Deep Learning-Driven Simultaneous Layout Decomposition and Mask Optimization

Wei Zhong*[†], Shuxiang Hu*[†], Yuzhe Ma[‡], Haoyu Yang[‡], Xiuyuan Ma*[†], Bei Yu[‡]
*DUT-RU International School of Information Science & Engineering, Dalian University of Technology, China
[†]Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, China
[‡]Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong SAR
zhongwei@dlut.edu.cn, byu@cse.cuhk.edu.hk

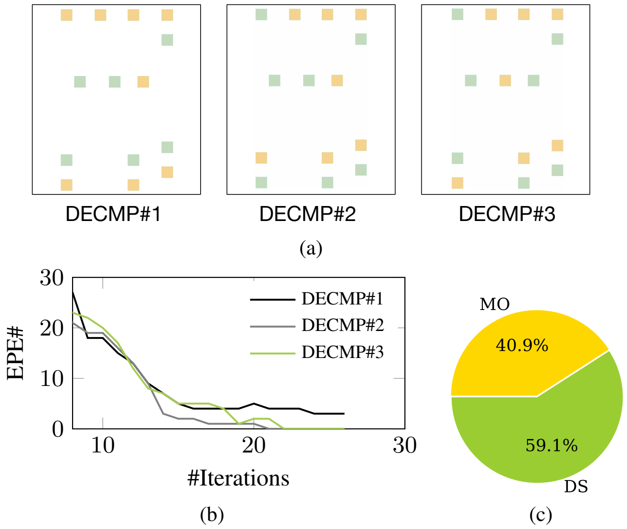
***Abstract*—Combining multiple pattern lithography (MPL) and optical proximity correlation (OPC) pushes the limit of 193nm wavelength lithography to go further. Considering that layout decomposition may generate plenty of solutions with diverse printabilities, relying on conventional mask optimization process to select the best candidates for manufacturing is computationally expensive. Therefore, an accurate and efficient printability estimation is crucial and can significantly accelerate the layout decomposition and mask optimization (LDMO) process. In this paper, we propose a CNN based prediction and integrate it into our new high performance LDMO framework. We also develop both the layout and the decomposition sampling strategies to facilitate the network training. The experimental results demonstrate the effectiveness and the efficiency of the proposed algorithms.**

I. Introduction

The shrinkage of device feature size has reached the resolution limit of the 193nm wavelength lithography, thus various resolution enhancement techniques (RETs) are heavily applied in the post-layout design flow to maintain a good pattern printability when transferring patterns from mask to wafer. Multiple patterning lithography (MPL) and optical proximity correlation (OPC) are two of very promising solutions among all the RETs.

MPL is currently widely applied to enhance the resolution in industry. The key step in MPL is the layout decomposition which assigns the conflicted patterns on a layout to separated masks for manufacturing. To achieve better decomposition quality, various methods have been proposed [1]–[4]. OPC or mask optimization is able to handle the optical distortions in subwavelength lithography [5] by refining the pattern shapes on a mask. There are also different approaches proposed to solve this problem [6]–[9].

After conducting layout decomposition, multiple solutions can be obtained, as shown in Fig. 1(a). To further enhance the printability, mask optimization is performed and we can select the masks with the highest printability. Since the mask optimization is a subsequent step of the layout decomposition, the final quality is determined, to a large extent, by the layout decomposition result. Fig. 1(b) shows corresponding trajectories during mask optimization process of different decomposition results. It is observed that the printability is not consistently good or bad for a given instance. Only after the entire process is completed can we tell the good ones from the bad ones. However, it is computationally expensive to run all solutions through mask optimization process due to the overhead imposed by lithography modeling. Recently it has been demonstrated that a simultaneous layout decomposition and mask optimization framework can ease the gap and obtain high quality and high printability masks in a unified way [10], in which the final masks are generated by the collaboration of mask optimization engine and discrete optimization engine. However, there are still some deficiencies when selecting the best decomposition since selection is performed with a greedy pruning scheme which relies on mask optimization process to estimate the printability. Given a situation like Fig. 1(b), the method proposed in [10] is not an ideal solution. On one hand, leveraging mask optimization engine for printability estimation is expensive as mentioned before. Fig. 1(c) shows the proportion of mask optimization (MO) and decomposition selection (DS). It can be seen that time consumption on decomposition selection is even larger than that on mask optimization, which motivates us to explore a more effective and efficient way for decomposition selection, i.e., printability estimation. On the other hand, the greedy pruning is based on the printability of intermediate mask optimization results, which is not an accurate estimation and hence leads to sub-optimal solutions. Therefore, an efficient and accurate printability prediction approach is promising to enhance and accelerate the design flow.



(a)

(b)

(c)

Fig. 1 Optimization runtime and decomposition convergence comparison; (a) Different decomposition optimization results of the same layout; (b) Corresponding decomposition convergence of EPE; (c) The runtime breakdown: comparison between decomposition selection (DS) and mask optimization (MO).

