectangles in our case, are small relative to the overall layout extent bounding box. Since pattern bounding box is, in general, so small relative to the layout extent that it lies in one grid element in our application. Figure 3.13 explains it graphically. In this picture, bounding box has 3 by 3 grid elements to locate bins inside it. Because LOCATE_BIN has O(k) where k is the total number of grid elements overlapped by a search extent rectangle, it requires O(9)to locate ID1 and ID2 bin.

3.5.4 Computational Complexity

Let n denote total number of P1 and let m denote total number of all members in a layout. Since Algorithm 1 visits all origin members in P1, it is O(n) for the while loop. The scan-line search to find other members at the step (8) and step (18) inside the while loop requires m times topological check per each loop. Therefore, the complexity becomes O(nm). Because of m >= n, we can say that O(n(n+c)) where c is a constant, and it becomes O(n^2).

Those two steps in Algorithm 1 have been replaced with LOCATE_BIN for Algorithm 2. Since LOCATE_BIN's time complexity is O(k) where k is the total

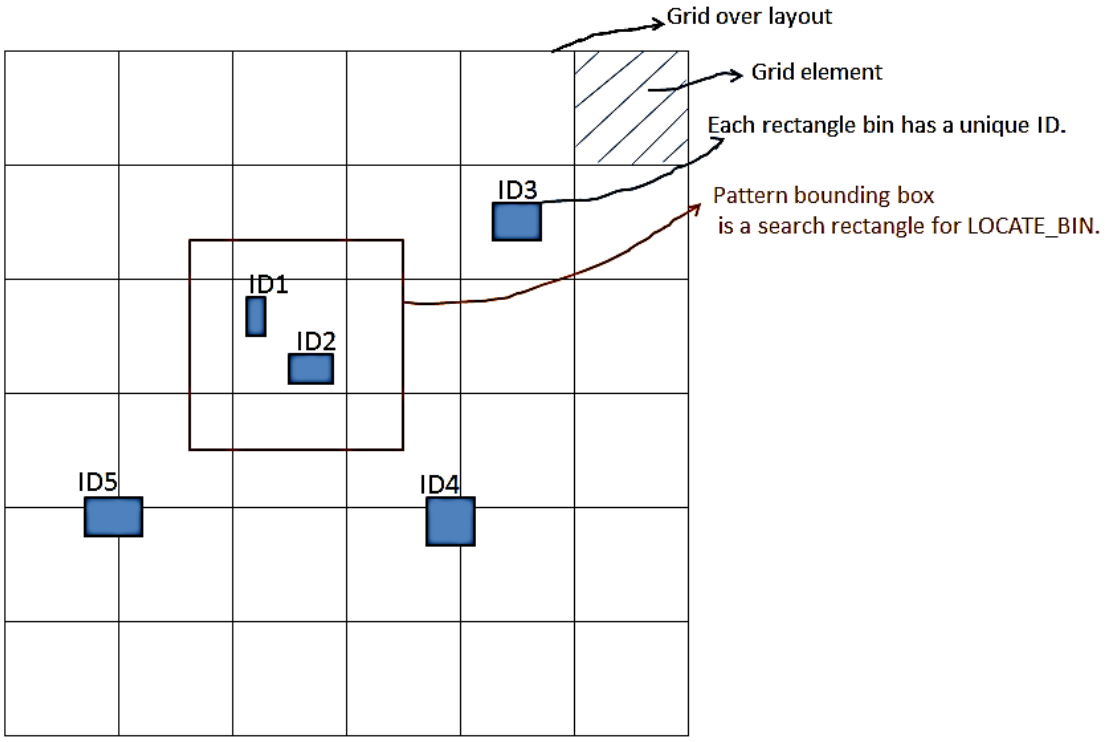


Figure 3.13: Bin-search gird

number of grid elements overlapped by a search box, Algorithm 2 has O(nk). Because k «n or k = 1 in general in our pattern match process, it is O(n) in practice. Table II compares these two algorithms and shows substantial runtime difference.

3.5.5 Fuzzy Pattern Match [2]

With our simple approach to pattern match, we could see another benefit of EDDR_PM when it comes to fuzzy pattern matching. Figure 3.14 illustrates this. Because we can use not only == but also other relational operators (>, =>, <, =<) for pattern description, we can describe a pattern in a fuzzy way and do

[2]There is a limitation in our approach for fuzzy match. Our approach cannot handle matching from post-OPC (Optical Proximity Correction) pattern to pre-OPC pattern. However, tolerance-based match [10] can be easily accomplished in our approach. Refer to 3.5.6.

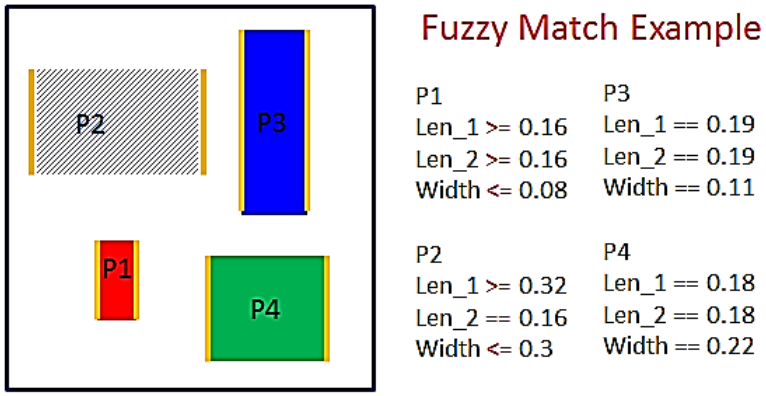


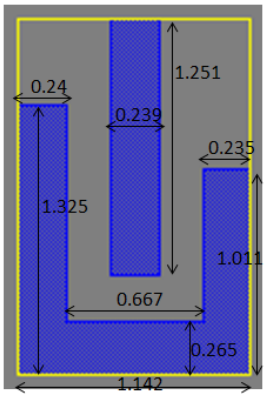
Figure 3.14: Any P1 meeting the constraints can be a member of the pattern. Any P2 meeting the constraints can be a member of the pattern.

a fuzzy pattern match.

In this case, the vector space information is no longer valid. We can use the number of each member inside bounding box for fuzzy pattern match. Therefore, we skip validation checks at the step (24) and (11) of Algorithm 1 for members that are derived from relational operators except == operator. It also must either have at least one member rectangle created by only == operator inside the bounding box or have a configuration where origin member rectangle's center point is unchanging.

Since PDB has information about relational operator used for each member, we can decide whether to do fuzzy match or exact match for each member. If == operator is not used for Len_1, Len_2, or Width/Space, we do fuzzy match for that member by skipping member validation check at step (24) of Algorithm 1. If reference member is described as fuzzy member, we skip the step (11) of Algorithm 1 and need to iterate 8 times for fuzzy match. Algorithm 3 is fuzzy match algorithm. Note that step (9) and step (28) in Algorithm 3 for fuzzy member check are added.

Figure 3.15 shows fuzzy match examples in details. (b) is for exact match where



(a)

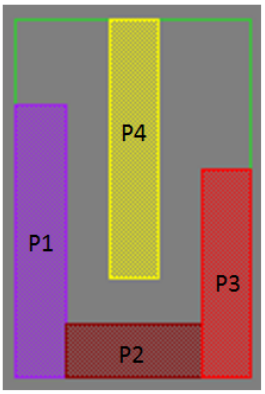
P1
Len_1 == 1.142
Len_2 == 0.240
Width == 1.325

P2
Len_1 == 1.142
Len_2 == 0.667
Width == 0.265

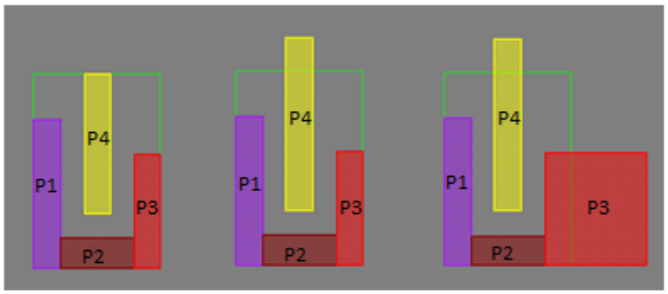
P3
Len_1 == 1.142
Len_2 == 0.235
Width == 1.011

P4
Len_1 == 1.251
Len_2 == 1.251
Width == 0.239

(b)



(c)



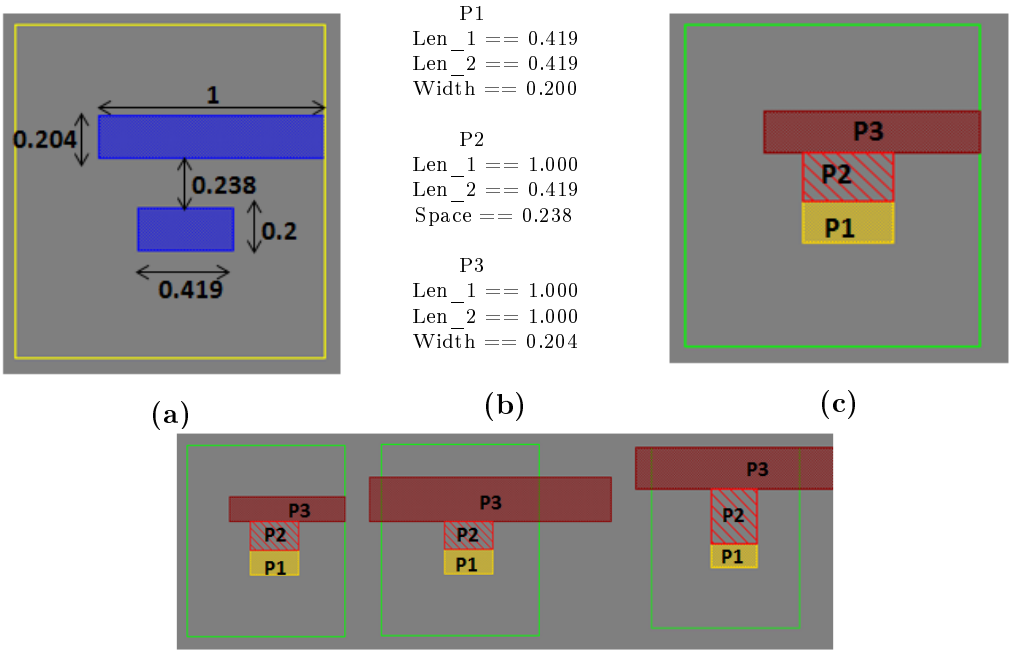
(d)

P1	P2	P3	P4
Len_1 >= 1.142	Len_1 >= 1.142	Len_1 >= 1.142	Len_1 >= 1.251
Len_2 == 0.240	Len_2 == 0.667	Len_1 >= 0.235	Len_1 >= 1.251
Width == 1.325	Width == 0.265	Width == 1.011	Width == 0.239

(e)

Figure 3.15: (a) geometric pattern to match; (b) exact pattern description using only == operations; (c) member rectangles created from exact patern description of (b) and green bounding box for matched pattern; (d) member rectangles created from fuzzy pattern description of (e) and green bounding boxes indicating matches; (e) fuzzy pattern description using range relational operations

we have four member rectangles, P1, P2, P3, and P4. (e) uses relational operators to perform the fuzzy match. Green bounding boxes are outputs from EDDR_PM when it found matches. Another fuzzy match example is described at Figure 3.16 where Space member rectangle is applied.



P1	P2	P3
Len_1 == 0.419	Len_1 >= 1.000	Len_1 >= 1.000
Len_2 == 0.419	Len_2 == 0.419	Len_1 >= 1.000
Width == 0.200	Space >= 0.238	Width >= 0.204

(e)

Figure 3.16: (a) geometric pattern to match; (b) exact pattern description using only == operations; (c) member rectangles created from exact patern description of (b) and green bounding box for matched pattern; (d) member rectangles created from fuzzy pattern description of (e) and green bounding boxes indicating matches; (e) fuzzy pattern description using range relational operations

In the fuzzy match example of Figure 3.15, note that P1 and P2's center points are not changed to perform successful fuzzy pattern matching. As long as we have two unchanging center points, one or two iteration is sufficient for pattern match. If there is only one member with its unchanging center point, we have to iterate 8 times to cover all the 8 orientations, which is the case of Figure 3.16.

3.5.6 Tolerance-based Match

Widely used resolution enhancement technique in lithography process during chip manufacturing may create process-hotspots from patterns which are quite similar to a hotspot pattern and only have tiny width or space differences. Basically, process-hotspot can have slightly different topologies from hotspot patterns for the match. Enumerating all these variant topologies as hotspot patterns is not practical, and there must be a representative pattern with edge tolerance and incomplete specified region [10].

Our approach can be easily extended to solve this issue by adding range relational operations along with partial match we presented at Section 3.5.7. For example, a member can be defined as:

Len_1 => a <= b (Len_1 is between length a and b.)

Len_2 >= c <= d (Len_2 is between length c and d.)

Width >= e <= f (Width is between width e and f.)

With this range specification, we can specify edge tolerance. By using this edge tolerance and our "Don't care region" (incomplete specification) approach, we can find process-hotspots as [10].

3.5.7 Partial Match

Another interesting case for our approach is when we try to do partial match. For example, we can match a pattern in Figure 3.17 by skipping the step (16) in Algorithm 1. Therefore, we can match patterns that contain non-orthogonal edges as well. As long as there are a couple of member rectangles in a pattern, we can do partial match, which means it can do match for incompletely-specified patterns.

Other approaches for partial match create "Donŕ care" regions. For example,

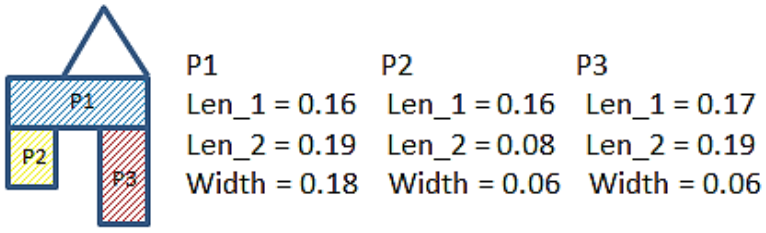


Figure 3.17: Partial match example

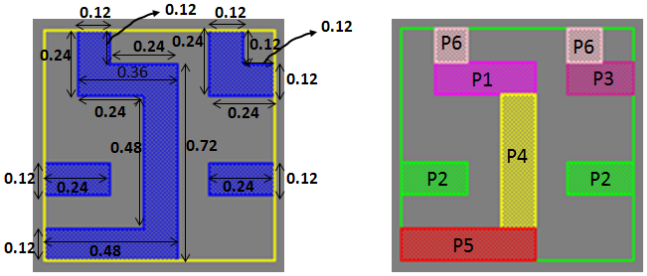
[10] tried to redefine their method to reflect the impacts of "Donť care" regions for partial match. However, our approach does not need to add additional efforts to deal with a concept of "Donť care" regions for a partial match because it is automatically defined. If some polygons or parts of a polygon are not specified in a bounding box of a pattern like Figure 3.17, our algorithm does not care those and perform a partial match.

Figure 3.18 shows partial match experimental results using "Ind1" test pattern of [11]. Figure 14 (a) and Figure 14 (b) illustrates "Ind1" pattern configuration and its pattern description. Figure 14 (c) is showing partial match results when we skip the step (16) in Algorithm 1. Figure 14 (d) depicts partial match results when we not only skip it but also we donť specify P4 in pattern description not to create members for parts of a polygon at the first place as Figure 14 (e).

3.6 Experimental Result

Our experiments were performed on a Linux platform with 3.7 GHz clock CPU and 32 GB RAM. We created 9 different patterns to match (Figure 3.19). Real industry layout was used for this experiment. The area and number of polygons inside each layout are shown at Table 3.1.

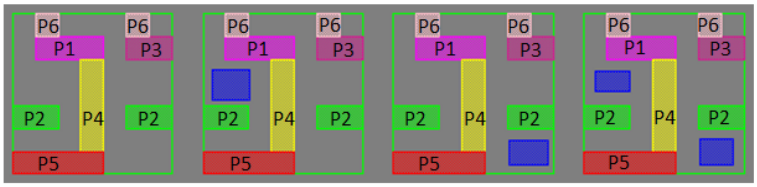
First we compared performance between Algorithm 1 and Algorithm 2. Table



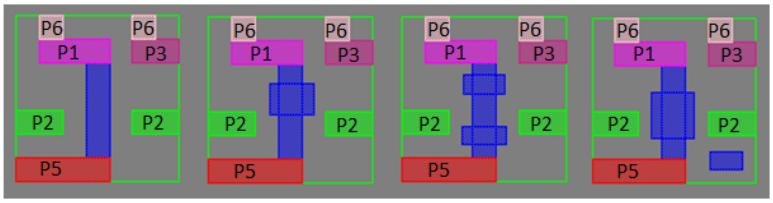
(a)

P1	P2	P3	P4	P5	P6
Len_1 == 0.24 Len_2 == 0.72 Width == 0.36	Len_1 == 0.12 Len_2 == 0.12 Width == 0.24	Len_1 == 0.24 Len_1 == 0.12 Width == 0.24	Len_1 == 0.48 Len_2 == 0.72 Width == 0.12	Len_1 == 0.12 Len_2 == 0.72 Width == 0.48	Len_1 == 0.24 Len_2 == 0.12 Width == 0.12

(b)



(c)



(d)

P1	P2	P3	P5	P6
Len_1 == 0.24 Len_2 <= 0.72 Width == 0.36	Len_1 == 0.12 Len_2 == 0.12 Width == 0.24	Len_1 == 0.24 Len_1 == 0.12 Width == 0.24	Len_1 == 0.12 Len_2 <= 0.72 Width == 0.48	Len_1 == 0.24 Len_2 == 0.12 Width == 0.12

(e)

Figure 3.18: (a) Ind1 pattern to match; (b) exact pattern description using only == operations; (c) partial match results by skipping non-member check. member rectangles created from exact patern description of (b) and green bounding boxes for matched patterns; (d) partial match results by both skipping non-member check and removing P4 member creation. member rectangles created from fuzzy pattern description of (e) and green bounding boxes indicating matches; (e) partial pattern description in fuzzy way using range relational operations

Table 3.1: Layout information for TableII and TableIII (*tp9 4M: 4 million of tp9 exist in the layout.)

