Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

SVM based layout retargeting for fast and regularized inverse lithography

Kai-sheng LUO[†], Zheng SHI, Xiao-lang YAN, Zhen GENG
(*Institute of VLSI Design, Zhejiang University, Hangzhou 310027, China*)
[†]E-mail: luoks@vlsi.zju.edu.cn

Received Dec. 9, 2013; Revision accepted Mar. 11, 2014; Crosschecked Apr. 11, 2014

Abstract: Inverse lithography technology (ILT), also known as pixel-based optical proximity correction (PB-OPC), has shown promising capability in pushing the current 193 nm lithography to its limit. By treating the mask optimization process as an inverse problem in lithography, ILT provides a more complete exploration of the solution space and better pattern fidelity than the traditional edge-based OPC. However, the existing methods of ILT are extremely time-consuming due to the slow convergence of the optimization process. To address this issue, in this paper we propose a support vector machine (SVM) based layout retargeting method for ILT, which is designed to generate a good initial input mask for the optimization process and promote the convergence speed. Supervised by optimized masks of training layouts generated by conventional ILT, SVM models are learned and used to predict the initial pixel values in the 'undefined areas' of the new layout. By this process, an initial input mask close to the final optimized mask of the new layout is generated, which reduces iterations needed in the following optimization process. Manufacturability is another critical issue in ILT; however, the mask generated by our layout retargeting method is quite irregular due to the prediction inaccuracy of the SVM models. To compensate for this drawback, a spatial filter is employed to regularize the retargeted mask for complexity reduction. We implemented our layout retargeting method with a regularized level-set based ILT (LSB-ILT) algorithm under partially coherent illumination conditions. Experimental results show that with an initial input mask generated by our layout retargeting method, the number of iterations needed in the optimization process and runtime of the whole process in ILT are reduced by 70.8% and 69.0%, respectively.

Key words: Inverse lithography technology, Optical proximity correction, Layout retargeting, Support vector machine
doi:10.1631/jzus.C1300357 **Document code:** A **CLC number:** TN47

1 Introduction

Lithography is an important step in integrated circuit (IC) manufacturing for transferring patterns from mask to wafer; however, distortion always exists in lithography due to diffraction and interference of light waves (ITRS, 2012). Various resolution enhancement technologies (RETs) have been proposed to improve the pattern fidelity in lithography (Wong, 2001; Chiang and Kawa, 2007).

Optical proximity correction (OPC) is one of the most important RETs by adjusting the topology of the mask to make the printed pattern on the wafer as close to the desired pattern as possible. Traditional OPC methods, including rule-based approaches (Garofalo *et al.*, 1994; Park *et al.*, 2000) and model-based approaches (Cobb and Zakhor, 1995; Chen *et al.*, 2007; Lin *et al.*, 2011), solve the mask optimization problem based on the edges of the patterns. Segments of edges are shifted from the original locations according to either the rule table or simulation results, to compensate for the optical proximity effects. However, the traditional OPC is gradually becoming insufficient, as the 193 nm wavelength is now applied to the 36 nm node and even beyond. Taking advantage of the increasing computational powers, inverse lithography technology (ILT) is now being proposed to replace traditional OPC in advanced technology nodes. Unlike traditional OPC, ILT implements a pixel-based mechanism to achieve a more complete

© Zhejiang University and Springer-Verlag Berlin Heidelberg 2014

exploration of the solution space and better pattern fidelity.

The lithography system can be described using Eq. (1), where the function Litho(·) represents the lithography system, including resist development. ILT treats the mask optimization problem as an inverse problem, and the mathematical description of ILT is described using Eq. (2).

$$\mathbf{contour} = \mathrm{Litho}(\mathbf{mask}), \tag{1}$$

$$\mathbf{mask}^* = \mathrm{Litho}^{-1}(z), \tag{2}$$

where **contour** is the lithography result on the wafer, z represents the target patterns on the wafer, and $\mathbf{mask}^*$ is the optimal mask to be calculated in the ILT algorithm. However, the $\mathrm{Litho}^{-1}(\cdot)$ does not exist due to the ill-posedness of function Litho(·); therefore, the inverse lithography problem is translated into an optimization problem to find the optimal $\mathbf{mask}^*$ iteratively, with the aim of making the distance between the contour and the target as small as possible, which is described in Eq. (3):

$$\mathrm{mask}^*(x,y) = \arg\min_{\mathrm{mask}^*(x,y)} d(z(x,y), \mathrm{Litho}(\mathrm{mask}^*(x,y))), \tag{3}$$

where $d(\cdot,\cdot)$ is a distance metric.

In the past 20 years, various methods have been proposed to solve this optimization problem. In the early years, discontinuous methods based on pixel flipping or nonlinear programming were employed (Oh *et al.*, 1999; Erdmann *et al.*, 2004; Granik, 2005; 2006). More recently, this optimization problem was converted into an unconstrained, continuous optimization problem; gradient based methods or level-set based methods were used to optimize the mask iteratively (Poonawala and Milanfar, 2007a; Ma and Arce, 2007; 2008; Pang *et al.*, 2008; Shen *et al.*, 2009; Yang YW *et al.*, 2009; Ma *et al.*, 2012a; 2012b). However, no matter which methods are being used, ILT is always time-consuming due to the slow convergence of the optimization process.

In traditional model-based OPC (MB-OPC), the layout retargeting method, also known as pre-bias, which pre-warps the original layout design based on the rule table or simulation results, is always used to generate a better initial guess, therefore accelerating the convergence of the following MB-OPC (Kotani *et al.*, 2002; Hung and Balasingam, 2002; Yang E *et al.*, 2009; Banerjee and Agarwal, 2011). In addition, several artificial intelligence methods have been applied to the layout retargeting process for MB-OPC. Huang *et al.* (2006) proposed a radial basis function (RBF) network to predict optimal locations of the edges. Gu *et al.* (2008) used a linear regression model to estimate the approximate optimal mask edge movements.