Deep learning has drawn great attention for its ability of learning automatically from a large amount of data. In the electronic design automation (EDA) field, deep learning has been widely applied in various EDA applications. As the most conventional model, convolutional neural networks (CNNs) have been adopted for routability estimation [11], lithography hotspot detection [12] and resist modeling [13]. In this work, we propose a deep learning-driven framework based on CNN to predict and further improve the printability of masks. The framework contains a layout generation module, a printability estimation module and a mask optimization module. To generate high-quality decomposition candidates, we use n-wise method and minimum spanning tree (MST) to construct decomposition rules during the generating process. Since we can hardly evaluate the printability of a decomposition, CNN is used to help us select the best decomposition candidate. Besides, in order to promote the accuracy of prediction and accelerate the training process, we designed our sampling strategy. The main contributions of this work are as follows:

- We propose a CNN based layout decomposition evaluation module to predict the printability after mask optimization.
- We combine the MST and n-wise method to generate layout decomposition more efficiently.
- We develop a set of sampling approaches to sample the representative decomposition as the training set. A comparison to random sampling strategy shows our sampling method reduces about half of the edge placement errors (EPE).
- Experimental results show that our framework outperforms other previous works and reduces the EPE violations at least 68.0%.

The rest of this paper is organized as follows:

Section II introduces the background of double pattern lithography and gives the problem definition. Section III describes the overall flow of our framework. Section IV shows the details of training the CNN, including layout sampling, decomposition sampling, and CNN training. Section V and Section VI present the experimental results and conclusion.

II. Preliminaries

In this section, we will introduce some background of layout decomposition, inverse lithography technology and give the problem formulations.

The task of mask optimization for double pattern lithography is generating a pair of optimized masks to get the output of lithography simulation model $\boldsymbol{T}$ and the target image $\boldsymbol{T}'$ as close as possible. Gradient descent based inverse lithography technology (ILT) has been widely applied to generate the output mask. In order to make the param of lithography simulation model differentiable, we applied sigmoid function as follows

$$\boldsymbol{M}_i(x,y) = \frac{1}{1+e^{-\theta_m \boldsymbol{P}_i(x,y)}}, \tag{1}$$

In this way, we express the binary value mask $\boldsymbol{M}$ with unbound param $\boldsymbol{P}$, θ_m is the coefficient to control the slope of the sigmoid function. Similarly, constant threshold photo resist model can be presented as

$$\boldsymbol{T}_i(x,y) = \frac{1}{1+e^{-\theta_z(\boldsymbol{M}_i(x,y)-\boldsymbol{I}_{th})}}. \tag{2}$$

In our implementation, θ_m, θ_z and I_{th} are set to 8, 120 and 0.039 respectively. In double pattern inverse lithography technology, printed image is organized in the following form

$$\boldsymbol{T}(x,y) = \min\{\boldsymbol{T}_1(x,y) + \boldsymbol{T}_2(x,y), 1\}. \tag{3}$$

Then we can derive the gradient of $\boldsymbol{T}(x,y)$ with respect to $\boldsymbol{P}_i(x,y)$, and update corresponding $\boldsymbol{M}_i(x,y)$ by performing gradient descent process. More details about the gradient formulation can be seen in [10].

Definition 1 (EPE)**.** *EPE measures the manufacturing distortion by the difference of edge placement between the printed image and the target layout. A checkpoint will be marked as an EPE violation if its EPE greater than a given threshold value.*

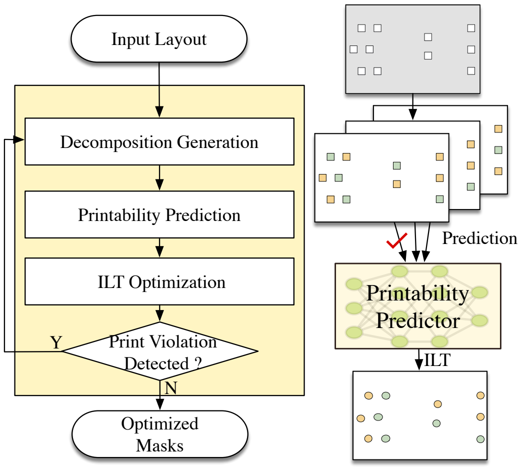


Fig. 2 Overview of the proposed LDMO flow.

Definition 2 (L_2 Error)**.** *L_2 Error indicates the difference between the printed image $\boldsymbol{T}$ and the target image $\boldsymbol{T}'$, which is defined as $\|\boldsymbol{T}-\boldsymbol{T}'\|_2^2$.*

EPE is one of the most important criterions of image printability. ILT process reduces the L_2 Error in each iteration to minimize the number of EPE indirectly. In our work, both EPE and L_2 Error are selected as printability metrics.