	Layout for tp1 to tp9	Layout for tp9 4M
Area(mm^2)	1.5 x 1.5	2 x 2
Number of Polygons	5,207,283	9,170,937

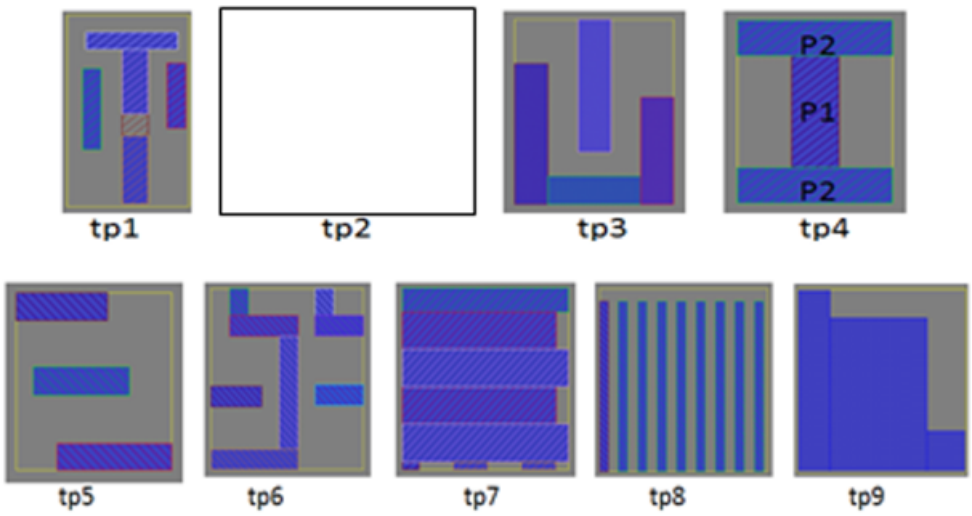


Figure 3.19: Test patterns to match for our experiments. tp2 is a clip from the real design of layout 1, which is whited out due to proprietary concerns. tp3, tp4, tp5, tp6, tp9 are from [1].

3.2 shows the result. This result makes it clear how efficient Algorithm 2 is and at the same time how inefficient the brute force Algorithm 1 is when there are many hotspots to match. It is important to note that the result shown in the table proves our previous assertion: "It scales as well with increasing number of patterns to match because it needs to go through all edges only once in the chip on which it does pattern matching, regardless of the number of patterns to match."

Note in Table 3.2 that # of P1 is equal to a number of hotspots in the layout. tp9 4M is a test case where there are 4 million of tp9. DRC1 denotes edge operation

Table 3.2: Algorithm 1 VS. Algorithm 2

	Pattern	# of locations of DRC1	# of locations of DRC2	# of P1	DRC1 (sec)	DRC2 (sec)	EDDR(sec)	Total(sec)	Success rate
Algo 1	tp1	3,298,447	38,663	6,336	40	2	17	59	100%
	tp2	13,772,975	4,627,518	7,219	44	17	25	86	100%
	tp3	1,934,668	115,371	37,581	37	1	172	210	100%
	tp4	488,596	172,476	112,856	37	1	389	427	100%
	tp5	243,343	28,656	9,336	37	1	13	51	100%
	tp6	721,911	154,422	25,518	37	1	111	149	100%
	tp7	337,058	357,696	19,480	39	1	66	106	100%
	tp8	271,188	406,782	15,236	40	1	81	122	100%
	tp9	6,664,603	163,984	40,996	38	1	112	151	100%
	tp9 4M	26,491,623	16,025,600	4,016,400	57	23	205,772	**205,852**	100%
Algo 2	tp1	3,298,447	38,663	6,336	40	2	5	47	100%
	tp2	13,772,975	4,627,518	7,219	44	17	5	66	100%
	tp3	1,934,668	115,371	37,581	37	1	4	42	100%
	tp4	488,596	172,476	112,856	37	1	4	42	100%
	tp5	243,343	28,656	9,336	37	1	4	42	100%
	tp6	721,911	154,422	25,518	37	1	4	42	100%
	tp7	337,058	357,696	19,480	39	1	4	44	100%
	tp8	271,188	406,782	15,236	40	1	4	45	100%
	tp9	6,664,603	163,984	40,996	38	1	4	43	100%
	tp9 4M	26,491,623	16,025,600	4,016,400	57	23	16	**96**	100%

time to run through all edges in the layout for finding Len_1 and Len_2 for member rectangles. DRC2 is edge operation time to create member rectangles based on the found edges in DRC1. EDDR column means time for Edge Driven Dissected Rectangle Pattern Match.

The only difference for EDDR between Algo1 and Algo2 is that Algo2 used bin-search grid algorithm while Algo1 is a topological scan-line search algorithm. The success rate is matching success rate. There are no false positive matches. Our approach guarantees 100% accurate match.

Secondly, we compared our Algorithm 2 with a state-of-art DRC-based commercial pattern matching tool which uses the hashing of polygon vertices [23]. We summarized its result in Table 3.3 where you can see that our approach is 2 to 10 times faster than the commercial tool.

Table 3.3: Alogrithm 2 VS. Commercial Vertices Hashing

Pattern	Algo2 (sec)	Vertices hashing	Speed up
tp1	47	186	4.0
tp2	66	365	5.5
tp3	42	268	6.4
tp4	42	211	5.0
tp5	42	403	9.6
tp6	42	447	10.6
tp7	44	380	8.6
tp8	45	387	8.6
tp9	43	96	2.2
tp9 4M	96	167	1.7

We also compared [11]'s result to our approach. Because it is not a commercial tool available for us to try, we followed the author's metrics of [11] to have a fair comparison. As you can see at Table 3.4, their approach suffers longer runtimes when there are too many locations that DRC reports from their critical rules. For example, in their Layout2, they had tp9 called "Stair 1", and DRC reported less than 4 million of locations of it. But their approach caused a dramatic runtime impact. Depending on a pattern configuration, [11]'s approach suffers runtime degradation in their Pre-filtering as well as Finalization, while our approach shows fast EDDR_PM time and consistent DRC runtime regardless of pattern configurations because we go through all edges only once using simple edge operation rules to create locations of interest.

To be more convinced that our approach is better than [11], we obtained benchmark data from the authors of [11], which included their layout, patterns, and matching results. Figure 3.20 presents their test patterns and layout information. For a fair comparison, we set exactly the same experimental settings as their benchmark, using the same 2.4 GHz CPU and 16GB memory on a Linux machine. Table 3.5 shows our results against their benchmark data. As we expected, our

Table 3.4: Alogrithm 2 VS. [11]'s critical design rule extraction method

	Pattern	# of location of DRC		# of location after pre-filtering	number of hotspots	DRC(sec)		pre-filtering (sec)	Finalization (sec)	Total (sec)	Success rate
[11]	tp3(mountain)	97,418		9,600	9,600	11		2.98	3.54	17.52	100%
	tp4(I)	67,640		38,400	9,600	9		2.27	56.25	67.52	100%
	tp5(Stair2)	115,950		19,200	9,600	9		3.99	14.18	27.17	100%
	tp6(Ind1)	895,377		9,600	9,600	35		50.44	3.61	89.05	100%
	tp9(Stair1)	3,710,439		57,592	9,600	63		164	125	352	100%
Algo2		DRC1	DRC2			DRC1	DRC2		EDDR(sec)		
	tp3(mountain)	1,934,668	115,371	N/A	37,581	37	1	N/A	4	42	100%
	tp4(I)	488,596	172,476	N/A	112,856	37	1	N/A	4	42	100%
	tp5(Stair2)	243,343	28,656	N/A	9,336	37	1	N/A	4	42	100%
	tp6(Ind1)	721,911	154,422	N/A	25,518	37	1	N/A	4	42	100%
	tp9(Stair1)	6,664,603	163,984	N/A	40,996	38	1	N/A	4	43	100%
	tp9(Stair1) 4M	26,491,623	16,025,600	N/A	4,016,400	57	23	N/A	16	96	100%

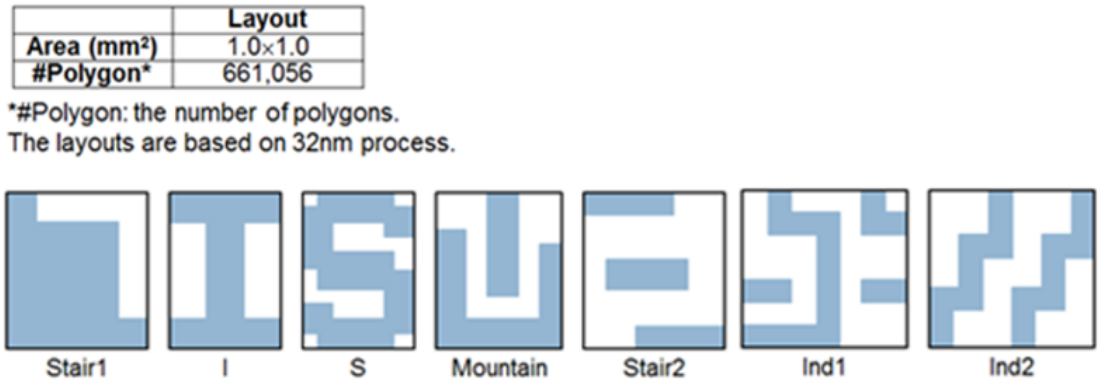


Figure 3.20: [11]'s layout information and test patterns

approach was 20 times faster on average. It was 58 times faster for "Stair1".

As another data point, there is a recent work [28] that compared exactly same benchmark data of [11] we used in this paper. Table III in [28] shows [28]'s approach is faster than only 5.6 times on average and only about 18 times faster for "Stair1". Table 3.5 includes [28]'s result as well.

Table 3.5: Alogrithm 2 VS. [11] and [28]

[11]						[28]			Algorithm 2				
Pattern	# of hotspots	DRC (sec)	pre-filtering + finalization (sec)	Total (sec)	Success Rate	Total (sec)	Success Rate	Speedup	DRC1 + DRC2	EDDR	Total (sec)	Success Rate	Speedup
Mountain	12,800	14.83	7.09	21.92	100%	19.9	100%	1.1	4	0.2	4.2	100%	5.2
S	12,800	45.89	86.00	131.89	100%	22.5	100%	5.9	8	0.5	8.5	100%	15.5
Stair1	12,800	45.81	318.93	364.74	100%	19.9	100%	18.3	6	0.3	6.3	100%	57.9
I	12,800	10.74	52.47	63.21	100%	20.1	100%	3.1	3	0.2	3.2	100%	19.8
Ind1	12,800	46.68	69.76	116.44	100%	21.6	100%	5.4	5	0.5	5.5	100%	21.2
Ind2	12,800	55.83	43.63	99.46	100%	23.5	100%	4.2	5	0.5	5.5	100%	18.1
Stair2	12,800	9.61	17.57	27.18	100%	19.7	100%	1.4	4	0.2	4.2	100%	6.5

3.6.1 Multi-layer pattern matching

Since DRC operations to create member rectangles for EDDR PM can be performed not only on a single layer but also in between two layers, it is simple and easy to expand EDDR PM to multi-layer pattern matching. For two-layer pattern matching, We can run DRC LENGTH operation defined in the section, 3.4.3, on one layer for Len_1 and the other layer for Len_2. And then, WIDTH or SPACE operation defined in the same section performs on the edges between Len_1 and Len_2 to create member rectangles. Figure 3.21 depicts an example of creating member rectangles for two-layer pattern matching. As shown at the Figure 3.21, P1 and P2 member rectangles are created from edges between two different layers. If we want to put more constraints for the match, we can create P3 and P4 member rectangles as well.

In a similar way, three-layer pattern matching can be performed as shown at Figure 3.23. The only difference is that there are more ways and freedom to generate member rectangles. The question of how many member rectangles we need depends on whether we want a partial match or not. For example, if we want

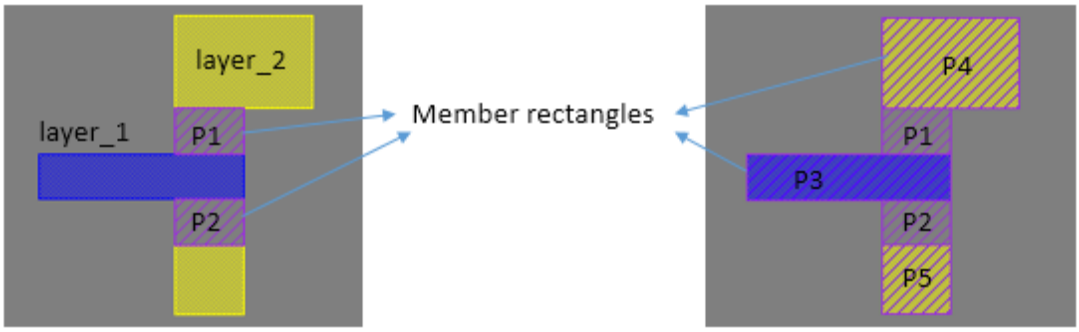


Figure 3.21: Member rectangles creation for two-layer pattern matching. P1 and P2 are created between two different layers. P3, P4, and P5 can be created as well for exact match.

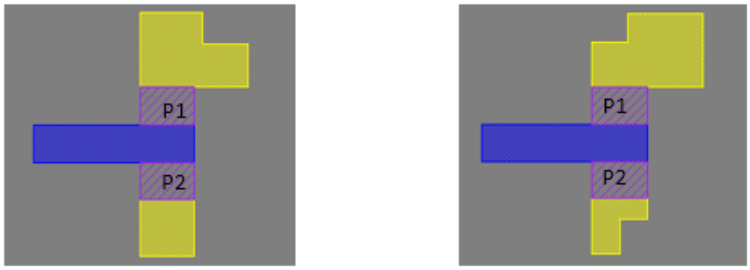


Figure 3.22: Two-layer partial match when only P1 and P2 are created.

an exact match of 3.21, we have to create all member rectangles such as P1, p2, P3, P4, and P5. However, if we create only P1 and P2, there is a possibility to match non-exact patterns such as 3.22.

3.6.2 Other DRC operations for future work

We only used several DRC operations such as LENGTH, WIDTH, SPACE, and ANGLE for our EDDR PM. Those are sufficient for exact matching, partial matching, and fuzzy matching. However, there are many other DRC operations that may be applied to demonstrate their effectiveness for other matching space such as multi-layer matching or for enhancement of EDDR PM. In this subsection, we introduce several more DRC operations as possible candidates for future work

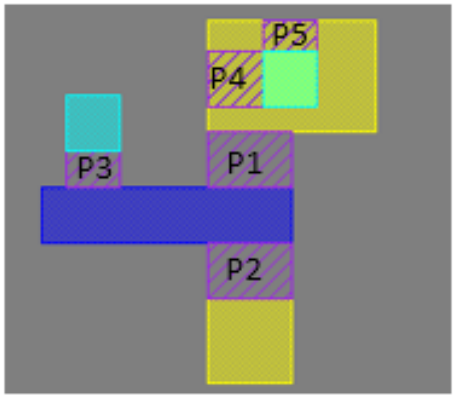


Figure 3.23: Member rectangles creation for three-layer pattern matching. Members are created between edges from two different layers.

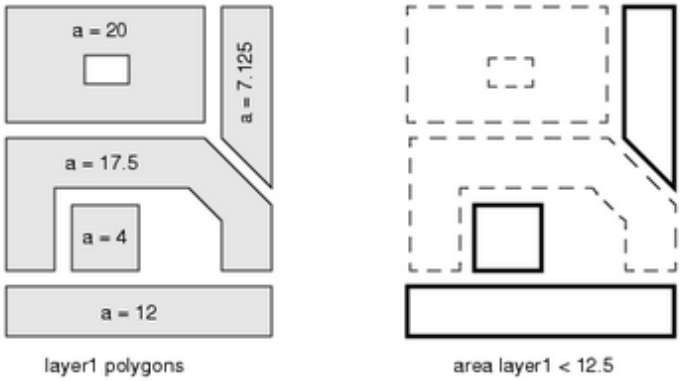


Figure 3.24: AREA operation. Area constraint is < 12.5. Black solid polygons are output polygons from AREA operation. Figure from [24]

related to pattern matching.

1. AREA

   AREA operation takes a polygon layer and selects all polygons that have areas meeting area constraint. Figure 3.24 shows an example of AREA operation. This operation may be useful for fuzzy matching when used together with EDDR PM.

2. DEANGLE

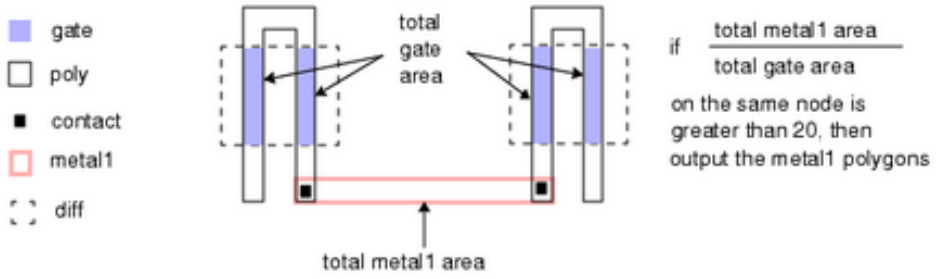


Figure 3.25: NET AREA RATIO operation. NET AREA RATIO metal1 gate > 20. Figure from [24]

DEANGLE operation replaces skewed edges with orthogonal edges. This operation may be used to remove all skewed edges before member creation so that EDDR PM can handle a complicated pattern having many skewed edges.

3. NET AREA

   NET AREA operation selects all polygons that lie on an electrical node (on the same net) and calculate the area of them to decide whether it meets NET AREA constraints or not. It outputs the polygons that meet the constraints.