However, no layout retargeting method has been proposed for ILT algorithms. Currently, ILT methods always employ the target patterns to initialize the optimization process. In this paper we propose a support vector machine (SVM) based layout retargeting method to generate a good initial input mask and accelerate the convergence for ILT. Unlike layout retargeting methods for traditional MB-OPC, our proposed layout retargeting method for ILT modifies the pixel values of the mask patterns, instead of the edge locations, to fit the pixel-based mechanism of ILT. As a supervised machine learning technique, SVM is very suitable to implement the pixel-based layout retargeting process, which treats generation values of pixels as a classification problem according to their contexts. Supervised by optimized masks of conventional ILT algorithms, several SVM models for different kinds of pixels are constructed and used to predict the pixel values in the 'undefined areas' of the new layout. By this process, an initial input mask close to the final optimized mask of the new layout is generated and reduces the number of iterations needed in the following optimization process in ILT.

Manufacturability is another critical issue in ILT research. Different regularization algorithms have been proposed to enhance the manufacturability of the mask generated by ILT (Poonawala and Milanfar, 2007b; Yu and Pan, 2007; Shen *et al.*, 2008; Ma and Li, 2011; Geng *et al.*, 2013; Lv *et al.*, 2013). Irregular patterns will be generated on the input mask after our layout retargeting due to the prediction inaccuracy of the SVM models. To compensate for this drawback, a spatial filter is applied onto the retargeted mask for complexity reduction.

As a widely used ILT algorithm, the regularized level-set based ILT (LSB-ILT) algorithm (Shen *et al.*, 2009) with a total variation (TV) penalty (Poonawala and Milanfar, 2007b; Geng *et al.*, 2013) is used to generate the training data set for SVM models in this

study. Note that other ILT algorithms and regularization terms are also suitable to our SVM based layout retargeting method. The whole process is implemented under the partial coherent illumination condition.

2 Preliminaries

2.1 Forward lithography model

The function Litho(·) representing the lithography system consists mainly of an optical model and a resist development model. The optical model of practical partially coherent illumination can be decomposed into a sum-of-coherent-systems (SOCS) model (Cobb and Zakhor, 1995; Cobb *et al.*, 1996) based on Hopkins' function (Hopkins, 1953). With the SOCS model, intensity of a single pixel (x, y) can be calculated by

$$\boldsymbol{I}(x,y;\boldsymbol{M})=\sum_{i=1}^{N}\lambda_i\,|\,\boldsymbol{K}_i\otimes\boldsymbol{M}\,|^2 =\sum_{i=1}^{N}\lambda_i[|\operatorname{Re}(\boldsymbol{K}_i)\otimes\boldsymbol{M}\,|^2+|\operatorname{Im}(\boldsymbol{K}_i)\otimes\boldsymbol{M}\,|^2], \quad (4)$$

where λ_i represents the ith weight value, $\boldsymbol{K}_i$ is the ith lithography kernel, $\operatorname{Re}(\boldsymbol{K}_i)$ and $\operatorname{Im}(\boldsymbol{K}_i)$ are the real and imaginary parts of $\boldsymbol{K}_i$ respectively, $\boldsymbol{M}$ is the mask matrix, and $\otimes$ is the convolution operator.

The resist development model used in this work is the constant threshold resist (CTR) model, whose input is light intensity $\boldsymbol{I}(x, y)$ and whose output is resist thickness. The CTR model can be described by the sigmoid function

$$\operatorname{sig}(\boldsymbol{I})=\frac{1}{1+\mathrm{e}^{-\alpha(\boldsymbol{I}-t_{\mathrm{r}})}}, \quad (5)$$

where α is the steepness factor and t_{r} is the threshold.

Therefore, the lithography model can be obtained by combining the optical model and the resist development model:

$$\operatorname{Litho}(\boldsymbol{M})=\operatorname{sig}(\boldsymbol{I}(\boldsymbol{M})). \quad (6)$$

2.2 Regularized level-set based ILT algorithm

When an L_2 norm based distance metric is used, the objective function in Eq. (3) of the optimization problem in ILT turns into

$$\boldsymbol{M}=\arg\min_{\boldsymbol{M}}\|\operatorname{Litho}(M(i,j))-T(i,j)\|^2, \quad (7)$$

where $\boldsymbol{M}$, $\boldsymbol{T}\in\mathbb{R}^{m\times n}$, $\boldsymbol{M}$ is the optimized mask matrix, and $\boldsymbol{T}$ represents the designed patterns or target patterns.

The level set method proposed by Osher and Sethian (1988) is a numerical technique for tracking interfaces and shapes, and has been used for ILT in recent research (Pang *et al.*, 2008; Shen *et al.*, 2009). In LSB-ILT algorithms, the transformation between mask matrix $\boldsymbol{M}$ and level-set matrix $\boldsymbol{\varphi}$ is introduced to convert the optimization problem to an unconstrained, continuous optimization problem, defined as

$$M(x,y)=\mathrm{B}(\boldsymbol{\varphi})=\begin{cases}1, & x:\varphi(x,y)>0,\\ 0, & x:\varphi(x,y)\le 0,\end{cases} \quad (8)$$

$$\varphi(x,y)=\begin{cases}-d(x,y), & (x,y)\in\boldsymbol{M}^-,\\ 0, & (x,y)\in\partial\boldsymbol{M},\\ d(x,y), & (x,y)\in\boldsymbol{M}^+,\end{cases} \quad (9)$$

where function $d(\cdot,\cdot)$ computes the distance between pixel (x, y) and the boundary of $\boldsymbol{M}$. $\boldsymbol{M}^-$, $\partial\boldsymbol{M}$, and $\boldsymbol{M}^+$ are the inner part, boundary, and outer part of mask pattern $\boldsymbol{M}$, respectively. After this transformation process, the objective function of LSB-ILT can be described as

$$\boldsymbol{\varphi}=\arg\min_{\boldsymbol{\varphi}}\|\operatorname{Litho}(\mathrm{B}(\boldsymbol{\varphi}))-\boldsymbol{T}\|^2. \quad (10)$$