The task of layout decomposition for double pattern lithography is to generate a pair of decomposed masks so that the masks follow the design rule. This process can be presented like

$$f_{decomp}(\boldsymbol{T}') = \boldsymbol{M}_1, \boldsymbol{M}_2, \tag{4}$$

where $\boldsymbol{T}'$ is the target image and $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ are decomposition result. The subsequent ILT process generates optimized masks to minimize the differences between printed image and target image. Equation (5) described this process.

$$f_{ILT}(\boldsymbol{M}_1, \boldsymbol{M}_2) = f_{ILT}(f_{decomp}(\boldsymbol{T}')) = \boldsymbol{T}, \tag{5}$$

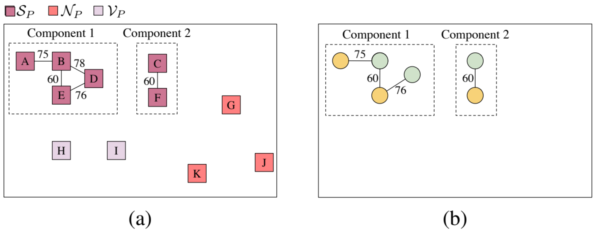
where $\boldsymbol{T}$ is the printed image after ILT optimization. We can evaluate its quality by organizing the combination form of EPE and L_2 Error. Thus, the f_{ILT} converts to a continuous function. In this work, we try to use CNN to regress the continuous function derived from Equation (5) and use the network to guide the process of Equation (4) to generate better decomposition.

Based on the above definitions, the layout decomposition and mask optimization (LDMO) problem can be described as

Problem 1 (LDMO)**.** *Given a target image $\boldsymbol{T}'$, generate a mask decomposition so that the subsequent mask optimization process can get minimal EPE violations.*

III. Overall Flow

The overall flow of our optimization framework is shown in Fig. 2. First, decomposition candidates are generated according to the input layout. Since the number of candidates increases exponentially with the number of patterns, we build several MSTs and apply n-wise method to avoid generating violated decomposition. Then all candidates are sent to printability prediction module. The trained convolutional neural network scores every candidate and output the best layout decomposition. Next, ILT is operated to optimize masks and outputs the final masks. To avoid the estimation error, print violations will be checked during the ILT optimization process. Once print violation occurs, we will select a decomposition candidate



(a) (b)

Fig. 3 $\mathcal{S}_P$ distribution solution; (a) Constructed weighted graph; (b) MST solution.

again. The following subsections will give a detail description of each step.

A. Decomposition Generation

Decomposition generation module produces decomposed mask candidates for the next module according to the given layout. The flow of this subsection is described in Algorithm 1. As shown in Fig. 3(a), layout patterns are first divided into three sets: separated pattern ($\mathcal{S}_P$) set, violated pattern ($\mathcal{V}_P$) set, and normal pattern ($\mathcal{N}_P$) set. According to the distance d from the nearest patterns, the belonging of pattern $\mathcal{T}$ is determined by Equation (6).

$$\mathcal{T} \in \begin{cases} \mathcal{S}_P, & \text{if } d \leq n_{min}, \\ \mathcal{V}_P, & \text{if } n_{min} < d \leq n_{max}, \\ \mathcal{N}_P, & \text{if } n_{max} < d. \end{cases} \tag{6}$$

Print violation occurs when the distance between two patterns is less than n_{min}. As the increasing of distance d, print violation disappears gradually, but there still exists the interaction between two patterns until the distance reaches n_{max}. Therefore, patterns in $\mathcal{S}_P$ always cause print violations, so they are ought to be separated from each other. $\mathcal{V}_P$ are the kind of patterns that tend to cause printability decline while $\mathcal{N}_P$ have a slight effect on the performance compared to the other two types. When we generate decomposition candidates, we will take different strategies according to the pattern type. In our implementation, n_{min} is set to 80, and n_{max} is set to 98.

Algorithm 1 Decomposition Generation

```
Require: Input layout $L$.
  $\mathcal{S}_P$, $\mathcal{V}_P$, $\mathcal{N}_P$ ← PatternClassify($L$);
  $V$ ← SolveMST($\mathcal{S}_P$);
  $Arrs1$ ← GetThreeWiseArrays($V$, $\mathcal{S}_P$,$\mathcal{V}_P$);
  $Arrs2$ ← GetTwoWiseArrays($\mathcal{N}_P$);
  $mergedArrs1$ ← CheckAndMerge($Arrs1$);
  $mergedArrs2$ ← CheckAndMerge($Arrs2$);
  $S$ ← CombineSolution($mergedArrs1$, $mergedArrs2$);
  for $j = 1 \rightarrow S.$size do
      $K$ ← DrawImage($S_j$);
      $Img$.save($K$);
  end for
  return $Img$
```