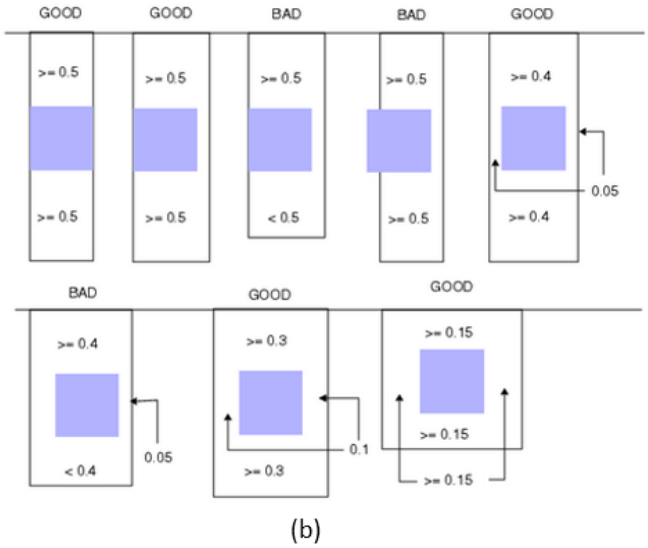
4. NET AREA RATIO

   NET AREA RATIO operation does a similar job to NET AREA. The difference between those is that NET AREA RATIO calculates a ratio of polygon areas from two or more layers. This operation is most promising for multi-layer pattern matching. Figure 3.25 is an example of this operation.

5. RECTANGLE ENCLOSURE

```
RECTANGLE ENCLOSURE contact metal
          GOOD 0.00  0.50  0.00  0.50
          GOOD 0.05  0.40  0.05  0.40
          GOOD 0.10  0.30  0.10  0.30
          GOOD 0.15  0.15  0.15  0.15
```

(a)



(b)

Figure 3.26: RECTANGLE ENCLOSURE operation. (a) operation with left,top,right, and bottom constraint. (b) possible results from the operation (a). Figure from [24]

RACTANGEL ENCLOSURE operation checks enclosureness of rectangles. It is used for efficient enclosure checking of rectangles as shown at Figure 3.26. This DRC operation has a potential to work well for multi-layer pattern matching.

6. INSIDE

This operation selects all polygons that are inside of polygons from another layer. This operation may be used for multi-layer pattern matching with EDDR PM as well. Figure 3.27 shows an example of this operation.

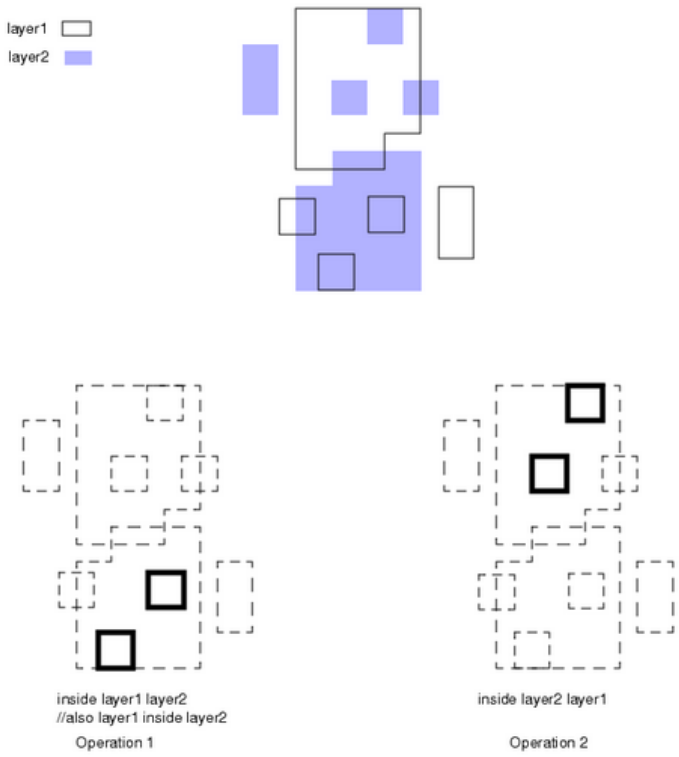


Figure 3.27: Operation 1 selects all layer1 polygons that lie completely inside any layer2 polygons (this includes coincident edges). Operation 2 reverses the layer order; therefore, it selects all layer2 polygons that lie completely inside layer1 polygons (again, this includes coincident edges). Figure from [24]

3.6.3 EDDR PM Conclusion

In this section, we presented a novel methodology for fast and accurate pattern matching. Along with the idea of employing super-fast DRC edge operations to do edge driven dissected rectangles pattern match, we showed how to utilize the mathematical vector concept to avoid the unnecessary 8 iterations for detecting the same pattern in 8 different orientations.

We also presented possible applications of our approach such as fuzzy match and partial match. Our results show that our approach achieves 100% accurate match and it is significantly faster than other methods. Since member rectangle

creation is based on simple DRC edge operations, it is not only fast and easy to describe a pattern but also it enables us to perform fuzzy pattern matching and partial matching efficiently. The flexibility of EDDR PM for fuzzy matching and partial matching comes from its concept of members. Depending on whether we allow non-members or not, the degree of partial matching can be different. Similarly, depending on how to put constraints or where to put constraints on member creation, the degree of fuzzy pattern matching can vary. The beauty of this member concept for hotspot detection is it is so simple that we can apply this concept easily to other pattern matching applications.

We showed that this technique can be easily adapted in multi-layer pattern matching. For example, if a pattern includes several layers such as metal_1, via, and metal_2, we can use edge operations not only in metal_1 layer but also between those layers to derive member rectangles that match the configuration of those layers and perform EDDR_PM.

We discussed future work by utilizing additional DRC operations such as AREA, DEANGLE, NET AREA, NET AREA RATIO, RECTANGLE ENCLOSURE, and INSIDE for pattern matching.

Algorithm 3 EDDR PM Fuzzy Match

```
1: procedure EDDR–PM(P1, P2, ...., Pn, nonMem, PDB)
2:     Inputs (P1..n, nonMem, and PDB) are the same as Algorithm 1.
3:     ADD_BIN for each member from P1 to Pn.
4:     LOCATE_BIN for all the origin members of P1 and get bin_counts
5:     for i = 1 → bin_counts for p_i in P1 bins  do
6:         LOCATE_BIN a reference member, p2, in P2 bins by searching
7:         the vector distance between P1's center and P2'2 center.(PDB has this
   info.)
8:         if found then
9:             if the found p2(reference member) is fuzzy member then
10:                Create one of 8 bounding boxes using 8 different d1, d2, d3,
11:                and d4 sets stored in PDB.
12:                Iterate 8 times from step(19) to step(35).
13:            else
14:                if the found p2 a valid reference member then
15:                    Create a bounding box using d1, d2, d3, and d4 determined
   by
16:                    vector info between P1 and the found P2's center.
17:                else
18:                    No match. continue to next p1 in P1
19:            if nomMem exists inside the bounding box then
20:                No match. continue to next p1 in P1
21:            LOCATE_BIN for other members in P3 · · · P_n
22:            inside the bounding box.
23:            for each member inside the bounding box do
24:                n = number of each member inside bounding box
25:                m = number of each member described in PDB
26:                if n != m then
27:                    No match. continue to next p1 in P1
28:                if member is fuzzy member then
29:                    skip member validation check
30:                else
31:                    if valid member == false then
32:                        No match. continue to next p1 in P1
33:            Matching pattern found at this point.
34:            Output the bounding box to indicate the match.
35:            continue to next p1 in P1
36:        else
37:            No match. continue to next p1 in P1
```

4

Litho-aware Machine Learning Based Hotspot Detection

In this chapter, we propose a novel methodology for machine learning (ML) based hotspot detection that uses lithography information to build SVM (Support Vector Machine) during its learning process. Unlike previous researches that use only geometric information or require a post-OPC (Optical Proximity Correction) mask, this proposed method utilizes detailed optical information but bypasses post-OPC mask by sampling latent image intensity and use those points to train an SVM model. The results suggest high accuracy and low false alarm, and faster runtime compared with methods that require a post-OPC mask.

There are two major machine learning algorithms: Supervised machine learning and Unsupervised machine learning. All of ML-based hotspot detection approaches use supervised one since training data sets are categorized into two, hotspots and non-hotspots, and we are not drawing inferences from the data sets which unsupervised machine learning can do, but we want to identify hotspots using what ML has learned from the past data sets where supervised machine learning can be applied. More information about machine learning is summarized in 4.1 and 4.2.

4.1 Supervised Machine Learning

Supervised learning discovers patterns in the data with a target (class) attribute, which means the training data is labeled data. For example, data set of hotspot with 1 and non-hotspot with -1 can be learned to predict new data's attribute. In

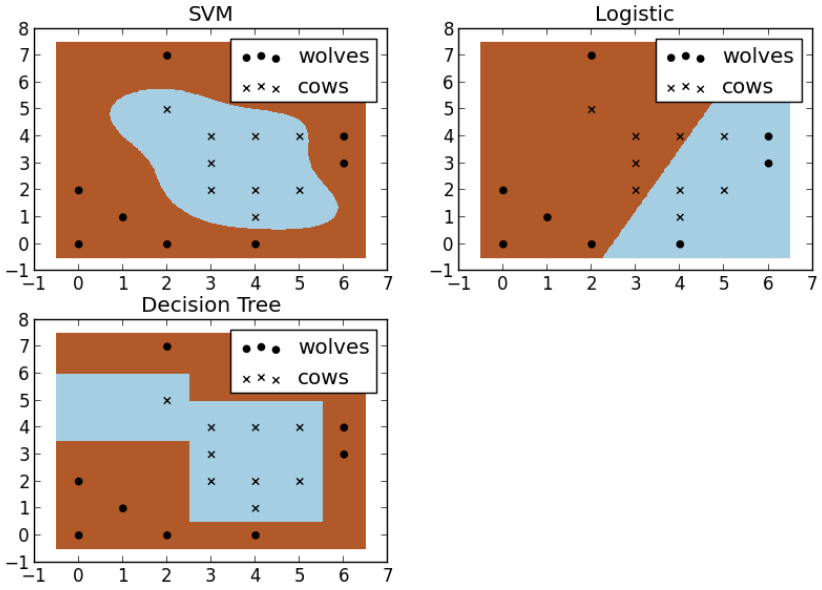


Figure 4.1: Non-linear SVM Kernel vs linear classifiers from [36]

other words, It learns using labeled data (class) to classify new data into a proper class.

There are many classifiers such as K-NN classifier [1], Perceptron classifier [21], SVM (Support Vector Machine) classifier [37, 67, 69], Logistic classifier [16], Decision Tree classifier [47] and etc. Among those, SVM is widely adopted because it can capture complex relationships between training data points. SVM uses a technique called Kernel trick (4.1.5) to transform data in order to find an optimal boundary classifying the data with maximum margin. If SVM's kernel is non-linear, the separation boundary (hyperplane) is also non-linear. Figure 4.1 shows this benefit of SVM.

The basic idea in SVM is to find the optimal separating hyperplane by maximizing the margin between classes' closest points known as support vectors which are shown in Figure 4.2. In this example, linear Kernel is used since training

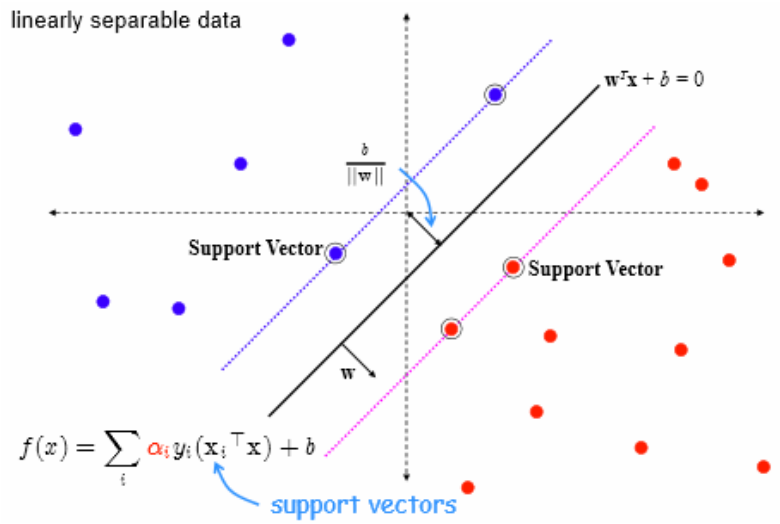


Figure 4.2: Support Vectors from [89]

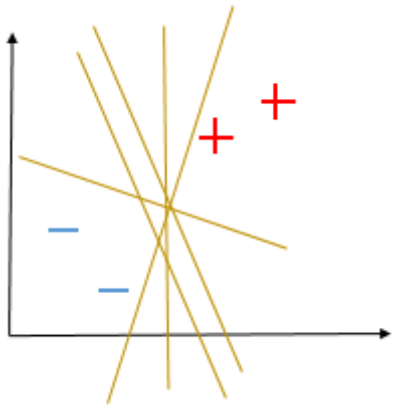


Figure 4.3: Arbitary separation lines

data is linearly separable.

4.1.1 Support Vector Machine

There are many separation boundaries, for an example of 2D as shown Figure 4.3, we can draw to separate data. SVM can pick the best boundary with maximum margin. Let's see how it works with the simple case. Figure 4.4 has minus samples and plus samples. The decision rule for the unknown input, $\vec{u}$, is $\vec{w}\vec{u} + b \geq 0$ if

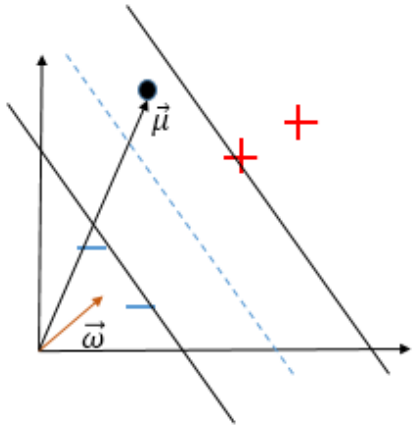


Figure 4.4: plus samples and minus samples; dashed line is a decision boundary.

the unknown is a plus sample where $\vec{w}$ is a perpendicular vector to the separation boundary dashed line.

If we set +1 for plus samples and -1 for minus samples, we can write the equation for plus samples and minus samples such as 4.1 and 4.2.

$$\vec{w}\vec{x}_+ + b \geq 1 \tag{4.1}$$

$$\vec{w}\vec{x}_- + b \leq -1 \tag{4.2}$$

In addition, for mathematical convenience, let's have a function y_i such that $y_i = +1$ for plus samples and $y_i = -1$ for minus samples. And then the equation 4.1 and 4.2 becomes one merged equation as 4.3.

$$y_i(\vec{w}\vec{x}_i + b) - 1 \geq 0 \tag{4.3}$$

SVM tries to maximize the width shown at Figure 4.5, which can be calculated by the equation, 4.4.

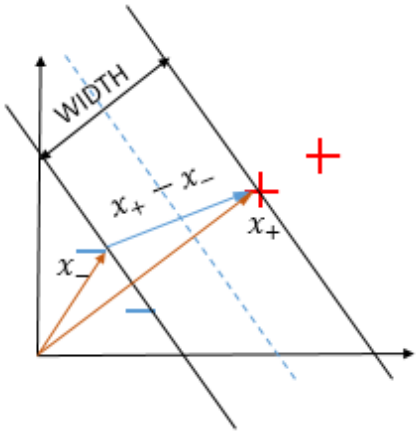


Figure 4.5: Maximizing width between solid line is the goal of SVM.

$$WIDTH = (x_+ - x_-)\frac{\vec{w}}{\| \vec{w} \|} \tag{4.4}$$

Since $\vec{w}\vec{x_+}$ is 1 - b and $\vec{w}\vec{x_-}$ is 1 + b according to the equations, 4.1 and 4.2. it becomes the equation, 4.5.

$$WIDTH = \frac{2}{\| \vec{w} \|} \tag{4.5}$$

Therefore, for maximum width, we need to minimize $\| \vec{w} \|$. Let's write it such as $\frac{1}{2}\| \vec{w} \|^2$ for mathematical convenience and our goal is to minimize it. When we have a minimization goal with a constraint function such as the equation, 4.3, the method of Lagrange multipliers can be used to find a solution.

Now, our goal is to find Lagrange multipliers of α_i in the equation of 4.6.

$$L = \frac{1}{2}\| \vec{w} \|^2 - \sum_i^N \alpha_i(y_i(\vec{w}\vec{x_i} + b) - 1) \tag{4.6}$$

Since we need to find maximum location of the function, 4.6, there are two

partial derivatives, 4.7 and 4.8, to be set as zero.