For less complexity in the optimized mask, the regularization penalty, defined as $\operatorname{Reg}(\partial\boldsymbol{\varphi}/\partial x, \partial\boldsymbol{\varphi}/\partial y)$, is always added to ILT to control the regularity of $\boldsymbol{\varphi}$. The problem of regularized LSB-ILT can be formulated as follows:

$$\boldsymbol{\varphi}=\arg\min_{\boldsymbol{\varphi}}\left[\operatorname{cost}(\mathrm{B}(\boldsymbol{\varphi}))+\lambda_{\mathrm{Reg}}\operatorname{cost}_{\mathrm{Reg}}\left(\frac{\partial\boldsymbol{\varphi}}{\partial x},\frac{\partial\boldsymbol{\varphi}}{\partial y}\right)\right], \quad (11)$$

$$\operatorname{cost}(\mathrm{B}(\boldsymbol{\varphi}))=\|\operatorname{Litho}(\mathrm{B}(\boldsymbol{\varphi}))-\boldsymbol{T}\|^2, \quad (12)$$

$$\operatorname{cost}_{\mathrm{Reg}}\left(\frac{\partial\boldsymbol{\varphi}}{\partial x},\frac{\partial\boldsymbol{\varphi}}{\partial y}\right)=\int_{\Omega}\operatorname{Reg}\left(\frac{\partial\boldsymbol{\varphi}}{\partial x},\frac{\partial\boldsymbol{\varphi}}{\partial y}\right)\mathrm{d}x, \quad (13)$$

where λ_{Reg} is a user-defined weight of the regularization function $\text{cost}_{\text{Reg}}(\cdot,\cdot)$, and $\Omega=\{(x, y)|0\le x\le m, 0\le y\le n\}$ means the area of level-set matrix $\boldsymbol{\varphi}$. The TV penalty is used in this study for regularized LSB-ILT, described as

$$\text{Reg}_{\text{TV}}\left(\frac{\partial\boldsymbol{\varphi}}{\partial x},\frac{\partial\boldsymbol{\varphi}}{\partial y}\right)=\sqrt{\left(\frac{\partial\boldsymbol{\varphi}}{\partial x}\right)^2+\left(\frac{\partial\boldsymbol{\varphi}}{\partial y}\right)^2}. \quad (14)$$

A time-dependent scheme is employed to solve this optimization problem of LSB-ILT (Shen *et al.*, 2009), in which $\boldsymbol{\varphi}$ is optimized iteratively as

$$\boldsymbol{\varphi}_{k+1}(x)=\boldsymbol{\varphi}_k(x)+\frac{\partial\boldsymbol{\varphi}}{\partial t}\Delta t, \quad (15)$$

where $\frac{\partial\boldsymbol{\varphi}}{\partial t}$ is the speed function, and the $\frac{\partial\boldsymbol{\varphi}}{\partial t}$ of Eq. (11) is

$$\frac{\partial\boldsymbol{\varphi}}{\partial t}=-\alpha(\boldsymbol{M},t)|\nabla\boldsymbol{\varphi}|+\lambda_{\text{Reg}}\beta(\boldsymbol{\varphi},t)|\nabla\boldsymbol{\varphi}|, \quad (16)$$

where $\alpha(\boldsymbol{M}, t)$ is the Jacobian speed, and can be calculated as

$$\begin{aligned}\alpha(\boldsymbol{M},t)&=\frac{\partial}{\partial\boldsymbol{M}}(\text{sig}(\boldsymbol{I})-\boldsymbol{T})^2\\&=\sum_{i=1}^{N}\lambda_i\cdot a\big[\text{Re}(\boldsymbol{K}_i)\otimes(\boldsymbol{T}-\text{sig}(\boldsymbol{I}))\oplus\text{sig}(\boldsymbol{I})\\&\oplus(1-\text{sig}(\boldsymbol{I}))\oplus(\text{Re}(\boldsymbol{K}_i)\otimes\boldsymbol{M})+\text{Im}(\boldsymbol{K}_i)\otimes(\boldsymbol{T}-\text{sig}(\boldsymbol{I}))\\&\oplus\text{sig}(\boldsymbol{I})\oplus(1-\text{sig}(\boldsymbol{I}))\oplus(\text{Im}(\boldsymbol{K}_i)\otimes\boldsymbol{M})\big],\end{aligned} \quad (17)$$

where $\oplus$ is the dot multiplication operator, and $\beta(\boldsymbol{\varphi}, t)$ is the regularization speed. When the TV penalty is employed, $\beta(\boldsymbol{\varphi}, t)$ can be calculated as follows:

$$\beta(\boldsymbol{\varphi},t)=\nabla\cdot\left(\frac{\nabla\boldsymbol{\varphi}}{|\nabla\boldsymbol{\varphi}|}\right). \quad (18)$$

The convolution operations in LSB-ILT are implemented by a fast Fourier transform (FFT) for high speed, with a computational complexity of $O(N^2\log_2 N)$.

2.3 Support vector machine

SVM as a new type of data classifier based on a statistical learning technique has been successfully applied to a number of applications (Corinna and Vladimir, 1995; Byun and Lee, 2002). As a supervised machine learning algorithm, two stages including training and prediction are involved in SVM.

In the training process, training data with known classification results is used to train SVM models. To classify sample data, SVM constructs a hyperplane with the largest distance to the nearest training data point of any class; support vectors which are vectors on boundaries of this hyperplane are selected and weighted to define the model (Fig. 1).

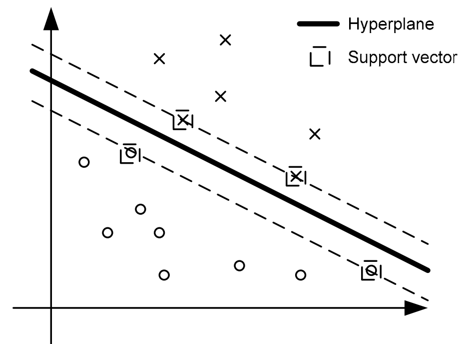

Fig. 1 Hyperplane constructed using SVM and support vectors on the boundaries
This example is shown under 2D and linear conditions. Note that SVM can be used to classify data under high-dimensional and nonlinear conditions

The training data set is presented as $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, in which $\boldsymbol{x}_i\in\mathbb{R}^d$ stands for the input vector, and $y_i\in\{-1, 1\}$ is the class label of the corresponding input vector. The problem of construction of the optimum hyperplane is a second-order program problem with constraints. The dual problem of this issue is given as follows:

$$\max_{\alpha}\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}y_iy_j\alpha_i\alpha_jk(\boldsymbol{x}_i,\boldsymbol{x}_j) \quad (19)$$

subject to

$$0\le\alpha_i\le C, \quad (20)$$

$$\sum_{i=1}^{m}\alpha_iy_i=0, \quad (21)$$

where α_i is the Lagrange coefficient, C is the soft

margin parameter, and $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the kernel function which can map the original data classification problem into high-dimensional space. The training process of the SVM model is just to find the solution of the optimization problem (19), with the output of the SVM model represented by support vectors and corresponding weights.