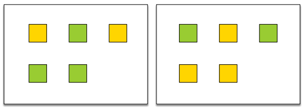
The problem of decomposition is usually solved by constructing a graph and converts to a coloring problem. For $\mathcal{S}_P$, they are going to be separated from each other. Take patterns in $\mathcal{S}_P$ as vertexes and the distance among them as the weight of edge, thus a weighted graph is built (see Fig. 3(a)). The closer two patterns are, the stronger their interaction is, so the nearest nodes should be separated in the first place. In the condition of double pattern lithography, this problem converts to finding the MST of a weighted graph. As shown in Fig. 3(b), there are two connected components, so the MST problem can be solved independently. Based on the result of MST, two neighbor vertexes can be assigned to different masks to avoid the print violations. Besides, the relative position relationship of patterns in the same MST can be inferred, which provides the basis of our following decomposition analysis.

$Arrs1$	$\mathcal{S}_P$		$\mathcal{V}_P$	
	B	F	H	I
#1	1	0	0	1
#2	1	1	1	1
#3	0	0	1	1
#4	0	0	0	0
#5	0	1	1	0
#6	1	1	0	0
#7	1	0	1	0
#8	0	1	0	1

(a)

$Arrs2$	$\mathcal{N}_P$		
	G	J	K
#1	0	1	0
#2	0	0	1
#3	1	1	1
#4	1	0	0

(b)



(c)

Fig. 4 n-wise arrays and dual decomposition: (a) The generated three-wise arrays; (b) The generated two-wise arrays according to $\mathcal{N}_p$; (c) Two different images represent the same decomposition.

In the case of $\mathcal{V}_P$ and $\mathcal{N}_P$, we shall pay more attention to $\mathcal{V}_P$ because they may exercise more influence on the optimization result. Another requirement is that we want to reduce the number of decomposition candidates as much as possible. To generate representative decomposition candidates meanwhile limiting the size of candidate set, n-wise method is applied. The n-wise test method (also known as combinatorial test method) has been used to test compiler [14] by R. Mandl since 1985. In our decomposition process, covering all the combinations of patterns is prohibitively expensive which is similar to software testing. n-wise method is usually used to analysis the main factors affecting the experiment with the smallest test set. n represents the interaction of the maximum factors we can test according to the generated arrays. An example of two-wise (pairwise) arrays with 4 patterns is shown below.

$$\begin{array}{c|cccc} & \text{factor1} & \text{factor2} & \text{factor3} & \text{factor4} \\ \hline \text{instance \#1} & 1 & 0 & 0 & 0 \\ \text{instance \#2} & 1 & 1 & 1 & 1 \\ \text{instance \#3} & 0 & 1 & 0 & 1 \\ \text{instance \#4} & 0 & 0 & 1 & 1 \\ \text{instance \#5} & 0 & 1 & 1 & 0 \end{array}$$

In the generated arrays, each row is an instance of decomposition, each column is a pattern (factor) and the value determines which masks this pattern belongs to. Picking any two columns, the complete combination of them $(00, 01, 10, 11)$ exists, which means n-wise method reduces the strength of factor combination to minimize the generated arrays meanwhile maintaining the complete combination of any n factors. Naturally, if n is set to the number of factors, the test set becomes Cartesian Product of all factors.

On the basis of MST result, we obtained the relative position relationship of $\mathcal{S}_P$. Imaging a layout has N patterns in $\mathcal{N}_P$, V patterns in $\mathcal{V}_P$ and M MSTs. Randomly picking a pattern from every connected component as a factor, then we apply three-wise method together with the patterns in $\mathcal{S}_P$, one possible result $Arrs1$ is shown in Fig. 4(a). As for patterns in $\mathcal{N}_P$, two-wise method is used to generate $Arrs2$ (see Fig. 4(b)). After that, two arrays with different combination strength are created. From the table, you can see that the number of instances didn't grow too much with the number of factors.

The output of decomposition generation module is a gray-scale image with different grayscale levels to represent patterns distributed on different masks. n-wise method generates the minimal training set with strength n, but there exists the same decomposition candidates. Since the masks are unordered, a layout decomposition can be represented by two different images as shown in Fig. 4(c). To solve this problem, we manually number the masks and fix the pattern numbered 1 on $\boldsymbol{M}_1$ so that the two masks become ordered. Once pattern numbered 1 is distributed on $\boldsymbol{M}_2$, the value of this row will be reversed. Then we merge the group with the same value to drop the same decomposition. Note that this operation will not destroy the relative position relationship among patterns. So the total decomposition candidate number should be size$(mergedArrs1) \times$ size$(mergedArrs2)$.

B. Printability Prediction

The printability prediction module evaluates the decomposition printability by giving candidates scores. All decomposition candidates are fed to CNN in the form of gray-scale image. This module scores each input and outputs the decomposition with the minimal score which is equivalent to the best printability after the ILT optimization.