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum_{i}^{N} \alpha_i y_i \vec{x_i} = 0 \tag{4.7}$$

$$\frac{\partial L}{\partial b} = -\sum_{i}^{N} \alpha_i y_i = 0 \tag{4.8}$$

Therefore, $\vec{w} = \sum_{i}^{N} \alpha_i y_i \vec{x_i}$ which means $\vec{w}$ is a linear sum of all samples. And we can obtain the equation, 4.9, by plugging 4.7 and 4.8 into 4.6.

$$L = \sum_{i}^{N} \alpha_i - \frac{1}{2} \sum_{i}^{N} \sum_{j}^{N} \alpha_i \alpha_j y_i y_j \vec{x_i} \vec{x_j} \tag{4.9}$$

This is called Dual form to solve the quadratic programing SVM is dealing with. Note that maximization of the equation, 4.9, depends only on sample's dot product. Once all α_i is calculated, we can use the decision function, 4.10, to decide whether an unknown sample of $\vec{u}$ is a plus sample or minus sample.

$$D = \sum_{i}^{N} \alpha_i y_i \vec{x_i} \vec{u} + b \begin{cases} \text{plus sample} & \text{if } D \geq 0 \\ \text{minus sample} & \text{if } D < 0 \end{cases} \tag{4.10}$$

4.1.2 Linear classifier

A linear classifier has the form (4.11) where **w** is weight vector which is the normal to the line, and b is the bias. The separation is a line in 2-dimensional space. In 3D, it is a plane, and in nD, it is a hyperplane. Figure 4.6 graphically shows this.

$$f(x) = \mathbf{w}^\top x + b \tag{4.11}$$

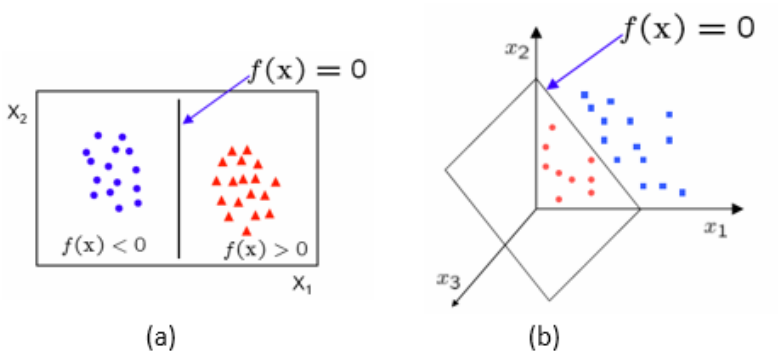


Figure 4.6: Separation in 2D and 3D; (a) 2D (b) 3D

Given training data (x_i, y_i) for $i = 1, \ldots, N$ with $x_i \in \Re^d$ and $y_i \in \{-1, 1\}$, a classifier 4.12 can be learned such that $y_i f(x_i) > 0$ always.

$$f(x_i) = \begin{cases} \geq 0 & \text{if } y_i = 1 \\ < 0 & \text{if } y_i = -1 \end{cases} \tag{4.12}$$

For an SVM learning a linear classifier 4.11, it is an optimization problem over $\mathbf{w}$ as following equation 4.13:

$$\begin{gathered} max_w \left(\frac{2}{\| w \|} \right) \\ subject\ to\ \mathbf{w}^\top x_i + b \begin{cases} \geq 1 & \text{if } y_i = 1 \\ \leq -1 & \text{if } y_i = -1 \end{cases} \quad for\ i = i \ldots N \\ Or,\ equivalently \\ min_w \left(\| w \| \right)^2 \\ subject\ to\ y_i \left(\mathbf{w}^\top x_i + b \right) \geq \ for\ i = 1 \ \ldots \ N \end{gathered} \tag{4.13}$$

This is a quadratic optimization problem subject to linear constraints. To maximize margin while sacrificing training error in a reasonable range depending

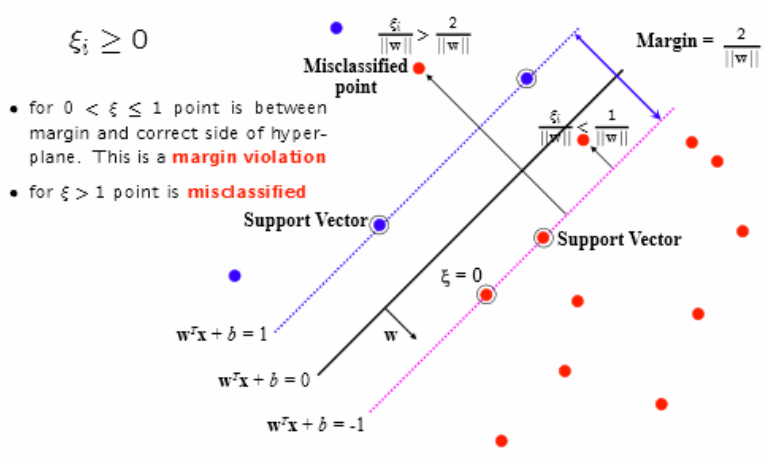


Figure 4.7: Slack variable [89]

on the application of SVM, a slack variable is introduced as explained at Figure 4.7.

Now, the optimization problem becomes as following:

$$\min_{\mathrm{w}} \left(\| w \|^2 + C \sum_{i}^{N} \xi_i) \right)$$

$$\text{where } y_i \left(w^\top x_i + b \right) \geq 1 - \xi_i \text{ for } i = 1, \ldots, N$$

which can be expressed as: 4.15

C is a parameter that controls trade-off between margin and training error. Small C allows large margin but increases the error. Large C decreases margin, but it allows small error.

The quadratic optimization problem 4.15 is known as the primal problem. We can derive the dual problem 4.17 from it by Representer Theorem that states the solution w can always be written as a linear combination of the training data:

$$w = \sum_{j}^{N} \alpha_j y_j x_j$$

The advantage of solving the dual form rather than the primal form is that it is much more efficient when a number of input data is much greater than a number of dimensions. It also enable us to use flexible SVM kernel for better classification when dealing with complex training data.

4.1.3 Primal and Dual

Primal version of classifier is:

$$f(x) = w^\top x + b \tag{4.14}$$

Primal optimization problem over w is:

$$\min_{\mathrm{w}} \left(\parallel w \parallel^2 + C \sum_i^N max(0, y_i f(x_i)) \right) \tag{4.15}$$

for $w \in \Re^d$ where d is dimension of feature vector x.

Dual version of classifier is:

$$f(x) = \sum_i^N \alpha_i y_i (x_i^\top x) + b \tag{4.16}$$

Dual optimization problem over α is:

$$max \sum_i^N \alpha_i - \frac{1}{2} \sum_{jk}^N \alpha_j \alpha_k y_j y_k (x_j^\top x_k) \tag{4.17}$$

for $\alpha \in \Re^N$ subject to $0 \leq \alpha_i \leq C$ for $\forall i$

and $\sum_i^N \alpha_i y_i = 0$ where N is number of training points.

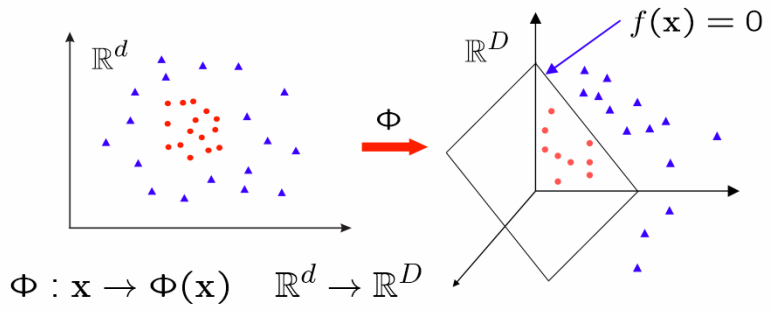


Figure 4.8: Feature map [89]

4.1.4 Feature map

When data is not linearly separable in a certain dimension, for an example of 2D, we can map the data to a higher dimension in order to make it linearly separable. It is called feature mapping which transforms data feature space. It is depicted at Figure 4.8.

With feature mapping, classifier in w for $\Re^D$ becomes $f(x) = w^\top \Phi(x) + b$ where $\Phi(x)$ is feature map.

Dual calssifier 4.16 in transformed feature space becomes:

$$f(x) = \sum_i^N \alpha_i y_i \Phi(x_i)^\top \Phi(x) + b \tag{4.18}$$

Dual optimization problem over α 4.17 becomes:

$$max \sum_i^N \alpha_i - \frac{1}{2} \sum_{jk}^N \alpha_j \alpha_k y_j y_k \Phi(x_j)^\top \Phi(x_k) \tag{4.19}$$

$$\text{for } \alpha \in \Re^N \text{ subject to } 0 \leq \alpha_i \leq C \text{ for } \forall i$$

$$\text{and } \textstyle\sum_i^N \alpha_i y_i = 0 \text{ where } N \text{ is number of training points.}$$

4.1.5 Kernel trick

By writing $k(x_i, x_j) = \Phi(x_i)^\top \Phi(x_j)$, 4.18 and 4.19 can be written as:

$$f(x) = \sum_i^N \alpha_i y_i \kappa(x_i)\kappa(x) + b \tag{4.20}$$

$$max_\alpha \sum_i^N \alpha_i - \frac{1}{2} \sum_{jk}^N \alpha_j \alpha_k y_j y_k \kappa(x_j)\kappa(x_k) \tag{4.21}$$

$$\text{for } \alpha \in \Re^N \text{ subject to } 0 \leq \alpha_i \leq C \text{ for } \forall i$$

$$\text{and } \textstyle\sum_i^N \alpha_i y_i = 0 \text{ where } N \text{ is number of training points.}$$

With Kernel trick, we can train SVM models on complex data which may require non-linear classification. For example, we can use Gaussian kernel 4.22 to support infinite dimensional feature space.

$$\kappa(x, x\prime) = e^{\frac{-\|x-x\prime\|^2}{2\sigma^2}} \; for \; \sigma > 0 \tag{4.22}$$

Therefore, we can write Radial Basis Function SVM like below.

$$f\left(x\right) = \sum_i^N \alpha_i y_i e^{\frac{-\|x-x_i\|^2}{2\sigma^2}} + b, \; b : bias \tag{4.23}$$

4.2 Unsupervised Machine Learning

Training data for unsupervised learning does not have a target attribute (class). As the name suggests, input data is not labeled so that learning process is not supervised. During training stage, unsupervised learning tries to find intrinsic

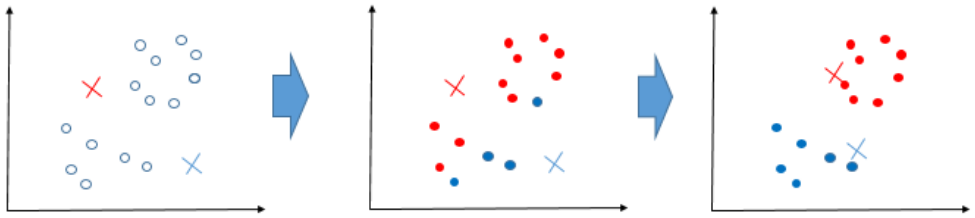


Figure 4.9: K-means algorithm: K is 2. Two centroids, red cross and blue cross

pattern inside data and cluster the data. Based on this learning, it can put any new input data in an appropriate cluster. Clustering approaches of unsupervised learning include k-means [34], mixture models [72], hierarchical clustering [3, 53] and etc. Since we use labeled data for hotspot detection, we don't pay much attention to unsupervised learning for our thesis. This section is just a brief summary of major unsupervised learning models for reference.

4.2.1 K-means

When classifying unlabeled data into clusters, you first decide how many clusters, K, you need to generate. Once K is decided, K-means algorithm creates K number of seeds, named cluster centroids, at random location inside of the data to start with. It calculates distances from the centroids, where the distance metric can be any user-defined metric in feature space, and assign each data point to the closest centroid. After this clustering, it calculates the average point of each cluster and moves centroids to the mean point, and it starts the process again until it meets some stopping criterion which can be the total number of iteration or no more cluster assignments change. 4.9 depicts K-means algorithm graphically.

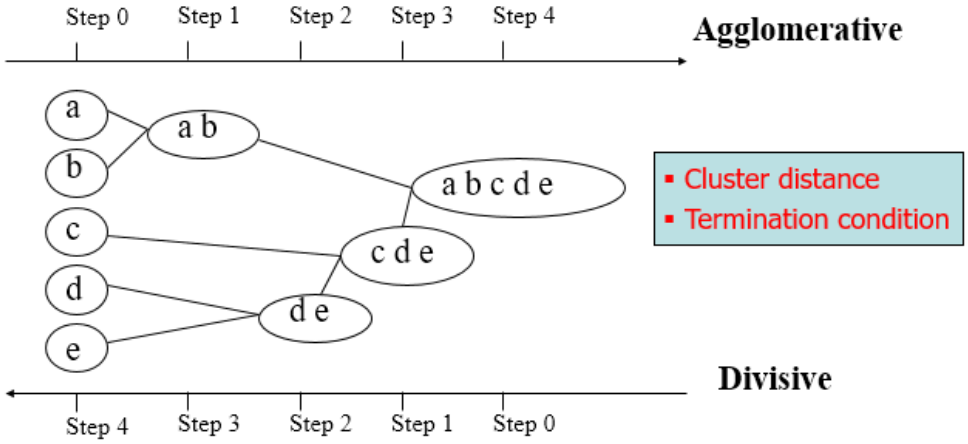


Figure 4.10: Hierarchical clustering. Figure is from [7]

4.2.2 Hierarchical clustering

Using a distance metric (a measure of distance between pairs of samples), hierarchical clustering algorithm combines samples together (agglomerative or bottom-up) or split samples (divisive or top-down) hierarchically and generate tree-like clusters. There is no need to know the number of clusters in advance. 4.10 illustrates hierarchical clustering process.

4.3 Background

As feature sizes in chip design and semiconductor manufacturing technology node scale down further, the industry is being faced with a great challenge to cope with the sub-wavelength lithography gap [22]. Even with various sophisticated resolution enhancement techniques (RETs), multiple pattern lithography (MLP), and design for manufacturing (DFM), semiconductor manufacturing process will often run into lithography hotspots which produce pinching and bridging errors.

Lithography simulation has been used to find hotspots because it can be very

accurate [19, 29, 30]. However, it is too computationally expensive for full-chip scale.

To tackle this problem, there have been some alternative hotspot detection approaches. Pattern matching and machine learning based techniques are mainly adopted to reduce high computational complexity and to enable design verification at the early design phase.

Pattern matching approach can be categorized into three areas. String-based pattern matching [2, 8], DRC-based pattern matching [11, 18, 23, 63], and explicit model-based pattern matching [79, 83, 86].

Even though these pattern matching techniques have shown reasonable success and been employed by the industry as alternative hotspot detection to avoid the expense of rigorous lithography simulation, they have an significant shortcoming: It is difficult to recognize previously unseen hotspots. Hotspot pattern library for pattern matching cannot cover all possible hotspots even if fuzzy pattern matching is adopted.

In contrast, Machine Learning (ML) is capable of identifying previously unseen hotspots when it is well trained, and the characterization vector is relevant to the problem, but just as pattern matching approaches have their strength and weakness, ML also has limitations. False alarms are inevitable, and therefore, it is critical to create an optimal model and develop methods to reduce the false alarm rate.

ML-based approaches for hotspot detection have typically used a supervised learning model, e.g., artificial neural network (ANN) [50] or support vector model (SVM) [43]. Recent research involves hierarchical learning [49], data clustering [53], fuzzy cluster growing [83], and topological classification and critical feature

extraction [9].

All of these ML-based hotspot detections suffer from false alarms when they try to achieve higher accuracy of real hotspot detection. Figure 15 of [9] shows their experimental data to point out the tradeoff between accuracy and false alarm.

To address this issue with a ML-based approach for hotspot detection, we propose in this paper to use lithography information and lithographic related domain knowledge for machine learning. As well explained in [14], prior knowledge (Domain knowledge) plays a crucial role to have machine learning trained as accurate as possible. In addition to training example, we have to select features which are especially informative for the training.

In this chapter, we propose to use aerial image intensity information produced by the same illumination as chip manufacturing process. We also propose to select features based on hotspot type such as bridging and pinching. Furthermore, if the illumination is asymmetric, those two hotspot types split into four types: Horizontal bridging, vertical bridging, horizontal pinching, and vertical pinching. Therefore, we create four SVM kernels that are trained with aerial image intensity information. We run those four kernels, and each kernel will serve us to find hotspots in design. It is important to note that the simulation of these intensity points is obtained by using the drawn (design target) structures, thus eliminating the need of a post-OPC mask.

Our test experiment result against 2012 CAD contest at ICCAD shows almost 100% accuracy and low false alarm rate. In fact, it shows our approach outperforms all other previous works by a significant margin. Our contributions are briefly summarized as follows:

1. We propose a robust and accurate SVM kernel training method utilizing real

lithography illumination information and domain knowledge of lithographic failure type.

2. We present experiment results demonstrating high accuracy with low false alarm rate, which means it overcomes the biggest drawback of ML-based hotspot detection.

The remaining section is organized as follows. Section 4.4 describes the problem presented at 2012 ICCAD contest. Section 4.5 explains our approach in detail. Section 4.6 shows experimental data comparing the contest winner and [9]. We conclude our ML work in Section 4.7.

4.4 PROBLEM at 2012 ICCAD CONTEST

In 2012 CAD contest at ICCAD, fuzzy pattern matching problem for physical verification to detect previously unseen hotspots was given to contestants [64]. They were given a training data set of hotspots and non-hotspots patterns in a layout for training. They were also given a testing layout which has previously unseen hotspots for testing their methods. Maximizing accuracy was the primary goal while minimizing false alarm rate was a secondary goal.

Figure 4.11 shows hotspot or non-hotspot pattern in the training data set. It is a layout clip with a core and its ambit. A core is an area that has expanded from the center of a hotspot or non-hotspot location. The amount of expansion is determined by interaction distance of optical illumination, which is 0.6um from the center in this contest making 1.2 um square box. Ambit is a peripheral part of the clip. Usually, these hotspot and non-hotspot data set are provided by foundry or lithography simulation.

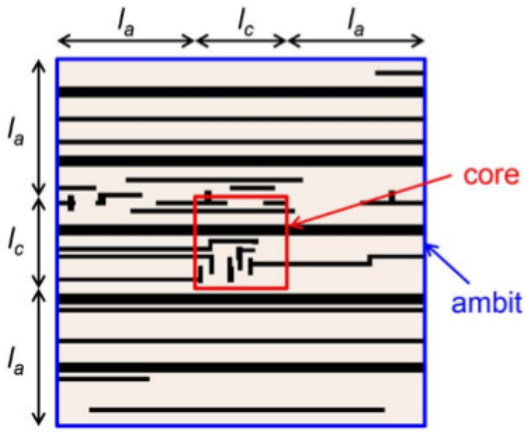


Figure 4.11: Hotspot or non-hotspot pattern configuration. Figure from [64]

Definition 4.4.1. A hotspot is a layout pattern that may induce printability issue in lithography process.

Definition 4.4.2. A hit is a correctly detected hotspot when core of actual hotspot is inside the hit region shown as Figure 4.12.

Definition 4.4.3. Accuracy is the ratio of the hits over the actual hotspots in total.

Definition 4.4.4. An extra is a non-hotspot detected as a hotspot.

Definition 4.4.5. False alarm is the ratio of the extras over the testing layout area.

An identified hotspot is considered as a hit when the core of the hotspot interacts with the core of an actual hotspot, which means there must be at least overlapping area between the two cores.