In the prediction process, the class of new data can be predicted using

$$f(\boldsymbol{x}) = \mathrm{sign}\left(\sum_{\boldsymbol{x}_i \in \mathrm{SV}} \alpha_i y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b\right), \qquad (22)$$

where $\boldsymbol{x}_i$ is a support vector in the SVM model, α_i and y_i are the corresponding weight and class label respectively, and bias b is computed according to the Karush-Kuhn-Tucker (KKT) conditions.

3 SVM based layout retargeting for ILT

The layout retargeting method proposed in this paper is designed to generate an initial mask close to the optimized mask so as to accelerate the convergence of the following optimization process of ILT. The input of this process is the target layout pattern, and the output is the retargeted mask close to the final optimized mask. For this purpose, SVM is employed to establish the relationship between the pixel values in the final optimized mask and target layout patterns. To train the SVM models, in this study we use optimized masks of training layout patterns generated by a regularized LSB-ILT algorithm to supervise the training process. The LSB-ILT algorithm in our earlier research (Geng *et al.*, 2013) is used here. The following subsections will elaborate on the individual steps and provide details of our layout retargeting method.

3.1 Identification of undefined areas

The concept of flipping in this work is defined as the value of a pixel on the optimized mask being different from the value of the same pixel on the target layout pattern. To reduce the pixels that need to be predicted by SVM, target layout patterns are divided into defined areas and undefined areas in our layout retargeting method. Defined areas are defined as those in which pixels always flip or do not flip; undefined areas are those in which pixels are not fixed on flipping. We identify undefined areas according to the training layouts and corresponding optimized masks; only the pixels in the undefined areas need to be predicted by the SVM models in our layout retargeting process.

For the ILT algorithm with a TV regularization penalty, we find that flipping always occurs in the areas near the edges of the target layout patterns (Poonawala and Milanfar, 2007b; Shen *et al.*, 2009; Geng *et al.*, 2013), which means that the undefined areas exist only in these areas. Therefore, three kinds of areas, including areas around the concave corner, areas around the convex corner, and areas around the line edge, are used to identify the undefined areas.

The concave corner area and convex corner area are represented by matrices $\boldsymbol{M}_{\text{concave}}$ and $\boldsymbol{M}_{\text{convex}}$, respectively, and the line edge area is represented by vector $\boldsymbol{V}_{\text{edge}}$ (Fig. 2). It is easy to prove that all pixels near the edges can be located in these three matrices through direction rotation. Sizes of these three matrices are user-defined parameters, which should guarantee that all areas where values of points change after the ILT process compared to their original value on the target layout pattern are covered by at least one of these three matrices. Sizes of the three matrices are all set to the min-feature size.

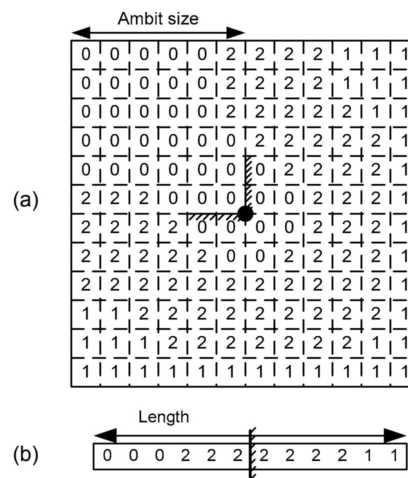

Fig. 2 Examples of concave corner area matrix $\boldsymbol{M}_{\text{concave}}$ (a) and line edge area vector $\boldsymbol{V}_{\text{edge}}$ (b)

Pixels with a value of 2 are the identified undefined areas; pixels with a value of 0 mean that pixel values in these points are all 0 in the optimized masks of the training layout; pixels with a value of 1 mean that pixel values in these points are all 1 in the optimized masks of the training layout

With the above definition, the optimized masks of the training layout patterns are scanned to identify the undefined areas in these three areas. Pseudo code of the undefined areas identification method is presented in Algorithm 1, with the undefined areas identified as pixels with a value of 2 in $\boldsymbol{M}_{\text{concave}}$, $\boldsymbol{M}_{\text{convex}}$, and $\boldsymbol{V}_{\text{edge}}$ (Fig. 2).

Algorithm 1 Identification of the undefined areas

Require: optimization results $\boldsymbol{M}_{\text{opt}}$ of the training layout, desired patterns $\boldsymbol{T}$ of the training layout

```
Set the ambit size of M_concave and M_convex, length of V_edge
Initialize elements of M_concave, M_convex, V_edge to −1
For each pixel T(x, y) in T
  If T(x, y) belongs to M_concave^a
    Calculate location (m, n) of T(x, y) in M_concave^b
    If M_concave(m, n) equals −1
      Set M_concave(m, n) to M_opt(x, y)
    Else if M_concave(m, n) is not equal to M_opt(x, y)
      Set M_concave(m, n) to 2
    Break
  End if
  …^c
End for
Return M_concave, M_convex, and V_edge
```

[a] A pixel belongs to the concave corner area if a concave corner exists in the area around this pixel with ambit of the concave matrix; the same is true for the convex corner area and line edge area. If a pixel belongs to more than one area, only one area is selected. $\boldsymbol{M}_{\text{concave}}$ has the highest priority and $\boldsymbol{V}_{\text{edge}}$ has the lowest priority in this process
[b] Direction rotation is considered in location calculation
[c] A similar process is applied to pixels belonging to $\boldsymbol{M}_{\text{convex}}$ and $\boldsymbol{V}_{\text{edge}}$

Note that the concept of undefined areas is also suitable to other ILT algorithms with different methods of identifying undefined areas.