Considering the predicting error sometimes happens, the decomposition selected by the printability prediction module may cause print violations in the subsequent mask optimization phase. So we mark the previous outputs and when facing the same decomposition, we drop it to avoid giving the same output. Details about CNN training will be discussed in Section IV.

C. ILT Optimization

At this point, the decomposition candidate has been obtained. To avoid the predicting error, we check it by detecting print violations indirectly.

Considering the print violations may happen at any stage of ILT optimization, we detect them every three iterations. Once violations are detected, we go back to decomposition generation step and select another decomposition solution, otherwise we continue to optimize the masks. The gradient g of $\|\boldsymbol{T}-\boldsymbol{T}'\|_2^2$ with respect to $\boldsymbol{P}_i$ is used to update parameters in the form of $\boldsymbol{P}_i = \boldsymbol{P}_i - \text{stepSize} \times g$. Next, we update $\boldsymbol{M}_i$ and $\boldsymbol{T}$ with new $\boldsymbol{P}_i$ accodring to Equations (1) to (3). In our implementation, the max iteration round is set to 29.

IV. Training of Prediction Network

In this section, sampling and training approaches are detailed. We first present the layout sampling and decomposition sampling strategy, then we discuss the network structure and decomposition evaluation metrics of the model. The overall flow of the prediction network training is illustrated in Fig. 5.

A. Layout Sampling

The number of patterns and their distribution make the count of layout practically unlimited. But due to the diffraction and interference of light, printed image quality is largely dependent on the nearby patterns. To capture layout common features (e.g. recognize typical pattern distribution, ignore slight layout movement and rotation) and express light propagation property, scale-invariant feature transform (SIFT) [15] is applied in our layout sampling method. SIFT has been widely used in computer version filed to capture local features of an image. The feature points are stable for rotation and scaling which are suitable for choosing as the reference for classifying the layout. There is an example of SIFT feature points in Fig. 6, even such actions as translation, rotation and scaling occurs, these points will not change. And local features are attached to the points, so they can be used to represent local similarity of an image. We measure the similarity of the two layouts by matching the feature points and calculate the distance between them indirectly .

Let $\boldsymbol{p}$, $\boldsymbol{q}$ be the 128-dimensional feature vector calculated by the SIFT algorithm. D_{th} is the threshold to determine if the two feature points are matched. Therefore, the distance between two feature points (vectors) is defined as

$$d(\boldsymbol{p},\boldsymbol{q}) = \begin{cases} \sqrt{\boldsymbol{p}\boldsymbol{q}^\top}, & \text{if } \sqrt{\boldsymbol{p}\boldsymbol{q}^\top} \le D_{th}; \\ 1, & \text{otherwise}, \end{cases} \tag{7}$$

where $\boldsymbol{q}^\top$ is transform of $\boldsymbol{q}$. Equation (7) shows that if the two feature points are close enough, the distance is the euclidean distance. Otherwise, it means they are not matched, the distance between them is their L2-Norm which is 1. We set D_{th} to 0.7 in our implementation. Based on the distance definition of two points, we are able to calculate the distance between two layouts.

Algorithm 2 Calculate Layout Similarity

```
Require: Layout $L_w$ and $L_s$.
  Let $\boldsymbol{p}_1^w, \boldsymbol{p}_2^w, ..., \boldsymbol{p}_n^w$ be the feature poins in $L_w$;
  Let $\boldsymbol{p}_1^s, \boldsymbol{p}_2^s, ..., \boldsymbol{p}_m^s$ be the feature poins in $L_s$;
  Initialize empty array $D^{ws}$;
  for $i = 1 \to n$ do
      for $\boldsymbol{p}_i^w$, find minimum $d(\boldsymbol{p}_i^w, \boldsymbol{p}_j^s)$ with unmatched $\boldsymbol{p}_j^s$;
      if $d(\boldsymbol{p}_i^w, \boldsymbol{p}_j^s) \le D_{th}$ then
          mark $\boldsymbol{p}_i^w$, $\boldsymbol{p}_j^s$ matched;
          put $d(\boldsymbol{p}_i^w, \boldsymbol{p}_j^s)$ in $D^{ws}$;
      else
          put 1 in $D^{ws}$;
      end if
      sort $D^{ws}$ in an ascending order;
  end for
  $S(L_w, L_s) = \sum_{k=1}^{c} D_k^{ws}$;
  return $S(L_w, L_s)$;
```

Algorithm 2 shows the similarity calculation between layout w and layout s. For a feature point in L_w, we need to find a feature point in L_s that is not marked as matched and ensure the point distance $d(\boldsymbol{p}_i^w, \boldsymbol{p}_j^s)$ is minimum. If the $d(\boldsymbol{p}_i^w, \boldsymbol{p}_j^s)$ is less than D_{th}, it means the similarity of these two points are high and the pattern distribution near this point is very similar, so we mark them as matched. D^{ws} records the distance between matched feature points. Because the number of feature points differs as the change of layout, the length of D^{ws} after above process is not the same. In order to make all distances comparable, we sort D^{ws} in an ascending order, then take the first c additions as the layout distance $S(L_w, L_s)$.