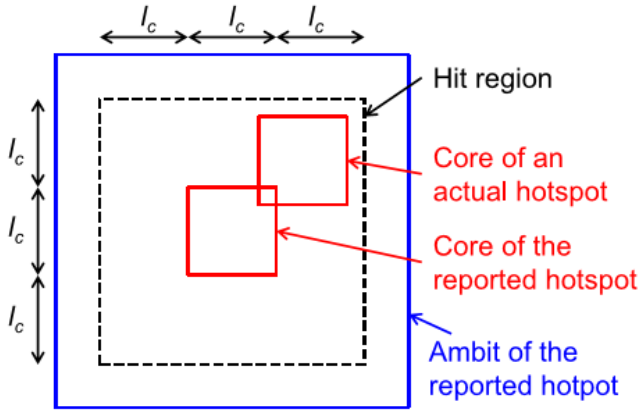


Figure 4.12: Hit means acutal hotspot is inside of the Hit region. Figure from [9]

4.5 Litho-aware Machine Learning

4.5.1 Prepare Training Data

In our approach, we categorize hotspots to two cases as we know that there are two critical failures in transferring layout on the mask to wafer: Bridging and Pinching. Bridging may happen when two nearby polygons are too close, whereas pinching may occur when polygon dimension is too small. We also further separate those two to four cases for machine learning process. When the illumination is not symmetric, we have to take into account whether it is horizontal or vertical failures, which leads to four hotspot types: Horizontal bridging (HB), Vertical bridging (VB), Horizontal pinching (HP), and Vertical pinching (VP). Figure 4.13 shows the four hotspot types.

Since asymmetric Quadruple illumination shown at Figure 4.14 was used to generate hotspots in the training data set at the 2012 ICCAD contest, we categorize all hotspots into the four categories. It is because asymmetric illumination has a different image formation impact on horizontal line/space versus vertical line/space. After we categorize all hotspot types, we train four

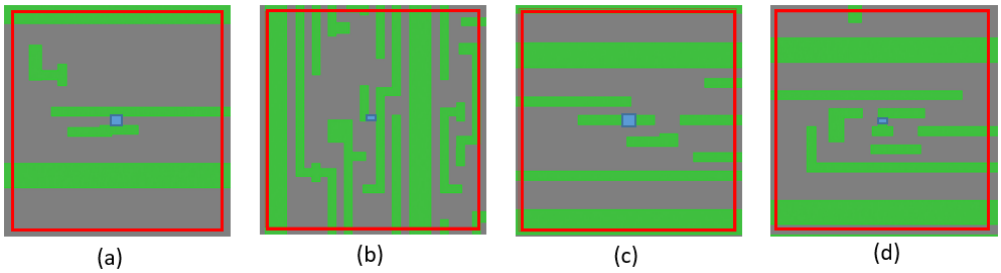


Figure 4.13: Four types of hotspots: (a) HB, (b) VB, (c) HP, (d) VP; Red boxes are cores. Box at the center of a core is hotspot location.

Figure 4.14: Asymmetric Quadruple illumination used for hotspot generation at 2012 ICCAD contest.

SVMs respectively using aerial image intensity information produced by the same asymmetric Quadruple illumination.

Aerial image Intensity is calculated on 6% attenuated mask which is used for generating hotspots in the training data. 12 points are selected at the center of a core for intensity to be used for SVM. The distance between the points is 3 or 5 nm depending on the technology node (28nm or 32nm). The number of points is decided to ensure that the intensity line crosses at least minimum width or space. In our case, it is 28nm or 32nm. Those selected points are lined vertically or horizontally from the center of the core. Figure 4.15 shows it graphically. During the training phase, we decide which direction is best for each SVM (HB, VB, HP, and VP SVM). During cross-validation using training data, we choose the best

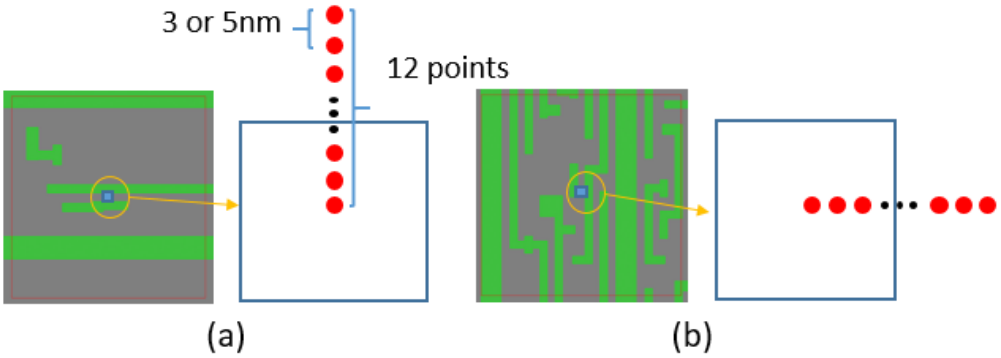


Figure 4.15: Simulation points. Distance between points is 3 nm for 28nm design and 5 nm for 32 nm design; (a) vertical line of points; (b) horizontal line of points

one.

4.5.2 Prepare hotspot candidates

Each trained SVM is applied to a testing layout to find hotspots. Since we know that bridging and pinching occurs mostly at locations with a minimum width or minimum space of layout design, we generate possible locations for hotspot candidates only at those locations.

By generating hotspot candidates this way, we dont́ check every location in the design. We only check those hotspot candidates, which reduces the number of locations to check and improves run time of hotspot detection. Besides, we also categorize those candidates into the four hotspot types: HB, VB, HP, and VP hotspot candidates. Figure 4.16 shows examples of those four hotspot candidate types on which each SVM will run respectively.

As shown in Figure 4.16, a long line is broken into several pieces to check along the line. Otherwise, we end up checking only one place in the long line which is a center of the line. It is necessary to break up a long line because bridging or pinching may happen somewhere along the line. The distance between hotspot

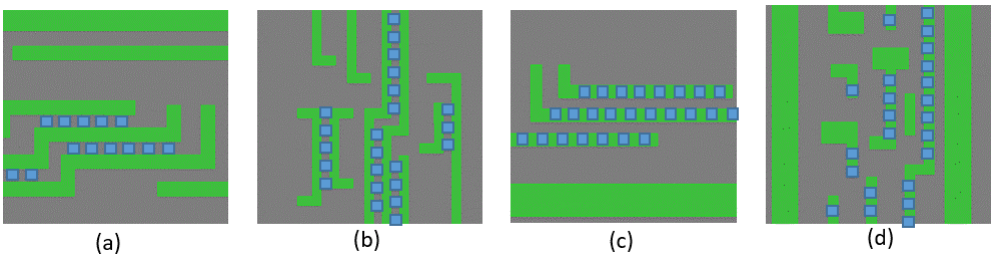


Figure 4.16: Hotspot candidates; (a) HB candidates, (b) VB candidates, (c) HP candidates, (d) VP candidates

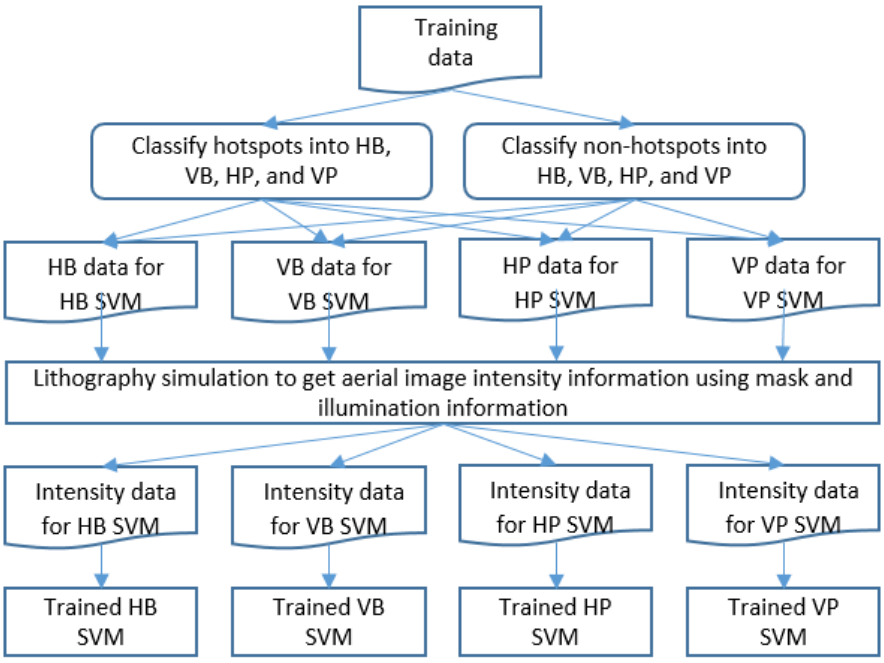
checking points is 45 nm for 28 nm design and 50 nm for 32 nm design. These distances are chosen based on a minimum width of the testing layouts.

For most technology nodes, it is well known that optical proximity effect is about 0.6 um in radius from one location. The effect, however, gets reduced exponentially as it goes out further from the location. And the ripple effect along a line causing pinching or bridging is directly related to edge fragmentations for OPC. Since we dont́ know edge fragmentation scheme for OPC which is applied to the test layouts to generate hotspots, we chose minimum width as the distance between hotspot checkpoints along a line trying not to miss a hotspot between those points. Note this distance is not decided by analytical quantification. This is a limitation of our approach and it should be further studied.

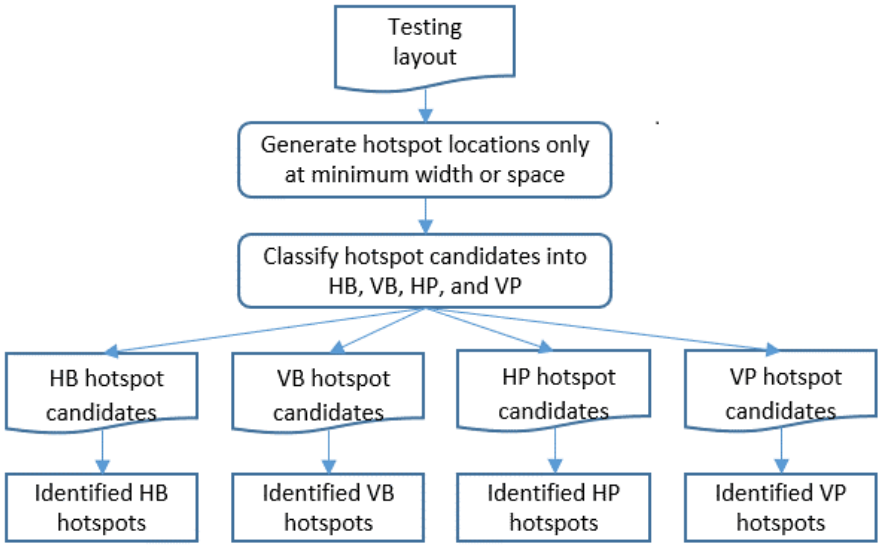
Figure 4.17 shows the framework of our lithography-aware ML for hotspot detection. Most important data during SVM training stage is illumination, mask, and hotspot type information.

4.5.3 Supervised Machine Learning

We adopted C-type Support Vector Machine (SVM) which supports binary classification: hotspot and non-hotspot in our case. Binary classification SVM transforms training data to a high-dimensional feature space and produces a



(a)



(b)

Figure 4.17: Our hotspot detection framework; (a) Training flow, (b) Testing flow

decision function (classifier) to separate the data into two classes with a maximum margin. SVM is chosen among many supervised learning models since it has shown excellent performance in handling a small nonlinear data set [66, 67], which is an important aspect considering there are usually a small number of hotspots for training.

If SVM kernel function is a symmetric positive semi-definite function, it is guaranteed to have a global optimum solution. We chose radial basis function as our SVM kernel for that reason.

C-type SVM solution to derive a decision function can be reduced to the solution of the dual form of quadratic programming, which is given as follows.

$$\max\ (f(\alpha)) = \sum_{i}^{N} \alpha_i\ - \frac{1}{2} \sum_{jk}^{N} \alpha_j \alpha_k y_j y_k k(x_j, x_k)$$

$$\text{Subject to } 0 \leq a_i \leq c\ for\ \forall i$$

$$\sum_{i}^{N} \alpha_i y_i = 0$$

$$k(x_j,\ x_k) = e^{\frac{-\|x - x_i\|^2}{2\sigma^2}}$$

$$\alpha = (\alpha_1,\ \ldots, \alpha_N)^T$$

Decision function (radial basis function SVM):

$$f\,(x) = \sum_{i}^{N} \alpha_i y_i e^{\frac{-\|x - x_i\|^2}{2\sigma^2}} + b,\ b : bias$$

N is size of training data and γ is N dimensional vector to be learnt. Given training data x_i, i=1, ...,N, we have a binary label function $y_i \in \{1, -1\}$, 1 for

hotspot and -1 for non-hotspot in our usage. Note that Gaussian Kernel, k, is used, which is a symmetric positive semi-definite function. C controls the trade-off between classification accuracy and the norm of the decision function. The slack variable, γ which is $1/2\sigma^2$ portion of the Gaussian kernel, is to handle non separable data.

Setting an appropriate C and γ is critical to have a good training result. Therefore, we performed 3-fold cross-validation [35] during iterations with different C and γ each time to pick up the best decision function for each SVM kernel. The initial value of C is 1 and γ is 0.0001. The iteration loop is nested with these two variables. Outer loop is γ stepping 5 times previous value until γ becomes 1. Inner loop is C stepping 5 times previous value until it reaches 100,000. In short, we perform 64 times of the cross validation and pick the best C and γ.

Since there are a small number of hotspots and a large number of non-hotspots in training data set, we balanced those two numbers to avoid degradation of the training quality. Otherwise, the imbalanced number between hotspot and non-hotspots can destroy the soft margin leading to a poor training result. We rebalanced it by including only the same number of non-hotspots as the number of hotspots in the training data. We discarded the rest of non-hotspots during SVM training.

Another thing important to consider for a high quality of trained model is how to sample non-hotspots data [88]. Since most hotspots occur at a minimum width or space, the non-hotspot data set should also be generated at minimum a width or space. We generated those locations for non-hotspot data set. Also, there should be a suitable metric to sample appropriate ones among non-hotspots. This metric should draw a clear line between hotspots and non-hotspots. We used

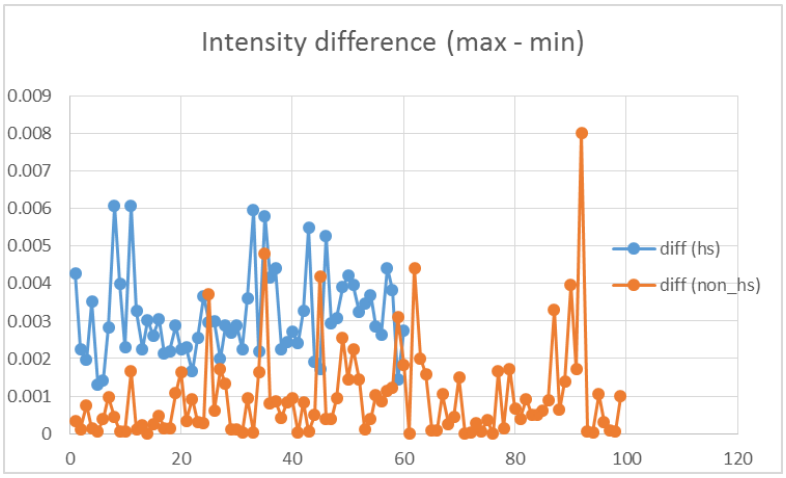


Figure 4.18: Metric to select non-hotspots: Maximum and minimum intensity difference less than 0.001 is used to select non-hotspots.

maximum and minimum intensity difference in the 12 intensity calculation points as the metric to select non-hotspots as our training data set. Figure 4.18 shows an example of how the intensity difference space look like for hotspots and non-hotspots. In this example, we select sixty non-hotspots that have the intensity difference less than 0.001.

4.6 Experimental Result

We implemented our approach in C++ programming language with SVM library LIBSVM [37]. Our experiments were performed on a Linux platform with 3.7 GHz clock CPU and 32 GB RAM. Six industrial designs, two 32 nm and four 28 nm designs which were provided by 2012 CAD contest [64], were used for our experiment in order to have fair comparisons. Table 4.1 from [9] shows a list of the designs.

Table 4.2 summarizes our experimental results against 2012 CAD contest winners, [83] and [9]. As it shows, our approach achieves the highest accuracy

Table 4.1: 2012 CAD Contest at ICCAD Benchmark Statistics

Training data			Testing layout			
Name	#hs	#nhs	Name	#hs	area (um^2)	process
MX_benchmark1_clip	99	340	Arrary_benchmark1	226	12,516	32nm
MX_benchmark2_clip	176	5,285	Arrary_benchmark2	499	106,954	28nm
MX_benchmark3_clip	923	4,643	Arrary_benchmark3	1,847	122,565	28nm
MX_benchmark4_clip	98	4,452	Arrary_benchmark4	192	82,010	28nm
MX_benchmark5_clip	26	2,716	Arrary_benchmark5	42	49,583	28nm
			MX_blind_partial	55	224,975	32nm

#hs: number of hotspots; #nhs: number of non-hotspots. The core size is 1.2 x 1.2um^2, while the clip size is 4.8 x 4.8um^2

among other previous works, ranging from 97.40% to 100% across the entire testing layouts. As mentioned before, accuracy is primary criteria for a winner while the false alarm is secondary. Since missing one hotspot kills a design, the primary object is accuracy.

False alarm result of our approach is also excellent considering high accuracy achievement. As pointed out with the experimental data in Figure 15 of [9], there is a huge tradeoff between accuracy and false alarm especially when trying to exceed 95% accuracy. However, our result shows we minimized the tradeoff achieving high accuracy with low false alarm. In fact, if we look at the data of **"MX_blind_partial"**, our approach produced 100% accuracy with the lowest false alarm. It is important data point because the test design is a layout as a whole while the others testing designs consist of a collection of clipped layouts, which means our approach outperformed on a real design environment.

4.6.1 Multi-layer hotspot detection

Multi-layer hotspots are mainly caused by overlay issues between layers. As explained at the section, 3.6.1, EDDR PM is capable of identifying those hotspots.