3.2 Construction of SVM models

SVM treats the generation of binary pixel values as a classification problem according to their contexts. The context of a pixel is defined as the surrounding patterns of this pixel on a target layout; the radius of the clipping window used to capture the context is a user-defined value, defined as the ambit size of the lithography kernel in this work. To establish the relationship between the class of pixels and their corresponding contexts, three different SVM models are constructed for pixels in the undefined areas of $\boldsymbol{M}_{\text{concave}}$, $\boldsymbol{M}_{\text{convex}}$, and $\boldsymbol{V}_{\text{edge}}$, respectively. Construction of the SVM model for pixels in the undefined areas of $\boldsymbol{M}_{\text{concave}}$ is discussed in detail in this subsection. Construction of the other two models is similar to this process.

To supervise the training process of the SVM model, training data consisting of pixel values of the optimized mask and corresponding contexts on the training layout is sampled to construct the training data set. Note that only the pixels in the undefined areas of $\boldsymbol{M}_{\text{concave}}$ are sampled. The optimized mask of the training layout is generated by the regularized LSB-ILT algorithm here.

A concentric square sampling method proposed by Gu and Zakhor (2008) is used to create feature vectors for the contexts of the pixels (Fig. 3). Pixel values of different points are sampled at the four corners and the midpoint of each side of the squares that overlap the clipping window. The radii of the squares are 0, 2, 4, 6, …, R_1, R_1+4, R_1+8, …, R_2, R_2+8, R_2+16, …, R_c, where R_1 and R_2 control the sampling density and R_c is the radius of the clipping window. With this method, the training data set represented by $\{(\boldsymbol{v}_i, y_i)\}_{i=1}^{m}$ is generated, where $\boldsymbol{v}_i \in \mathbb{R}^d$ stands for the feature vector of the context and $y_i \in \{0, 1\}$ is the corresponding pixel value on the optimized mask.

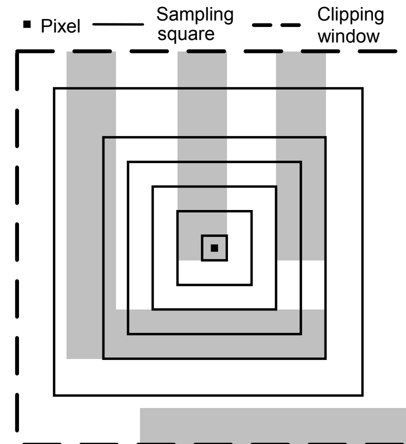

Fig. 3 Examples of the concentric square sampling method to encode the context of the pixel

With the training data set $\{(\boldsymbol{v}_i, y_i)\}_{i=1}^{m}$, a sequential minimal optimization (SMO) algorithm (Platt, 1998) is used to construct the SVM model for the undefined areas in $\boldsymbol{M}_{\text{concave}}$ (Algorithm 2). The Gaussian radial basis function (RBF), which can map the original data into infinite-dimensional space, is used as the kernel function:

$$k(\boldsymbol{v}_i, \boldsymbol{v}_j) = \exp(-r \| \boldsymbol{v}_i - \boldsymbol{v}_j \|^2). \tag{23}$$

Algorithm 2 Training the SVM model by SMO

Require: training data set $\{\boldsymbol{V}, y\}$

Set parameters C and r, stopping tolerance ϵ=1e-3
Initialize weight vector $\boldsymbol{\alpha}$ and error vector $\boldsymbol{E}$, pre-computed self dot product vector $\boldsymbol{M}$ of vectors in $\boldsymbol{V}$
While 1 **do**
  Select optimization pair ($\boldsymbol{V}_i$, $\boldsymbol{V}_j$) based on a heuristic algorithm according to KKT conditions
  If selection fails
    Break
  Calculate upper bound H and lower bound L of $\boldsymbol{\alpha}_j$
  Calculate $\eta=2*k(\boldsymbol{V}_i, \boldsymbol{V}_j)-k(\boldsymbol{V}_i, \boldsymbol{V}_i)-k(\boldsymbol{V}_j, \boldsymbol{V}_j)$
  Store α_j in temp
  Update weight α_j+=$y_j*(E_j-E_i)/\eta$
  Adjust α_j according to the upper and lower bounds
  Update weight α_i+=$-y_i*y_j(\alpha_j-\text{temp})$
  Update bias b and error vector $\boldsymbol{E}$
End while
Return the SVM model consisting of support vector set ($\boldsymbol{V}_s$, y_s) and corresponding weight set α_s

3.3 Pixel-based layout retargeting and complexity reduction

When the training process is completed, the SVM models are used in the layout retargeting process to generate the initial input mask for regularized LSB-ILT. For a layout mask that needs to be retargeted, values of pixels in the undefined areas of this layout are predicted in the SVM prediction process according to their contexts. For a pixel with context encoded as $\boldsymbol{v}$, the pixel value is predicted using the following formula:

$$f(x)=\frac{1}{2}\left[\operatorname{sign}\left(\sum_{\boldsymbol{v}_i\in\text{SV}}\alpha_i y_i k(\boldsymbol{v}_i,\boldsymbol{v})+b\right)+1\right], \tag{24}$$

where SV is the support vector set of the corresponding SVM model, and $k(\boldsymbol{v}_i, \boldsymbol{v})$ is the RBF kernel function.

Pixels not in the undefined areas are divided into two conditions: pixels belonging to $\boldsymbol{M}_{\text{concave}}$, $\boldsymbol{M}_{\text{convex}}$, and $\boldsymbol{V}_{\text{edge}}$ are set to the same values of corresponding locations in $\boldsymbol{M}_{\text{concave}}$, $\boldsymbol{M}_{\text{convex}}$, and $\boldsymbol{V}_{\text{edge}}$, respectively; pixels not belonging to $\boldsymbol{M}_{\text{concave}}$, $\boldsymbol{M}_{\text{convex}}$, and $\boldsymbol{V}_{\text{edge}}$ are set to 0 for a dark-field mask. The pseudo code of the pixel-based layout retargeting method is illustrated in Algorithm 3. When the SVM models are defined, the runtime of the layout retargeting process is proportional to the area of the layout pattern and has a computational complexity of $O(N^2)$.