After calculating the layout distance, we use a clustering algorithm to divide the layout into several classes to obtain the representative layouts, and randomly select the layout from each class as the input of Section IV-B. The representative objects of k-medoids clustering are called medoids. They are the real points that exist in the cluster, and the k-medoids clustering is less sensitive to noise points compared to k-means. The performance of k-medoids is evaluated by the sum of layout distance (SLD), which is defined by

$$\text{SLD} = \sum_{i=1}^{M} \sum_{L_k \in C_i} S(L_k, L_i), \tag{8}$$

where C_i is a class whose center point is L_i. There are M classes and L_k is non-central point belongs to C_i. SLD represents the sum of distance from each non-central point to its respective center point. The object is to reduce SLD by changing center point and calculating the distance of each class. We set m to 50, c to 60 and we randomly

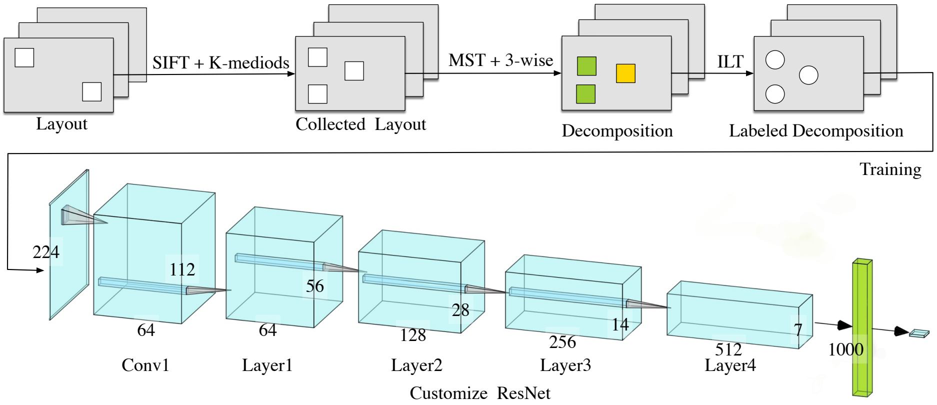


Fig. 5 CNN training flow.

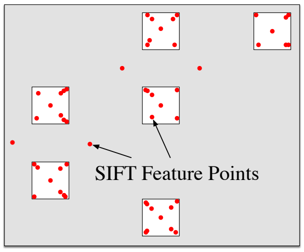


Fig. 6 Example of SIFT feature.

select 5 layouts in each cluster.

B. Decomposition Sampling

Similar to the problem of layout sampling, the decomposition number of a specific layout increases exponentially as the pattern number grows. For a layout with n patterns, there are 2^{n-1} different decompositions, which are too large to collect all of them as our training set.

Considering CNN has the property of translation invariant, and the EPE occurrence is highly related with nearby distribution. So the complete combination of patterns in a sub-region will be more helpful for our training and n-wise method is able to handle this problem. In order to focus more promising decomposition solution, we also combine MST and n-wise method to generate training set.

Different from decomposition generation phase, here we only focus on the patterns with the distance less than n_{min} as generating the score of decomposition is much more time-consuming. These patterns are divided into $\mathcal{S}_P$, while the rest patterns are non-violated pattern ($\mathcal{N}_P$) set.

Like generating $mergedArrs1$ in Section III-A, we first divide patterns into two types, $\mathcal{S}_P$ and $\mathcal{N}_P$. By solving MST problem, we can obtain the relative position relationship of $\mathcal{S}_P$, then we build the three-wise arrays together with $\mathcal{N}_P$. Finally, we reverse value and merge the same rows. In our implementation, setting n to 3 is a trade-off between prediction accuracy and simulation running time. Three-wise sampling strategy ensures that any sub-region with three patterns (part of patterns in $\mathcal{S}_P$ are excluded), the training set contains complete combination of them.

C. Model Training

The input of CNN is a gray-scale image, so we need to check the distribution of pattern numbered 1 and solve the dual issue as mentioned in Section III-A. In this paper, L_2 error and EPE numbers are selected as the evaluation metrics. Note the occurrence of print violations will lead to significantly decline in printability, the score of decomposition is organized as follows:

$$\text{score} = \alpha \times \#L_2 \text{ Error} + \beta \times \#\text{EPE} + \gamma \times \#\text{Violation}. \quad (9)$$

In our implementation, α, β and γ are 1, 3500 and 8000 respectively, and z-score regularization is applied to make the score comparable. The label in Equation (9) is continuous, which makes our model a regression problem.