Table 4.2: Comparison with 2012 CAD contest winner, [87], [81], [83], and [9]

Testing layout(Training data)	Methods	#hit	#extra	accuracy	false alarm (#xtra /area)	run time	area(um^2)
Array_benchmark1 (MX_benchmark1_clip)	1st place	212	1,826	93.81%	0.15	0m05.1s	12,516
	[87]	226	788	100%	0.06	0m10s	
	[81]	225	147	99.56%	0.01	0m51s	
	[83]	183	3,356	82.10%	0.27	0m14.4s	
	[9]	214	1,416	94.69%	0.11	1m01.6s	
	Ours	226	469	100%	0.04	2m43.1s	
MX_blind_partial (MX_benchmark1_clip)	1st place	51	66,818	92.73%	0.30	2m31.7s	224,974
	[9]	51	49,343	92.73%	0.22	2m10.6s	
	ours	55	27,880	100.00%	0.12	12m44.3s	
Array_benchmark2 (MX_benchmark2_clip)	1st place	489	20,383	98.00%	0.19	8m11.9s	106,954
	[87]	496	544	99.40%	0.01	1m43s	
	[81]	498	561	99.80%	0.01	6m30s	
	[83]	385	1,842	75.80%	0.02	3m04.8s	
	[9]	490	10,761	98.20%	0.10	1m02.7s	
	Ours	499	1,296	100.00%	0.01	13m58.4s	
Array_benchmark3 (MX_benchmark3_clip)	1st place	1,696	20,764	91.82%	0.17	18m44.0s	122,565
	[87]	1,801	2,052	97.51%	0.02	1m50s	
	[81]	1,806	2,660	97.78%	0.02	7m14s	
	[83]	1,271	2,407	68.80%	0.02	4m07.0s	
	[9]	1,697	13,025	91.88%	0.11	12m24.9s	
	Ours	1,846	2,938	99.95%	0.02	16m27.0s	
Array_benchmark4 (MX_benchmark4_clip)	1st place	161	3,726	83.85%	0.05	1m15.9s	82,010
	[87]	187	3,341	97.74%	0.04	1m9s	
	[81]	185	1,785	96.40%	0.02	5m34s	
	[83]	138	1,488	72.00%	0.02	1m43.3s	
	[9]	165	3,437	85.94%	0.04	5m29.1s	
	Ours	188	3,423	97.92%	0.04	5m57.5s	
Array_benchmark5 (MX_benchmark5_clip)	1st place	39	2,014	92.86%	0.04	0m26.6s	49,583
	[87]	40	94	95.12%	0.00	0m41s	
	[81]	40	245	95.12%	0.00	3m52s	
	[83]	28	444	63.40%	0.01	0m44.6s	
	[9]	39	1,111	92.86%	0.02	0m07.8s	
	Ours	42	700	100.00%	0.01	3m04.5s	

Since the intensity distribution of the 12 points is only for a single layer hotspot detection, Litho-aware Machine learning may not be used to detect multi-layer hotspots. In other words, intensities calculated using more than one layer are not

meaningful in terms of deciding whether there are hotspots or not between layers. In fact, overlay issue has not much to do with aerial image intensities. Therefore, if we want a machine learning model to be able to detect multi-layer hotspots, further research is needed to investigate which features are best rather than the 12 intensity features.

However, it may be worth of trying to combine the 12 intensity points at each layer, which are simulated independently, and use them in LAML SVM model training. For example, when we have two-layer hotspots, we create 24 intensity points, 12 per each layer, and we feed in these 24 features into our model training.

4.7 Litho-aware ML conclusion

We presented a novel methodology for machine learning-based hotspot detection. Our lithography-aware machine learning guides learning process using actual lithography information combined with lithography domain knowledge. While previous works for SVM modeling to identify hotspots have used only geometric related information which is not directly relevant to the lithographic process, our SVM model was trained with lithographic information which has a direct impact causing pinching or bridging hotspots. Furthermore, rather than creating a monolithic SVM trying to cover all hotspot patterns, we utilized lithography domain knowledge and separated hotspot types such as HB, VB, HP, and VP for our SVM model.

We also showed how we incorporated that lithography information into SVM kernel to accomplish an accurate decision function (classifier) for high accuracy result. The key point to create accurate SVM models for hotspot detection is to decrease model complexity by appropriate use of domain knowledge. Without

domain knowledge, it is difficult to find proper features that lead accurate models. Lithography simulation is all about aerial intensity distribution to create pattern images on a wafer to find hotspots. Therefore, considering intensity distribution in some way for hotspot detection machine learning is absolutely necessary for an accurate ML model. We used just 12 intensity points as 12 features for training SVM models for this thesis. It will be interesting to see if adding more intensity related features such as slope or curvature of the intensity points would enable us to generate more accurate models with lower false alarm rates.

Our experiment result confirms that our lithography-aware machine learning approach to detect hotspots outperforms all other previous works in this research field. As pointed out at 4.6, 100% accuracy with lowest false alarm rates for the real industry design, **"MX_blind_partial"**, is a remarkable result when considering other approaches are not even near 95%. In fact, our approach showed 100% accuracy with three testing layouts. Since not missing real hotspots is the first priority, this result is outstanding.

5

Conclusion

With a continued effort to shrink feature size as guided by Moor's law, Integrated Circuit Manufacturing process has become more and more prone to lithography hotspots. Even state-of-the-art semiconductor manufacturing processes adopting Optical Proximity Correction (OPC) [12] and Resolution Enhancement Techniques (RETs) such as Off-Axis illumination [59], Double or Multiple Patterning (DP or MP) [70, 82], and Phase-shift mask [85] often have challenges to avoid hotspots in layouts.

Since these lithography hotspots have an enormous impact on the yield of semiconductor manufacturing companies, it has become a critical task to find hotspots, which are caused by problematic patterns in a layout, during not only physical verification but also early physical design stage.

Lithography simulations can be used for the most accurate hotspot detection, and some researchers showed their works [19, 29, 30]. However, lithography simulations are extremely computational intensive, and it is not practical, if not almost impossible, to run the simulations on a full-chip level for identifying hotspots.

Recently, several approaches for hotspot detection have been proposed to avoid lithography simulations and to be accurate and efficient as well. They are mainly based on two ideas. One is pattern match, and the other is machine learning. Pattern match approach is fast and accurate, but it cannot identify previously unseen hotspots because it mainly relies on hotspot library which is composed of

previously known hotspot patterns to match for hotspot detection. In contrast, machine learning (ML) based hotspot detection has shown that it can identify previously unseen hotspots if the model is well trained. But, it suffers false alarm issue and needs special methods to suppress false alarm hotspots.

This thesis tackles the issue of pattern matching approach and the issue of ML-based approach for hotspot detection. We proposed our novel method of Edge Driven Dissected Rectangle based pattern match (EDDR PM) to be flexible enough for identifying previously unseen hotspot while maintaining accuracy not to miss any hotspot patterns in hotspot pattern library. Since our EDDR PM uses simple DRC rules, unlike other DRC-based pattern matching approaches, it does not suffer DRC rule explosion issue.

We showed that with our vector space concept, EDDR PM could avoid the unnecessary eight iterations to detect the same pattern with different orientations. Among all the member rectangles in a pattern, we used the center point of a member as an origin and the center point of another member as a reference point to create vector space and other necessary information. With this vector space information available at pattern matching process, it is possible to decide which orientation we are trying to match without going through eight iterations to cover all eight possible orientations of a pattern.

We detailed about how to implement EDDR PM for various pattern matching purposes such as exact match, partial match, and fuzzy match. By introducing range relational operations for pattern description, we demonstrated how flexible EDDR PM is to handle fuzzy pattern matching. We explained that partial matching is done by simply a skipping member check thanks to our core concept to EDDR PM, the concept of members or non-members in a pattern.

We also showed that our EDDR PM outperformed previous DRC-based pattern matching works. The biggest performance improvement came from Bin-search grid algorithm as explained at Section 3.5.3. Runtime complexity of EDDR PM has reduced from O(n^2) to O(n) by employing Bin-search grid algorithm. As expected, the runtime improvement for 4 million pattern matching test shown at Table 3.2 was tremendous from two days to 1.5 minutes. Our experimental result comparing [11] showed a huge speed up of 20 times faster on average shown at Table 3.5.

We presented how easy to extend EDDR PM for multi-layer pattern matching. It is one of many benefits of EDDR PM which comes from the fact that it adopted simple DRC edge rules for pattern match. We also discussed how additional DRC operations such as AREA, NET AREA, NET AREA RATIO, DEANGLE, RECTANGLE ENCLOSURE, and INSIDE may help for multi-layer pattern matching or another purpose of matching.

As a novel solution to ML based hotspot detection, we proposed Litho-aware machine learning (LAMA) which uses latent aerial image intensities. Our idea of LAMA directly relates lithography simulation information to SVM models during ML training using actual lithography information combined with lithography domain knowledge. Unlike previously developed ML based approaches which use only geometric information or require a post-OPC (Optical Proximity Correction) mask, our proposed method utilizes the use of detailed optical information but bypasses post-OPC mask by sampling latent image intensity and use those points to train an SVM model. The results suggest high accuracy and low false alarm, and faster runtime compared with methods that require a post-OPC mask.

Our domain knowledge about lithography illumination was used to decrease

model complexity by creating a separate SVM model for each horizontal bridging, vertical bridging, horizontal pinching, and vertical pinching hotspot types. Because our approach avoided the higher complexity of a lumped model dealing with all the possible hotspot types, it was possible for LAML to yield higher performance even with only 12 features (the 12 intensity points).

Another thing to note and emphasize the importance of domain knowledge is that lithography hotspot is mainly related to intensity distribution around the hotspot point. Therefore, this domain knowledge should be utilized to enable us to create a machine learning model. With this in mind, we found a reasonable and reliable metric to measure intensity distribution that distinguishes hotspot and non-hotspot pattern signature. As shown at Figure 4.18, the difference between maximum intensity and minimum intensity served well for our metric.

Since non-hotspot outnumbers hotspots, we matched the total number of non-hotspot data to the total number of hotspots in our training set to ensure a high-quality model. Otherwise, the imbalanced number between hotspot and non-hotspots can destroy the soft margin leading to a poor training result. We rebalanced it by including only the same number of non-hotspots as the number of hotspots in the training data. We discarded the rest of non-hotspots during SVM training.

With those domain knowledge incorporated in SVM model training, we created a machine learning model that achieved higher performance (accuracy) and lower false alarm rate. Our experimental result proved how important it is for domain knowledge to be incorporated into machine learning model. We presented that our LAML performance surpassed previous works for hotspot detection machine learning with a significant margin. Important data point in the comparison against

the ICCAD 2012 contest winner and [9] shown at Table 4.2 is that our LAML achieved 100% accuracy. There was no miss in finding real hotspots on an actual design, **"MX_blind_partial"**, which is the most important requirement for hotspot detection technology.

As a future topic to improve LAML, we suggested to include additional intensity related features such as the slope of the line of the 12 intensity points and curvature of the line. Since these features can be calculated so fast along with the 12 intensity points, there will be no impact on runtime both in training phase and testing phase while we can expect better performance in terms of accuracy.

6

Future Work

1. Additional DRC operations for pattern matching

   As discussed at Section 3.6.2, other DRC rules may show their importance in the field of pattern matching. Further research needs to be performed to determine that. Especially for multi-layer pattern matching, NET AREA RATIO and RECTANGLE ENCLOSURE operations seem excellent candidates for further studies.

2. Analytical Quantification of best distance between hotspot candidates

   As mentioned at 4.5.1, developing analytical quantification to decide the best distance between hotspot candidates along a long line would be helpful to make LAML more robust. Also, studying hybrid approach using EDDR PM and LAML would be an exciting research project in the future. Further research needs to be done by adding more features for SVM model training such as the slope of the 12 intensity points or curvature of the intensity line curve to see if this improves accuracy and false alarm rates.

3. Deep Neural Network using LAML idea

   As another future work, Deep Neural Network (DNN) using our LAML idea should be investigated. Since DNN performs automatic feature extraction on a distinct set of features based on the previous layer's output without human's intervention, it may discover important features we have not come up with yet. During DNN training, it can start with the features

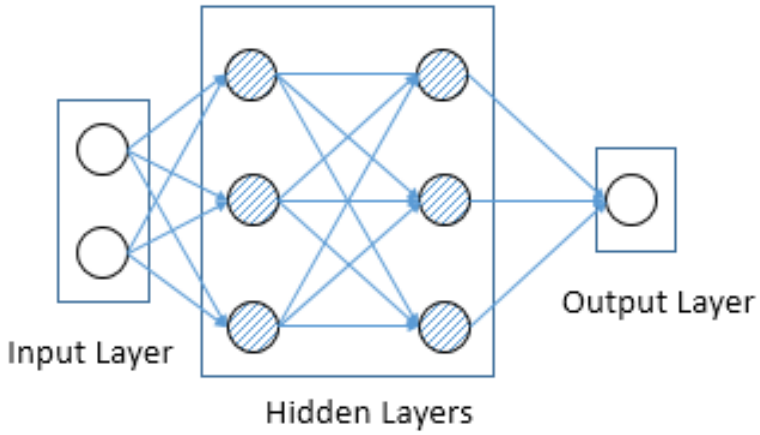


Figure 6.1: Deep Neuron Network with two hidden layers

we used for our work, the 12 intensity points, but generate a different set of features on the next layer of DNN, which may be important to train an accurate machine learning model which humans may not be able to discover. Through this feature hierarchy along layers on DNN, DNN is capable of handling complex features to model even more complex non-linear functions.

Figure 6.1 shows a typical four-layer DNN with two hidden layers. Besides that, there are many other forms of DNN such as Sparse Auto Encoder, Denosing Auto Encoder, Deep Belief Network, and etc (See Figure 6.2). Further investigation should be carried out to find out how many hidden layers are best for hotspot detection or to figure out what kind of network produces the most accurate model with lowest false alarm rate.

4. Hybrid approach using both EDDR PM and LAML

   As explained through out this thesis, pattern matching for hotspot detection is perfect for exact matches while it lacks to detect previously

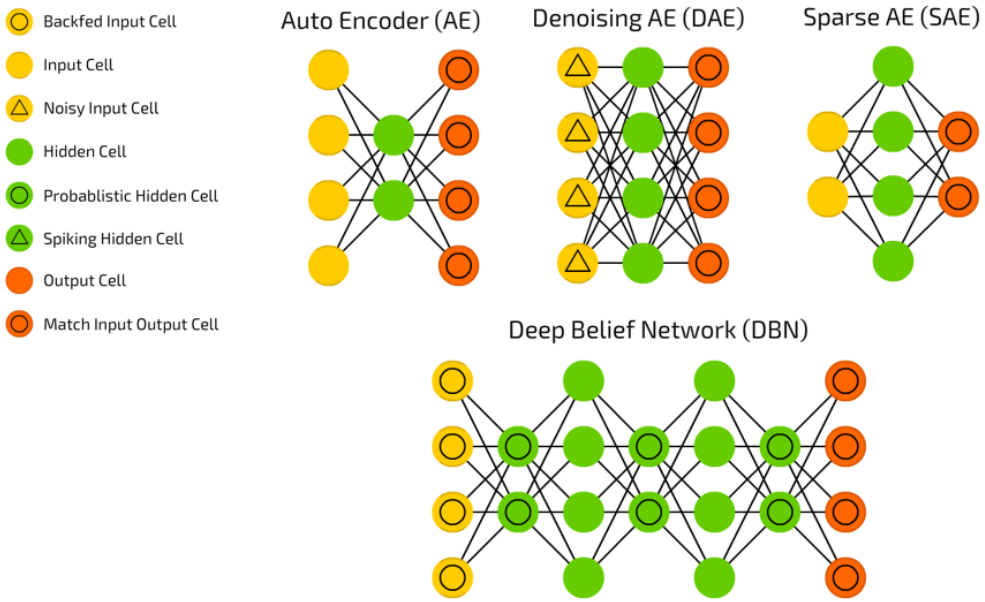


Figure 6.2: Several DNN types. Figure from [31]

unseen hotspots. In contrast, machine learning approach enables designers to detect problematic patterns that are not in hotspot libraries, but may increase false alarms.

Since EDDR PM is capable of not only finding previously known hotspots but also generating hotspot candidates with its flexible pattern description power in fuzzy ways, we can perform LAML on the candidates to decide whether those are hotspots or not. In this hybrid flow, both EDDR PM and LAML can work together such that it does not miss any known hotspots and covers previously unseen hotspots as well. Considering LAML's high accuracy with low false alarm rates, it may show a promising result unlike any other hybrid methods [3, 40, 42, 51, 83].

5. Lithography Friendly Routing

There has been a lot of research to avoid hotspot patterns in the earlier

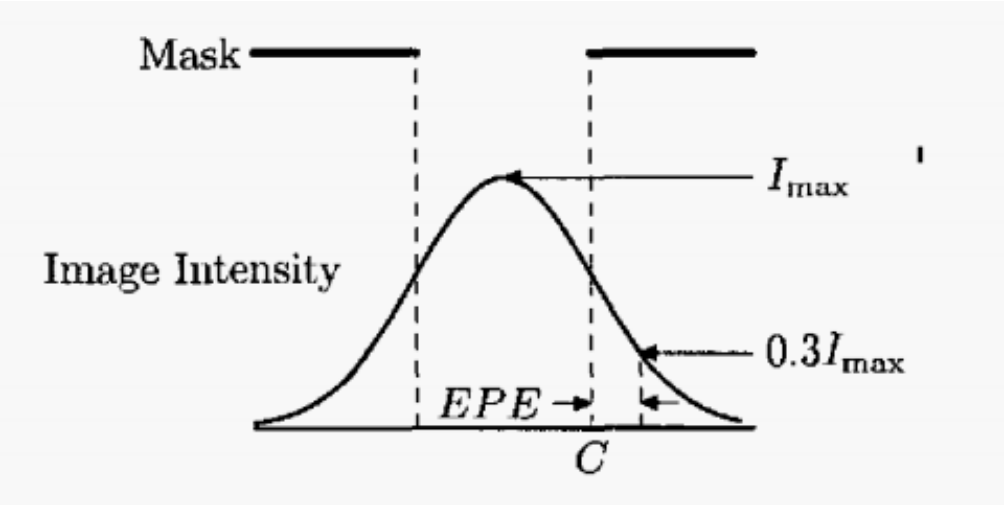


Figure 6.3: Edge Placement Error. The difference between the printed edge position and the original mask edge position is the edge placement error. It is approximated by a rule-of-thumb (30 % rule). Figure from [54]

design stages such as logic synthesis, placement, and routing [26, 58, 84]. Routing is probably a most critical physical design stage to solve hotspot issues. Research efforts have been focused in two paradigms, Construct-by-correction [54, 73, 80] and Correct-by-construction [5, 6, 56, 77].