Algorithm 3 Pixel-based layout retargeting

Require: Desired patterns $\boldsymbol{T}$ of the training layout, SVM models for $\boldsymbol{M}_{\text{concave}}$, $\boldsymbol{M}_{\text{convex}}$, and $\boldsymbol{V}_{\text{edge}}$

Set $\boldsymbol{M}_{\text{retargeted}}$ the same size as $\boldsymbol{T}$
Initialize elements of $\boldsymbol{M}_{\text{retargeted}}$ to 0
For each pixel $T(x, y)$ in $\boldsymbol{T}$
  If $T(x, y)$ belongs to $\boldsymbol{M}_{\text{concave}}$ [a]
    Calculate location (m, n) of $T(x, y)$ in $\boldsymbol{M}_{\text{concave}}$ [b]
    If $M_{\text{concave}}(m, n)$ equals 2
      Encode context of $T(x, y)$ as $\boldsymbol{v}$
      Predict $M_{\text{retargeted}}(x, y)$ as in Eq. (24) with the SVM model for $\boldsymbol{M}_{\text{concave}}$
    Else
      Set $M_{\text{retargeted}}(x, y)$ to $M_{\text{concave}}(m, n)$
    Break
  End if
  … [c]
End for
Return $\boldsymbol{M}_{\text{retargeted}}$

[a] A pixel belongs to the concave corner area if a concave corner exists in the area around this pixel with the ambit of the concave matrix; the same is true for the convex corner area and line edge area
[b] Direction rotation is considered in location calculation
[c] A similar process is applied to pixels belonging to $\boldsymbol{M}_{\text{convex}}$ and $\boldsymbol{V}_{\text{edge}}$

Due to the prediction inaccuracy of SVM models, many irregular patterns will be generated on the retargeted mask by our retargeting process (Fig. 4), which will increase the complexity of the final optimized mask of ILT. To compensate for this drawback, a spatial filter is applied to the retargeted mask for complexity reduction. The four pixels around pixel $\boldsymbol{p}$ on the perpendicular directions are defined as 4-neighbors of $\boldsymbol{p}$ (Fig. 5).

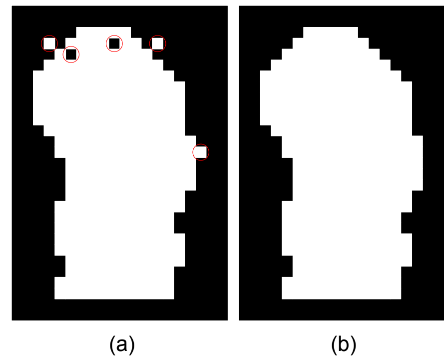

Fig. 4 Examples of a line-end on the retargeted mask after our pixel-based retargeting process: (a) before filtering; (b) after filtering

The spatial filter is designed to remove pixels with less than two same values in all 4-neighbors. The

filtering process can be described as

$$\boldsymbol{M}_{\mathrm{f}} = \{[\mathrm{sign}(\mathrm{abs}(\boldsymbol{M} \otimes \boldsymbol{K}) - 2) + 1] / 2\} \oplus \boldsymbol{M}, \quad (25)$$

where $\boldsymbol{M}_{\mathrm{f}}$ is the filtered mask, $\boldsymbol{M}$ is the retargeted mask before filtering, $\otimes$ is the convolution operator, $\oplus$ is the XOR operator, and $\boldsymbol{K}$ is the spatial filter kernel shown in Fig. 5. In Eq. (25), sign(x)=−1 if x=0.

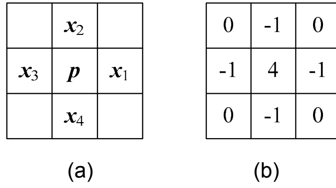

Fig. 5 Definition of 4-neighbors of pixel *p* (a) and spatial filter kernel *K* (b)

4 Experiments and discussions

We implemented the proposed layout retargeting method and regularized LSB-ILT algorithm in Matlab. The training process of the SVM models was implemented in C/C++. Experiments were designed and simulated to make a comparison between the regularized LSB-ILT algorithm with and without our layout retargeting method. Once established, SVM models can be reused for different layout patterns without rebuilding. The runtime of the training process was not considered in this comparison.

Different layout patterns of layout designs at the 36 nm node and a lithography model with λ=193 nm, NA=1.2, and quad-crescent illumination were used and tested. All simulations and testing were performed on a Dell PowerEdge R610 workstation with Xeon 2.8 GHz CPU and 32 GB memory.

The pixel grid was set to 3 nm×3 nm in both the retargeting process and ILT. Layout patterns involved in the experiments were divided into three groups: simple layout patterns (SLP), medium layout patterns (MLP), and large layout patterns (LLP).

Three layout patterns from the three groups respectively were chosen as training layout patterns and used in the training process of the SVM models. Parameters used in the training process were set as follows: the ambit size of $\boldsymbol{M}_{\mathrm{concave}}$ and $\boldsymbol{M}_{\mathrm{convex}}$ was 6, and the length of $\boldsymbol{V}_{\mathrm{edge}}$ was 12; the size of the clipping window to capture context was 256×256, the same as the size of the lithography kernel; R_1 in the concentric square sampling method was 20 and R_2 was 80, with R_{c} equal to 128; The size of the encoded context was 249. The regularized LSB-ILT with the same parameters was used to generate optimized masks for the three training layout patterns, and λ_{Reg} was set to 0.1. After identification of the undefined areas, the training data was sampled from the undefined areas of $\boldsymbol{M}_{\mathrm{concave}}$, $\boldsymbol{M}_{\mathrm{convex}}$, and $\boldsymbol{V}_{\mathrm{edge}}$ on the three layouts and used to construct the SVM models for $\boldsymbol{M}_{\mathrm{concave}}$, $\boldsymbol{M}_{\mathrm{convex}}$, and $\boldsymbol{V}_{\mathrm{edge}}$, respectively.

The layout patterns, other than those used in the training process, were used to test our layout retargeting method. We compared the number of iterations needed between the regularized LSB-ILT with and without our layout retargeting method in this experiment. In the regularized LSB-ILT process without layout retargeting, the target layout patterns were used as the initial input mask for optimization. In the regularized LSB-ILT process with our layout retargeting method, the initial input mask was the retargeted mask after spatial filtering.

The ILT process with and without layout retargeting both had the same parameters as the training layout patterns. ILT without the retargeting process was stopped after 250 iterations; ILT with the retargeting process was stopped at the same cost as the cost generated by the 250th iteration of ILT without the retargeting process, where cost was cost(B($\boldsymbol{\varphi}$)) as shown in Eq. (12). For fair comparison, the Matlab operation was constrained to one CPU. Table 1 shows the efficiency comparison results. With our layout retargeting method, the number of iterations needed in the optimization process was reduced by 70.8% on average, and the runtime of the whole process was reduced by 69.0% on average. Compared to the optimization process, the runtime of the layout retargeting process was very small; moreover, note that the high parallel computation is easy to apply in the proposed layout retargeting process.