TABLE I Comparison with previous frameworks

ID	[16]+ [6]		[17]+ [6]		[10]		Ours	
	EPE #	Time (s)	EPE #	Time (s)	EPE #	Time (s)	EPE #	Time (s)
1	1	1183	1	1183	1	1995	0	169
2	8	1421	4	1405	1	1996	1	208
3	5	867	5	1068	1	1990	0	228
4	1	1046	1	737	0	1996	0	176
5	5	1213	7	1207	6	1996	0	217
6	5	1048	8	1080	1	1989	2	295
7	13	1080	13	1061	1	1993	0	273
8	7	1046	3	1220	0	1997	1	234
9	5	1173	6	1176	1	1997	1	241
10	7	1171	8	774	3	1998	0	303
11	5	1188	2	1233	0	1987	1	229
12	5	773	4	1168	7	1999	4	299
13	11	1017	10	922	6	1998	0	272
Ave.	6.00	1094.31	5.53	1094.92	2.15	1994.69	0.69	241.84
Ratio	8.69	4.52	8.01	4.53	3.12	8.25	**1.00**	**1.00**

Considering that the ILT-based OPC phase may introduce noise, in order to overcome the effects of noise, the following mean absolute error (MAE) is applied as the cost function:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}, \quad (10)$$

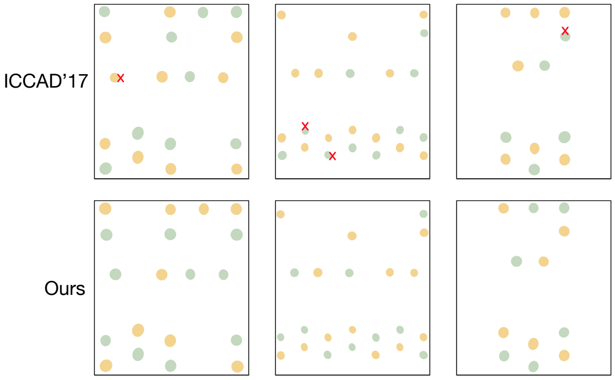
where y_i denotes the label of the i-th decomposition in the training set, $\hat{y}_i$ is the corresponding predicted value.

The image has 224 × 224 dimensions which increases the complexity of training. Here the Adam optimizer is selected to train the model. Compared to the mini-batch gradient, Adam computes individual adaptive learning rates for different parameters which is more suitable for large scale data.

We take the structure of ResNet18 as the basic regression network (see Fig. 5). The identity mapping between each block enables the net layer to become deeper to obtain a better regression of object function. The input of the net is 224 × 224 × 1 tensor to receive a grayscale image. Identity mapping is added between two 3 × 3 conventional layers. After average pooling, there is a 1000 dimensions layer, and a fully connected layer is added to output the score.

V. Experimental Result

Our optimization framework is implemented in C++ and validations are performed on Intel i7 3.6Ghz CPU. To generate n-wise sampling arrays, we use PICT [18], an open source C library. The EPE violation threshold is set to $10nm$. Experiments are conducted on an open source cell library NanGate [19]. We manually generate layout dataset that contains 8000 contact layout designs. These designs resemble NAND gate $45nm$ library and are verified with Mentor



(a) (b) (c)

Fig. 7 Comparison with ICCAD'17 [10] on: (a) AOI211_X1; (b) NAND3_X2; (c) BUF_X1. Compared with the state-of-the-art, in all three cases our proposed framework can effectively remove EPE.

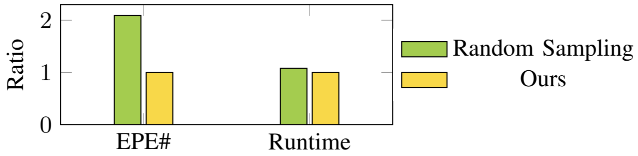


Fig. 8 Comparison with random sampling strategy.

Calibre design rule check.

In the first experiment, we compare the optimization result of our framework with previous unified framework and two-stage independent flow on standard cell library NanGate, as shown in TABLE I. We obtain binary from the authors of [10], and the results of the two other flows are directly from [10]. Columns "EPE" and "Time (s)" list the number of EPE violations and the cost of time when the optimization is convergent.

From TABLE I, our framework shows significant improvement in "EPE" and "Time" on average. The runtime of our framework is 241.84 seconds which achieves around four times speed-up in comparison with " [16]+ [6]" and " [17]+ [6]", and eight times speed-up comparing to " [10]". That is because there is no need to solve the SDP problem and operate decomposition selection by lithography simulation both of which are extremely time-consuming. In the term of "EPE", our framework reaches a three times reduction comparing to [10] and at least six times compared to two conventional flows (i.e. [16]+ [6] and [17]+ [6]). There are some examples of optimization result in Fig. 7.