In Construct-by-correction paradigm, routing is done first, and then hotspot detection process is carried out to fix found hotspots through post-routing optimization. It performs rip-up, re-route, doubling via, wire spreading/widening, and so on to remove found hotspots.

[54] attempted lithography simulations to create Edge Placement Error (EPE: See Figure 6.3) map and used it to measure overall printability issues. Guided by EPE map, they proposed RET-aware detailed routing where they tried EPE guided wire spreading, EPE guided rip-up and re-route.

[80] proposed model assisted routing. The authors also attempted lithography simulation to find real hotspots and fix them. A rule-based filtering is

Figure 6.4: Rule-Based Detection identifies hotspot candidates. These candidates are then confirmed or rejected by lithography simulation. Finally, hotspots are fixed by Rule-Based Correction. Figure from [73]

applied first to reduce the number of potential hotspots. Their fixing solution to remedy those hotspots is either widening gap to reduce bridging or making more room around shapes to reduce pinching. Below is a summary of [80]'s approach.

(1) In the first step, a conservative rule-based filter is applied to select potential hotspots.

(2) Router is then invoked to fix these potential hotspots.

(3) These two steps are repeated until the number of filtering hotspots is reduced to a manageable size.

(4) In the third step, they apply lithography simulation to the remaining filtering weak spots to determine the true hotspots.

(5) The router is called again to perform layout optimization to remove the lithography hotspots.

Luk-Pat [73] also tried a rule-based filtering similar with [80] before lithography simulation. Figure 6.5 shows an example of their rules and Figure 6.4 shows their workflow. Their solution for fixing hotspots is a rule-based correction that modifies found hotspot patterns in a pre-determined way. Their correction rule is tightly linked to a detection rule such that they don't identify hotspots that they don't know how to fix.

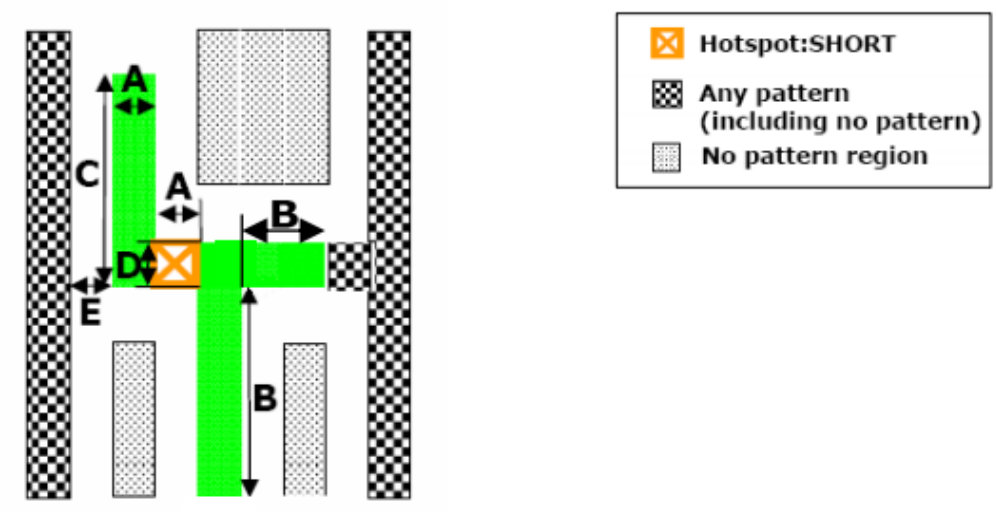


Figure 6.5: Example of a Detection Rule. The green wires must be present. The checkered wires may or may not be present. The finely-dotted regions must not be present. The green wires must have dimensions parameterized by A, B, C, D and E. These parameters must have values in specified ranges, where the ranges are depending on the process technology. Figure from [73]

In Correct-by-construction paradigm, routing considers lithography costs or constraints directly to avoid hotspots during routing. In essence, an effective Litho-metric capturing hotspots needs to be developed, and it must be incorporated efficiently into an existing routing framework such that routing generates Litho-friendly layouts.

One challenge of Correct-by-construction routing approach, also called lithography-aware routing, is that it is difficult to decide hotspots or not before a real routing path is created. Figure 6.6 shows an example of this difficulty. This example has a layout region with metal blockages and unrouted pins Pin1-Pin4. Because of this unrouted region, potential hotspots may occur by route Pin1-Pin2 as shown (b) of Figure 6.6.

Huang [77] proposed a maze routing method that considers lithography optical effect in the routing algorithm. The maze algorithm is a sequential

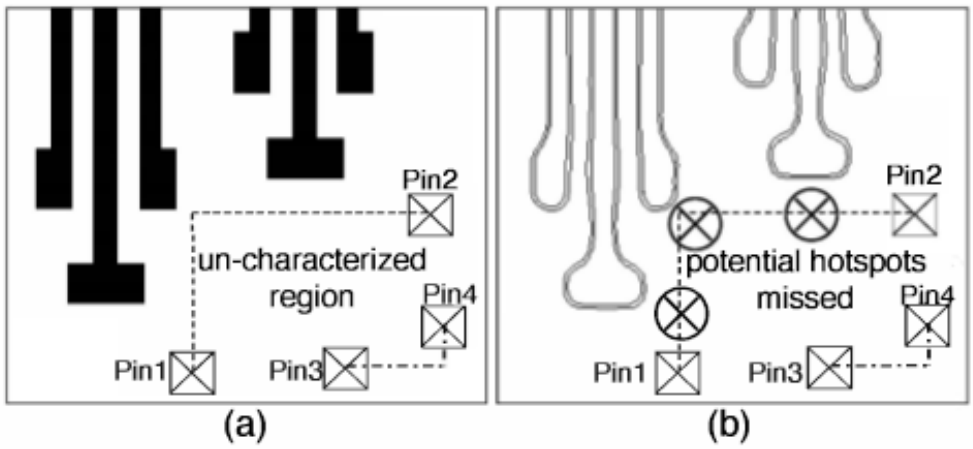


Figure 6.6: The hotspot detection challenge in the detailed routing stage. (a) routed layout region with metal blockages and unrouted pins, Pin1-Pin4. (b) lithography simulation to find hotspots. Note that due to unrouted pins Pin1-Pin4, potential hotspots may be missed. Figure from [52]

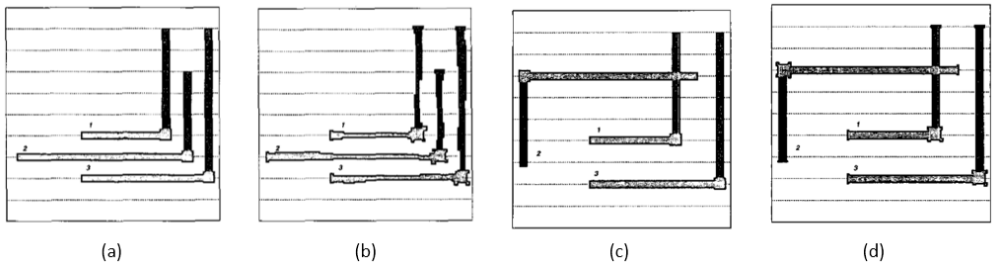


Figure 6.7: (a) Three nets with short lengths of two conductor layers, (b) OPC layout of (a), (c) The same three new with different paths, (d) OPC layout of (c). Note that fewer and less complicated OPC features are needed in the layout. The routing is friendlier to the OPC process, and the process window is wider. Figure from [77]

routing algorithm where one signal net is routed at a time. Based on their OPC (Optical Proximity Effect Correction) cost function, they try to find best routing paths as shown at Figure 6.7. Their OPC Constraint Maze Routing (OPCCMR) is a multi-constrained shortest path problem which can be solved by Lagrangian relaxation method. Their OPC cost function is derived from actual lithography simulations to create lookup tables.

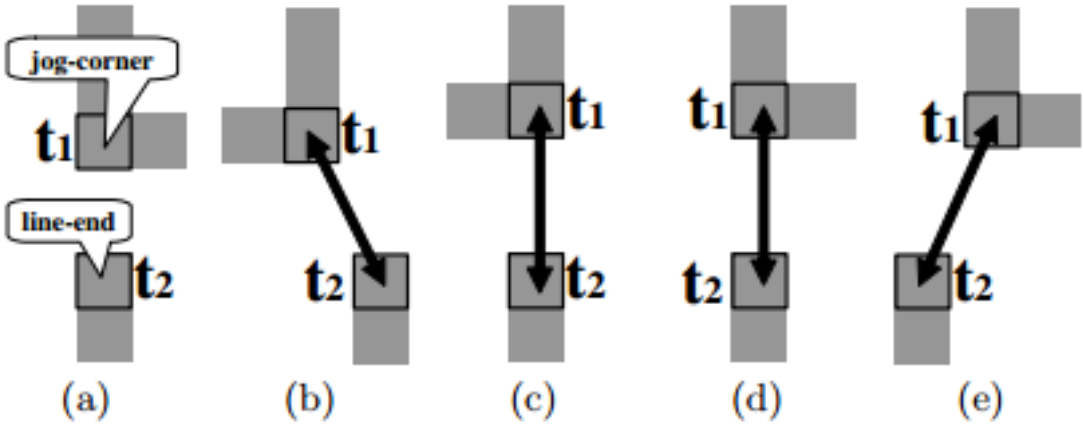


Figure 6.8: characterization for t1=jog-corner and t2=line-end is shown where (b), (c), (d), and (e) are the cases with the same distance. Thus, the mean EPE will characterize this interaction between t1 and t2 at this distance. Figure from [56]

Chen [6] also tried to consider OPC cost function during routing for Litho-Friendly Design. Their cost function was based on only line width, length, and adjacent lines.

Cho [56] proposed a compact post-OPC Litho-metric based on a statistical characterization. They predefined Litho-prone shapes, i.e., weak grids, such as jog-corner, via, line-end, and they characterized the interferences among them. They obtain the Litho-cost between weak-grid interactions at various distances based on post-OPC images. (See Figure 6.8) They estimated printability cost by calculating the summation of the Litho-cost among all weak-grid interactions within the lithography influence and process window. The Litho-metric derived from the characterization showed high fidelity to total edge placement error.

Chen [5] introduced Quasi-inverse lithography technique to estimate "OPC demand" which is their metric measuring OPC required area. Visualization of this "OPC demand" is depicted at Figure 6.9. The goal of their routing

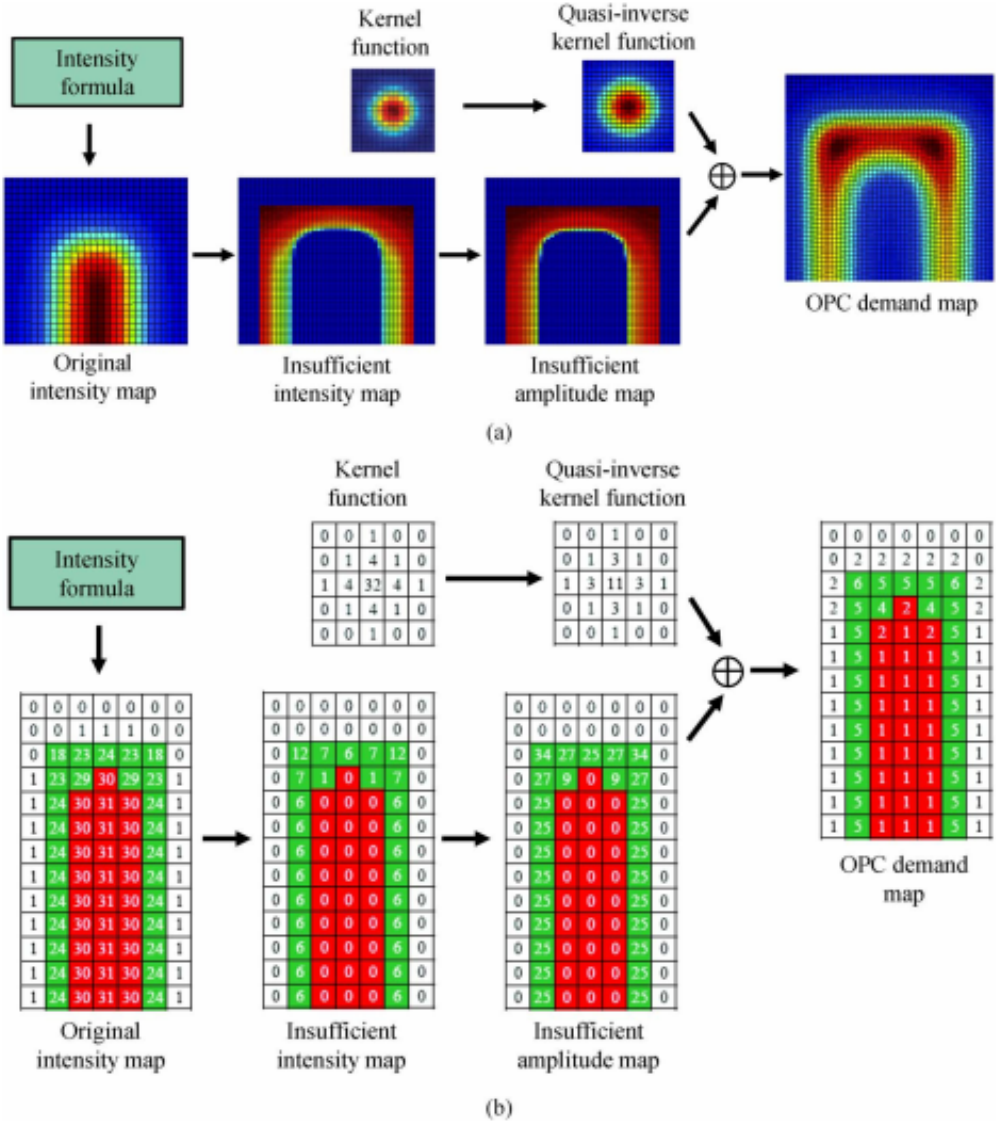


Figure 6.9: (a) Illustration of OPC demand calculation. (b) An example of OPC demand calculation. All numbers are multiplied by 100. The gray (green) and dark gray (red) grids denote the drawn and manufactured shapes, respectively. Note that (a) and (b) do not correspond to the same layout. Figure from [5]

is to minimize "OPC demand".

Construct-by-correction and Correct-by-construction are not perfect. These two approaches can be combined for a much better approach. First, we route

in Correct-by-construction fashion, trying to prevent most hotspots. Next, we perform the iterative process of Correct-by-construction by detecting hotspots and fixing those using post-routing optimization.

The main purpose of EDDR PM and LAML is to detect hotspots. Therefore, our work fits in Correct-by-construction. Since EDDR PM and LAML shows superiority to previous works, it will be interesting to try our work in Correct-by-construction for lithography-friendly routing. Furthermore, research about formulating a lithography cost function based on LAML idea should be studied for lithography-aware routing.

6. Feature Extraction

Feature extraction is a critical step in both machine learning and pattern matching. In fact, our thesis is an excellent example of the importance of feature extraction that is domain relevant for both pattern matching and machine learning. Simple but comprehensive feature information to represent patterns is key to an efficient and accurate hotspot detection. Therefore, more research on feature extraction should be carried out to find the best one for hotspot detection, especially for the machine learning approach.

Qiu [33] suggested a semi-supervised approach for feature extraction. He creates a semi-supervised model using Universum which is a data set sharing the same domain as the target problem. Recently, Universum is widely adopted for machine learning algorithm to encode prior knowledge (domain knowledge) into the model. In most machine learning problems, there are insufficient labeled data. Therefore, semi-supervised learning is required to enhance classification accuracy by using not only available labeled data but

also a great deal of unlabeled data.

Regarding semi-supervised model to generate extra labeled data from unlabeled data using Universum data, we create an initial model that uses available labeled data along with Universum data. Once the model is trained, it can classify each unlabeled data to obtain a confidence value about the classification. The classified unlabeled data that exceeds a specific threshold value of confidence is merged into the labeled set along with their classification labels. The new model can be constructed with the original labeled data and new labeled data to classify the rest of the unlabeled data set. In this process of semi-supervised learning, selecting the most effective Universum samples is important since not all the Universum samples are helpful.

As a case study, In 2006, Vapnik [68] experimented a binary classification of handwritten digits 5 and 8. For this classification problem, he created the following Universum sets:

(a) U1: randomly selected other digits (0, 1, 2, 3, 4, 6, 7, 9)

(b) U2: randomly mixing pixels from images 5 and 8

(c) U3: average of randomly selected examples of 5 and 8 (See Figure 6.10)

He showed that U-SVM (SVM used Universum data) outperformed a regular SVM. Below is his conclusion of U-SVM research which makes such an important point that we want to re-write it here.

"The idea of using an Universum is also about the use of additional data. However, here we do not require either the same distribution or labeling. The Universum idea is close to the Bayesian idea: the attempt to use

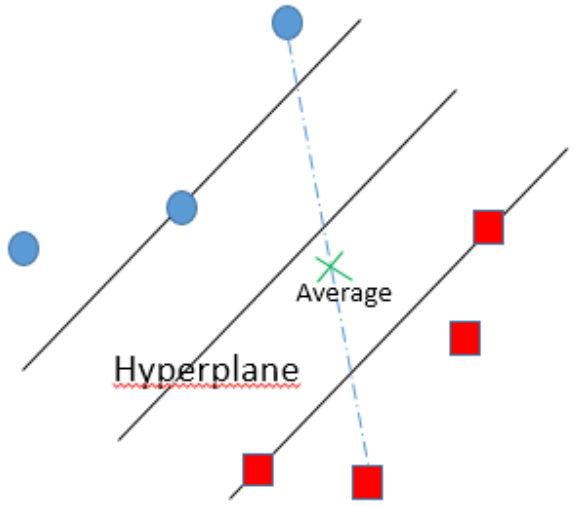


Figure 6.10: Universum data created by average of randomly selected examples.

prior knowledge. However there is a conceptual difference between the two approaches. In Bayesian inference, the prior knowledge is knowledge about decision rules, while the Universum is knowledge about the admissible collection of examples. People may have some feeling about a set of examples, but they may often know nothing about the distribution on the admissible set of functions. Like the Bayesian prior, the Universum encodes prior information. Unlike the Bayesian prior, the Universum distribution does not depend on the admissible family of functions. Our experiments show that the obtained performance depends on the quality of the Universum. The methodology of choosing the appropriate Universum is the subject of research. However, our results confirm that the Universum can be an important instrument for boosting performance, especially in the small sample size regime."