The RECTs is used as a metric of mask complexity in this work, expressed as (Ma and Li, 2011)

$$\mathrm{RECTs} = \sum_{i=1}^{K} \left(\frac{3}{4} \mathrm{concave}_i + \frac{1}{4} \mathrm{convex}_i \right), \quad (26)$$

Table 1 Performance comparison between regularized LSB-ILT with and without our layout retargeting method

Target	Area (μm^2)	Iters[a]	Iters[b]	ε	Rtime[a] (s)	$Rtime_{re}$ (s)	$Rtime_{ilt}$ (s)	Rtime[b] (s)	η
SLP1	0.41	250	67	73.2%	1271	14.20	354	368	71.0%
SLP2	0.45	250	64	74.4%	1392	16.06	359	375	73.1%
SLP3	0.86	250	60	76.0%	2731	28.40	677	705	74.2%
SLP4	0.94	250	71	71.6%	2954	31.78	863	895	69.7%
SLP5	1.34	250	64	74.4%	4259	46.41	1118	1164	72.7%
MLP1	3.96	250	79	68.4%	12482	119.60	3822	3942	68.4%
MLP2	4.95	250	62	75.2%	15478	168.22	3977	4145	73.2%
MLP3	5.28	250	80	68.0%	16381	181.36	5337	5518	66.3%
MLP4	4.80	250	73	70.8%	15230	143.25	4643	4786	68.6%
LAP1	10.31	250	87	65.2%	31869	367.12	11366	11733	63.2%
LAP2	9.63	250	83	66.8%	30018	328.15	10724	11052	63.2%
LAP3	13.52	250	85	66.0%	41391	471.42	14504	14975	63.8%
Average				70.8%					69.0%

Iters represents the number of iterations needed to reach the stop criterion; ε is the reduction rate of the number of iterations with layout retargeting. Rtime is the runtime of the whole process; $Rtime_{re}$ is the runtime of the retargeting process; $Rtime_{ilt}$ is the runtime of the ILT process with the retargeted mask as input; η is the reduction rate of the runtime of the whole process with layout retargeting. [a] Regularized LSB-ILT without layout retargeting; [b] regularized LSB-ILT with layout retargeting

where K is the number of all polygons of the mask, $concave_i \in \mathbb{N}^+$ denotes the number of all the concave vertices of one polygon, and $convex_i \in \mathbb{N}^+$ denotes the number of all the convex vertices of one polygon. RECTs represents the total number of rectangles.

Table 2 shows the complexity results of the optimized layout patterns generated by different processes. The optimized masks of the process with and without layout retargeting had almost the same complexity, all falling in the range of (−3%, +3%).

Fig. 6 displays the performance comparison of the optimized results of the target pattern of SLP5 between the process with and without retargeting. Fig. 6a is the target layout pattern. The layout pattern after retargeting and spatial filtering of SLP5 is presented in Fig. 6b, and the optimized layout patterns generated by regularized LSB-ILT with and without layout retargeting are shown in Figs. 6c and 6d, respectively. The corresponding simulation results of these layout patterns are presented in Figs. 6e–6h. We can find that the pattern fidelity has been greatly improved after the layout retargeting process and is close to the final optimized mask.

Fig. 7 shows the cost evolution curves of SLP5 with the regularized LSB-ILT process with and without layout retargeting, where cost is $\text{cost}(\text{B}(\boldsymbol{\varphi}))$ as shown in Eq. (12), computed by counting how many pixels are different between the target pattern and the simulated on-wafer pattern. The initial costs of ILT without and with retargeting were 11977 and 4787 respectively, and the final optimized cost was 4599. We can find that a much smaller initial cost, very close to the final optimized cost, was obtained using our layout retargeting process.

Table 2 Complexity comparison between regularized LSB-ILT with and without our layout retargeting method

Target	RECTs[a]	RECTs[b]	ξ
SLP1	282	275	−2.48%
SLP2	287	289	+0.70%
SLP3	520	514	−1.15%
SLP4	611	628	+2.78%
SLP5	829	843	+1.69%
MLP1	2694	2751	+2.11%
MLP2	3746	3735	−0.30%
MLP3	3293	3305	+0.36%
MLP4	3242	3300	+0.18%
LAP1	10851	10746	−0.97%
LAP2	8344	8372	+0.34%
LAP3	13944	14062	+0.85%

[a] Regularized LSB-ILT without layout retargeting; [b] regularized LSB-ILT with layout retargeting. ξ is the complexity difference rate of processes with and without layout retargeting

In practice, to achieve better performance, we can use different choices of the areas to identify the undefined areas, select more accurate appearances

and sizes for these areas, and make a compromise between the runtime of layout retargeting and the number of iterations of the following optimization process. In addition, different methods for encoding the context can be used and can provide balance in prediction accuracy and runtime.

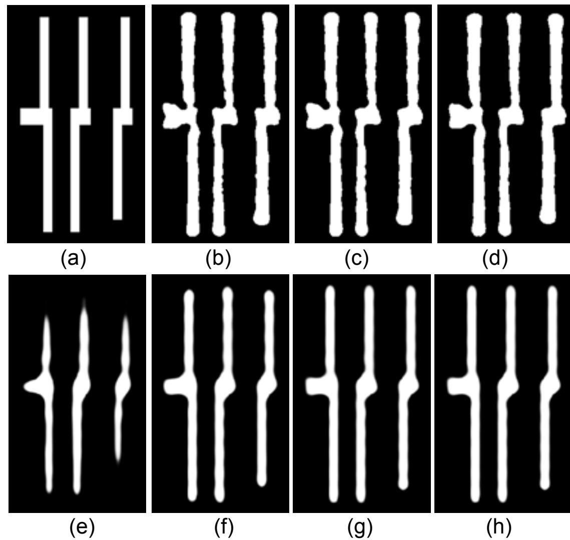

Fig. 6 Performance comparison of the optimized results of the target pattern of SLP5 between the process with and without retargeting