In the second experiment, we test the efficiency of our sampling strategy. "Random Sampling" in Fig. 8 is a network trained with the data we randomly selected in both the stages of layout sampling and decomposition sampling. "Ours" is trained in the way we introduced in Section IV. From the bar chart we can see that the EPE number of "random sampling" is about twice of ours. The reason is that for a specific layout, the solution space is too large, but many of the results can be discarded by formulating some physical rules. Our sampling strategy focuses on the possible decomposition while the random sampling is equivalent to sampling evenly at the solution space, which needs much more training data.

VI. Conclusion

In this paper, we propose a deep learning-driven framework to not just bridge the gap between layout decomposition and mask optimization, but also speed-up the procedure of selecting decomposition and ILT process. To improve the predicting accuracy and accelerate the training process, we use SIFT feature extraction, k-medoids clustering and n-wise method to generate the training set. We also combine the MST and n-wise method to generate decompositions with different pattern combination strengths. Experiments demonstrate that our framework can efficiently reduce EPE violations and accelerate the overall optimization process.

VII. Acknowledgment

This work is partially supported by National Natural Science Foundation of China (NSFC) under Grant 61906029, the Fundamental Research Funds for the Central Universities, and The Research Grants Council of Hong Kong SAR (No. CUHK24209017).

References

[1] J. Kuang and E. F. Y. Young, "An efficient layout decomposition approach for triple patterning lithography," in *Proc. DAC*, 2013, pp. 69:1–69:6.

[2] H.-Y. Chang and I. H.-R. Jiang, "Multiple patterning layout decomposition considering complex coloring rules," in *Proc. DAC*, 2016, pp. 40:1–40:6.

[3] B. Yu, K. Yuan, D. Ding, and D. Z. Pan, "Layout decomposition for triple patterning lithography," *IEEE TCAD*, vol. 34, no. 3, pp. 433–446, March 2015.

[4] S.-Y. Fang, Y.-W. Chang, and W.-Y. Chen, "A novel layout decomposition algorithm for triple patterning lithography," *IEEE TCAD*, vol. 33, no. 3, pp. 397–408, March 2014.

[5] W. Xiong, J. Zhang, Y. Wang, Z. Yu, and M.-C. Tsai, "A gradient-based inverse lithography technology for double-dipole lithography," in *International Conference on Simulation of Semiconductor Processes and Devices*, 2009, pp. 1–4.

[6] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan, "MOSAIC: Mask optimizing solution with process window aware inverse correction," in *Proc. DAC*, 2014, pp. 52:1–52:6.

[7] J. Kuang, W.-K. Chow, and E. F. Y. Young, "A robust approach for process variation aware mask optimization," in *Proc. DATE*, 2015, pp. 1591–1594.

[8] A. Poonawala and P. Milanfar, "Mask design for optical microlithography–an inverse imaging problem," *IEEE Transactions on Image Processing*, vol. 16, no. 3, pp. 774–788, 2007.

[9] Y.-H. Su, Y.-C. Huang, L.-C. Tsai, Y.-W. Chang, and S. Banerjee, "Fast lithographic mask optimization considering process variation," *IEEE TCAD*, vol. 35, no. 8, pp. 1345–1357, 2016.

[10] Y. Ma, J.-R. Gao, J. Kuang, J. Miao, and B. Yu, "A unified framework for simultaneous layout decomposition and mask optimization," in *Proc. ICCAD*, 2017, pp. 81–88.

[11] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *Proc. ICCAD*, 2018, pp. 80:1–80:8.

[12] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE TCAD*, 2018.

[13] Y. Lin, Y. Watanabe, T. Kimura, T. Matsunawa, S. Nojima, M. Li, and D. Z. Pan, "Data efficient lithography modeling with residual neural networks and transfer learning," in *Proc. ISPD*, 2018, pp. 82–89.

[14] R. Mandl, "Orthogonal latin squares: an application of experiment design to compiler testing," *Communications of the ACM*, vol. 28, no. 10, pp. 1054–1058, 1985.

[15] D. G. Lowe, "Lowe, d.: Object recognition from local scale-invariant features. in: Proc. iccv," in *Iccv*, 1999.

[16] Z. Chen, H. Yao, and Y. Cai, "SUALD: Spacing uniformity-aware layout decomposition in triple patterning lithography." in *Proc. ISQED*, 2013, pp. 566–571.

[17] B. Yu and D. Z. Pan, "Layout decomposition for quadruple patterning lithography and beyond," in *Proc. DAC*, 2014, pp. 53:1–53:6.

[18] Microsoft Corporation, "Pairwise independent combinatorial testing," https://github.com/microsoft/pict.

[19] "NanGate FreePDK45 Generic Open Cell Library," http://www.si2.org/openeda.si2.org/projects/nangatelib, 2008.