Note that U-SVM is particularly important when sample data set is small

such as hotspots. Therefore, research on how to create Universum hotspot data should be carried out to improve our hotspot detection model's performance. It would be interesting to see whether U3, which was experimented at [68], is going to have a positive impact on LAML in terms of accuracy and low false alarm rates.

It is also worth trying Deep Neural Network (DNN) with Universum data as well as intensity data from LAML concept. Even though DNN performs automatic feature extraction on a distinct set of features based on the previous layer's output and may discover important features we did not realize, initial input features which are sensible to the hotspot detection model is critical.

References

[1] Mahmoud A. M. Alshewimy; Mohamed T. Faheem Saidahmed Asmaa Hagar. A new object recognition framework based on pca, lda, and k-nn. pages 141–146. 11th International Conference on Computer Engineering and Systems (ICCES), 2016.

[2] H. Yao;S. Sinha; C. Chiang; X. Hong; Y. Cai. Efficient process-hotspot detection using range pattern matching. pages 625–632. ICCAD, 2006.

[3] Ning Ma; Justin Ghan ; Sandipan Mishra ; Costas Spanos ; Kameshwar Poolla ; Norma Rodriguez ; Luigi Capodieci. Automatic hotspot classification using pattern-based clustering. SPIE, 2008.

[4] J. Lin; Yao-Wen Chang. Tcg: A transitive closure graph based representation for non-slicing floorplans. pages 764–769. Design Automation Conference, 2001.

[5] Tai-Chen Chen; Guang-Wan Liao; Yao-Wen Chang. Predictive formulae for opc with applications to lithography-friendly routing. Design Automation Conference, 2008.

[6] Tai-Chen Chen; Yao-Wen Chang. Multilevel full-chip gridless routing with applications to optical-proximity correction. volume 26, pages 1041–1053. IEEE Trans. on Computer-aided Design, 2007.

[7] Ke Chen. Comp24111 machine learning at university of manchester. https://studentnet.cs.manchester.ac.uk/ugt/COMP24111/ materials/slides/Hierarchical.ppt.

[8] J. Xu; Subarna Sinha; Charles C. Chiang. Accurate detection for process-hotspots with vias and incomplete specification. pages 839–846. ICCAD, 2007.

[9] Yen-Ting Yu; Geng-He Lin; Iris Hui-Ru Jiang; Charles Chiang. Machine-learning-based hotspot detection using topological classification and critical feature extraction. volume 34, pages 460–470. IEEE TCAD, 2015.

[10] Yen-Ting Yu; Iris Hui-Ru Jiang; Yumin Zhang; Charles Chiang. Drc-based hotspot detection considering edge tolerance and incomplete specification. pages 101–107. ICCAD, 2014.

[11] Yen-Ting Yu; Ya-Chung Chan; Subarna Sinha; Iris Hui-Ru Jiang; Charles Chiang. Accurate process-hotspot detection using critical design rule extraction. pages 1167–1172. DAC, 2012.

[12] Nicolas Cobb. *Fast Optical and Process Proximity Correction Algorithms for Integrated Circuit Manufacturing*. PhD thesis, 1998.

[13] A. Veloso; S. Demuynck; M. Ercken; A. M. Goethals; M. Demand; J. F. de Marneffe; E. Altamirano; A. De Keersgieter; C. Delvaux; J. De Backer; S. Brus; J. Hermans; B. Baudemprez; F. Van Roey; G. F. Lorusso; C. Baerts; D. Goossens; C. Vrancken; S. Mertens; J. J. Versluijs; V. Truffert; C. Huffman; D. Laidler; N. Heylen; P. Ong; B. Parvais; M. Rakowski; S. Verhaegen; A. Hikavyy; H. Meiling; B. Hultermans; L. Romijn; C. Pigneret; S. Lok; A. Van Dijk; K. Shah; A. Noori; J. Gelatos; R. Arghavani; R. Schreutelkamp;

P. Boelen; O. Richard; H. Bender; L. Witters; N. Collaert; R. Rooyackers; P. Absil; A. Lauwers; M. Jurczak; T. Hoffmann; S. Vanhaelemeersch; R. Cartuyvels; K. Ronse; S. Biesemans. Full-field euv and immersion lithography integration in 0.186um2 finfet 6t-sram cell. pages 1–4. IEEE International Electron Devices Meeting, 2008.

[14] Bernhard Schölkopf Dennis Decoste. Training invariant support vector machines. *Journal Machine Learning*, 46:161–190, 2002.

[15] H. Sewell; Jan Mulkens ; Paul Graeupner ; Diane McCafferty ; Louis Markoya ; Sjoerd Donders ; Nandasiri Samarakone ; Rudiger Duesing. Extending immersion lithography with high index materials– results of a feasibility study. volume Vol. 6520, pp.65201M-65201M. SPIE, 2007.

[16] Rong en Fan, Kai wei Chang, Cho jui Hsieh, Xiang rui Wang, and Chih jen Lin. Liblinear: A library for large linear classification, 2008.

[17] Semiconductor Engineering. http://semiengineering.com/kc/knowledge_center/Design-Rule-Pattern-Matching/243.

[18] JR. F. Pikus; Truman W. Collins. Topological pattern matching. US Patent 018594 A1, July 2010.

[19] J. Kim; Minghui Fan. Hotspot detection on post-opc layout using full chip simulation based verification tool. pages 919–925. SPIE, 2003.

[20] INTERNATIONAL Technology Roadmap for Semiconductors. (available: http://www.itrs2.net/itrs-reports.html). Technical report, 2015.

[21] Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. In *Machine Learning*, pages 277–296, 1998.

[22] D.Z. Pan; Bei Yu; Jhih-Rong Gao. Design for manufacturing with emerging nanolithography. volume 32, pages 1458–1472. IEEE TCAD, 2013.

[23] F. Gennari. Fast pattern matching. US patent 7818707, Oct. 2010.

[24] Mentor Graphics. Standard Verification Rule Format(SVRF) Manual.pdf.

[25] Hailong Yao; S. Sinha; J. Xu; C. Chiang; Y. Cai; X. Hong. Efficient range pattern matching algorithm for process-hotspot detection. pages 2–15. IET Circuits, Devices and Systems, 2008.

[26] Shiyan Hu; Pratik Shah; Jiang Hu. Pattern sensitive placement for manufacturability. pages 27–34. Int. Symp. on Physical Design, 2007.

[27] W. Hu. https://www.utdallas.edu/ wxh051000/teaching/EE6322/Lecture

[28] Hong-Yan Su; Chieh-Chu Chen; Yih-Lang Li; An-Chun Tu; Chuh-Jen Wu; Chen-Ming Huang. A novel fast layout encoding method for exact multilayer pattern matching with prufer encoding. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions*, 34:95–108, 2015.

[29] E. Roseboom; Mark Rossman ; Fang-Cheng Chang ; Philippe Hurat. Automated full-chip hotspot detection and removal flow for interconnect layers of cell-based designs. SPIE, 2007.

[30] M. Cote; P. Hurat. Layout printability optimization using a silicon simulation methodoly. pages 159–164. International Symposium of Quality Electronic Design, 2004.

[31] Asimov Institure. http://www.asimovinstitute.org/wp-content/uploads/2016/09/neuralnetworks.png.

[32] M. Hagan; Howard B. Demuth; Mark Hudson Beale; Orlando De Jesus. Neural network design. volume I. Pws Boston, 1996.

[33] Xin Li Junyang Qiu; Yanyan Zhang; Zhisong Pan; Haimin Yang; Huifeng Ren. A novel semi-supervised approach for feature extraction. pages 3765–3770. Neural Networks (IJCNN) 2016 International Joint Conference, 2016.

[34] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation, 2002.

[35] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection, 1995.

[36] Greg Lamp. http://www.yaksis.com/static/img/02/cows_and_wolves.png.

[37] C. Chang; Chih-Jen Lin. Libsvm: A library for support vector machines. volume 2. ACM Transaction of Intelligent System and Technology (TIST), 2011.

[38] A. Talwalkar M. Mohri, A. Rostamizadeh. Foundations of machine learning., 2012.

[39] Chris Mack. Tutor42.doc; Version 4/20/03 The Lithography Expert (August 2003).

[40] Salma Mostafai; J. Andres Torres ; Peter Rezk ; Kareem Madkour. Multi-selection method for physical design verification applications. pages 263–270. SPIE, 2012.

[41] J. Wuu; Fedor G. Pikus ; Andres Torres ; Malgorzata Marek-Sadowska. Detecting context sensitive hotspots in standard cell libraries. volume 7275, 757512. SPIE, 2009.

[42] J. Wuu; Fedor G. Pikus ; Malgorzata Marek-Sadowska. Efficient approach to early detection of lithgrapic hotspots using machine learning systems and pattern matching. volume 7974. SPIE, 2012.

[43] J. Wuu; Fedor G. Pikus; Andres Torres; Malgorzata Marek-Sadowska. Rapid layout pattern classification. pages 781–786. ASP-DAC, 2011.

[44] Jianliang Li; Lawrence S. Melvin. Sub-resolution assist features in photolithography process simulation. IEEE Microprocesses and Nanotechnology, 2007. DOI: 10.1109/IMNC.2007.4456270.

[45] G. Milligan. A monte carlo study of thirty internal criterion measures for cluster analysis. pages 187–199. Psychometrika, 1981.

[46] Gordon Moors. Cramming more components onto integrated circuits, reprinted from electronics. volume 38, pages 114–ff. Electronics, 1965.

[47] Ramanathan Narayanan, Daniel Honbo, Gokhan Memik, Alok Choudhary, and Joseph Zambreno. An fpga implementation of decision tree classification. In *In Design, Automation and Test in Europe Conference(DATE)*, 2007.

[48] N. Nagase; Kouichi Suzuki; Kazuhiko Takahashi; Masahiko Minemura ; Satoshi Yamauchi; Tomoyuki Okada. Study of hotspot detection using neural network judgement. volume 6607, pages 141–146. SPIE, 2007.

[49] D. Ding;Andres J. Torres; Fedor G. Pikus; David Z. Pan. High performance lithographic hotspot detection using hierarchically refined machine learning. pages 775–780. ASP-DAC, 2011.

[50] D. Ding;Xiang Wu; Joydeep Ghosh; David Z. Pan. Machine learning based lithographic hotspot detection with critical feature extraction and classification. pages 219–222. ICICDT, 2009.

[51] Duo Ding; Bei Yu; Joydeep Ghosh; David Z. Pan. Efficient prediction of ic manufacturing hotspots with a unified meta-classification formulation. pages 263–270. ASP-DAC, 2012.

[52] Duo Ding; Jhih-Rong Gao; Kun Yuan; David Z. Pan. Aeneid: a generic lithography-friendly detailed router based on post-ret data learning and hotspot detection. pages 795–800. Design Automation Conference, 2011.

[53] J. Gao; Bei Yu ; David Z. Pan. Accurate lithography hotspot detection based on pcasvm classifier with hierarchical data clustering. volume 9053. SPIE, 2014.

[54] J. Mitra; Peng Yu; D. Z. Pan. Radar: Ret-aware detailed routing using fast lithography simulations. pages 369–372. Design Automation Conf., 2005.

[55] Jae-Seok Yang; Katrina Lu; Minsik Cho; Kun Yuan; David Z. Pan. A new graph theoretic, multi objective layout decomposition framework for double patterning lithography. Asia South Pac. Design Automation Conf., Taipei, Taiwan, 2010.

[56] Minsik Cho; Kun Yuan; Yongchan Ban; David Z. Pan. Eliad: Efficient lithography aware detailed router with compact post-opc printability prediction. Design Automation Conference, 2008.

[57] Youngchan Ban; Kevin Lucas; David Pan. Flexible 2d layout decomposition framework for spacer-type double pattering lithography. 48th ACM/EDAC/IEEE Design Automation Conference (DAC), 2011.

[58] P. Gupta; A. B. Kahngt; Chul-Hong Park. Detailed placement for improved depth of focus and cd control. pages 343–348. Pacific Design Automation ConI, 2005.

[59] F. Schellenberg. A little light magic [optical lithography]. *IEEE Spectrum*, pages 34–39, 2003.

[60] Lawrence Hubert; James Schultz. Quadratic assignment as a general data-analysis strategy. pages 190–241. Br. J. Math. Stat. Psychol, 1976.

[61] Mark Simmons. Calibre Pattern Matching: Picture It, Match It...Done!, Mentor Graphics: mentorpaper_57885.pdf.

[62] M. Levenson; N. S. Viswanathan; Robert A. Simpson. Resolution in photolithography with a phaseshifting mask. volume ED 29, no. 12, pages 1826–1836. IEEE Transaction on Electron Deuices, 1982.

[63] Jea Park; Robert Todd; Xiaoyu Song. Geometric pattern matching using edge driven dissected rectangles and vector space. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35, Issue:12:2046–2055, 2016. DOI: 10.1109/TCAD.2016.2535908.

[64] Andres Torres. Iccad-2012 cad contest in fuzzy pattern matching for physical verification and benchmark suite. ICCAD Special session, 2012.

[65] K. Ronse; Ph. Jansen; R. Gronheid; E. Hendrickx; M. Maenhoudt; M. Goethals; G. Vandenberghe. Lithography options for the 32nm half pitch node and beyond. pages 371–378. IEEE Custom Integrated Circuits Conference, 2008.

[66] B. E. Boser; Isabelle M. Guyon; Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. pages 144–152. Comput. Learn Theory (COLT), 1992.

[67] C. Cortes; Vladimir N. Vapnik. Support-vector networks. volume 20, pages 273–297. Journal of Machine Learning, 1995.

[68] Jason Weston; Ronan Collobert; Fabian Sinz; Leon Bottou; Vladimir Vapnik. Inference with the universum. pages 1009–1016. 23rd international conference on Machine learning, 2006.

[69] Vladimir N. Vapnik. The nature of statistical learning theory, 1999.

[70] G. Bailey; Alexander Tritchkov; Jea-Woo Park; Le Hong; Vincent Wiaux; Eric Hendrickx; Staf Verhaegen; Peng Xie; Janko Versluijs. Double pattern eda solutions for 32nm hp and beyond. SPIE, 2007.

[71] D. Drmanac; Frank Liu; Li-C. Wang. Predicting variability in nanoscale lithography processes. 46th IEEE/ACM Design Automation Conference (DAC), 2009.

[72] Kart-Leong Lim; Han Wang. Learning a field of gaussian mixture model for image classification. pages 1–5. 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), 2016.

[73] Gerard T. Luk-Pat; Alexander Miloslavsky; Atsuhiko Ikeuchi; Linni Wen. Correcting lithography hot spots during physical-design implementation. SPIE, 2006.

[74] Micheal White. http://electronicdesign.com/eda/pattern-matching-next-step-eda-s-evolution.

[75] Wikipedia. https://en.wikipedia.org/wiki/Immersion_lithography.

[76] Wikipedia. https://en.wikipedia.org/wiki/Design_rule_checking.

[77] Li-Da Huang; M. D. F. Wong. Optical proximity correction opc friendly maze routing. pages 186–191. Design Automation Conference, 2004.

[78] Manhua Shen; Manhua Shen; Qiang Wu. Sub-resolution assist feature (sraf) study for active area immersion lithography. pages 1–3. China Semiconductor Technology International Conference, IEEE, 2015.

[79] Andrew Kahng; Chul-Hong Park; Xu Xu. Fast dual graph based hotspot detection. volume 6349, pages 628–635. SPIE, 2006.

[80] Tim Kong; Hardy Leung ; Vivek Raghavan ; Alfred K. Wong ; Sarah Xu. Model assisted routing for improved lithography robustness. volume 6521. SPIE, 2007.

[81] Haoyu Yang. Imbalance aware lithography hotspot detection: A deep learning approach. SPIE, 2017.

[82] Jian Kuang; Wing-Kai Chow; Evangeline F. Y. Young. Triple patterning lithography aware optimization and detailed placement algorithms for standard cell-based designs. *IEEE VLSI System*, pages 1319–1332, 2016.

[83] Sheng-Yuan Lin; Jing-Yi Chen; Jin-Cheng Li; Wan yu Wen; Shih-Chieh Chang. A novel fuzzy matching model for lithography hotspot detection. pages 1–6. ACM/IEEE DAC, 2012.

[84] Minsik Cho; David Z. Pan; Kun Yuan. Manufacturability aware routing. in Handbook of Algorithms for VLSI Physical Automation (edited by Alpert, Mehta, and Sapatnekar), CRC Press, 2008.

[85] A. B. Kahng; S. Vaya; A. Zelikovsky. New graph bipartizations for double-exposure, bright field alternating phase-shift mask layout. pages 186–191. ASP-DAC (Design Automation Conference), 2001.

[86] J. Guo; Fan Yang; Subarna Sinha; Charles Chiang; Xuan Zeng. Improved tangent space based distance metric for accurate lithographic hotspot classification. pages 1173–1178. ACM/IEEE DAC, 2012.

[87] H. Zhang. Enabling online learning in lithography hotspot detection with information-theoretic feature optimization. pages 1–8. ICCAD, 2016.

[88] Z. Zeng; ShunZhi Zhu. A kernel-based sampling to train svm with imbalanced data set. pages 1–5. IEEE, 2013.

[89] A. Zisserman. http://www.robots.ox.ac.uk/ az/lectures/ml/lect2.pdf.