(a) Target layout; (b) Retargeted mask after filtering; (c) Optimized mask by ILT with layout retargeting; (d) Optimized mask by ILT without layout retargeting. (e)–(h) are the corresponding simulation results of (a)–(d), respectively

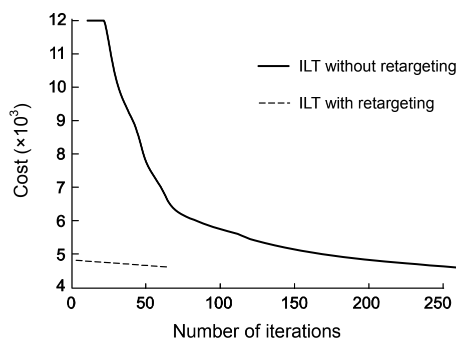

Fig. 7 Cost evolution curves of ILT with and without the layout retargeting process for target SLP5

5 Conclusions

In this paper, an SVM based layout retargeting method for ILT is proposed for fast convergence. Supervised by optimization results of conventional ILT algorithms, the SVM models are constructed and used to generate the initial input mask, which is close to the final optimized mask, for the optimization process in ILT. The concept of 'undefined areas' is proposed to reduce the runtime of the layout retargeting process. For less complexity in the final optimized mask, a spatial filter is applied to the retargeted mask to filter irregular patterns. Experiments designed and tested under the 36 nm node and partial coherent illumination conditions showed that with our layout retargeting method, the number of iterations needed in the optimization process and runtime of the whole process in ILT are reduced by 70.8% and 69.0% respectively, without increase of mask complexity.

References

Banerjee, S., Agarwal, K.B., 2011. Integrated model-based retargeting and optical proximity correction. *SPIE*, **7974**: 79740F. [doi:10.1117/12.879531]

Byun, H., Lee, S.W., 2002. Pattern Recognition with Support Vector Machine. Springer, Berlin, Heidelberg, p.571-591.

Chen, Y., Wu, K., Shi, Z., *et al.*, 2007. A feasible model-based OPC algorithm using Jacobian matrix of intensity distribution functions. *SPIE*, **6520**:65204C. [doi:10.1117/12.711763]

Chiang, C., Kawa, J., 2007. Design for manufacturability and yield for nano-scale CMOS. Series on Integrated Circuits and Systems. Springer, Dordrecht, The Netherlands, p.58-72. [doi:10.1007/978-1-4020-5188-3]

Cobb, N.B., Zakhor, A., 1995. Fast sparse aerial image calculation for OPC. *SPIE*, **2621**:534-545. [doi:10.1117/12.228208]

Cobb, N.B., Zakhor, A., Miloslavsky, E.A., 1996. Mathematical and CAD framework for proximity correction. *SPIE*, **2726**:208-222. [doi:10.1117/12.240907]

Corinna, C., Vladimir, V., 1995. Support-vector networks. *Mach. Learn.*, **20**(3):273-297. [doi:10.1007/BF00994 018]

Erdmann, A., Farkas, R., Fuhner, T., *et al.*, 2004. Towards automatic mask and source optimization for optical lithography. *SPIE*, **5377**:646-657. [doi:10.1117/12.533215]

Garofalo, J., Low, K., Otto, O., *et al.*, 1994. Automatic proximity correction for 0.35 μm I-line photolithography. Proc. IEEE Int. Workshop on Numerical Modeling of Processes Devices for Integrated Circuits, p.92-94. [doi:10.1109/NUPAD.1994.343483]

Geng, Z., Shi, Z., Yan, X.L., *et al.*, 2013. Regularized level-set-based inverse lithography algorithm for IC mask synthesis. *J. Zhejiang Univ.-Sci. C* (*Comput. & Electron.*), **14**(10):799-807. [doi:10.1631/jzus.C1300050]

Granik, Y., 2005. Solving inverse problems of optical microlithography. *SPIE*, **5754**:506-526. [doi:10.1117/12.600

141]
Granik, Y., 2006. Fast pixel-based mask optimization for inverse lithography. *J. Micro/Nanolith. MEMS MOEMS*, **5**(4):043002. [doi:10.1117/1.2399 537]
Gu, A., Zakhor, A., 2008. Optical proximity correction with linear regression. *IEEE Trans. Semicond. Manuf.*, **21**(2): 263-271. [doi:10.1109/TSM.2008.2000283]
Hopkins, H.H., 1953. On the diffraction theory of optical images. *Proc. R. Soc. Lond. A*, **217**(1130):408-432. [doi:10.1098/rspa.1953.0071]
Huang, W.C., Lai, C.M., Luo, B., *et al.*, 2006. Intelligent model-based OPC. *SPIE*, **6154**:615436. [doi:10.1117/12.657792]
Hung, M., Balasingam, P., 2002. Hybrid optical proximity correction: concepts and results. *SPIE*, **4889**:1173-1180. [doi:10.1117/12.468204]
ITRS, 2012. International Technology Roadmap for Semiconductors 2012 Update Overview. ITRS. Available from http://www.itrs.net/Links/2012ITRS/2012Chapters/2012Overview.pdf [Accessed on Sept. 1, 2013].
Kotani, T., Kobayashi, S., Ichikawa, H., *et al.*, 2002. Advanced hybrid optical proximity correction system with OPC segment library and model-based correction module. *SPIE*, **4691**:188-195. [doi:10.1117/12.474546]
Lin, B., Yan, X.L., Shi, Z., *et al.*, 2011. A sparse matrix model-based OPC algorithm with model-based mapping between segments and control sites. *J. Zhejiang Univ.-Sci. C* (*Comput. & Electron.*), **12**(5):436-442. [doi:10.1631/jzus.C1000219]
Lv, W., Xia, Q., Liu, S., 2013. Pixel-based inverse lithography using a mask filtering technique. *SPIE*, **8683**:868325. [doi:10.1117/12.2011469]
Ma, X., Arce, G., 2007. Generalized inverse lithography methods for phase-shifting mask design. *Opt. Expr.*, **15**(23):15066-15079. [doi:10.1364/OE.15.015066]
Ma, X., Arce, G., 2008. Binary mask optimization for inverse lithography with partially coherent illumination. *J. Opt. Soc. Am. A*, **25**(12):2960-2970. [doi:10.1364/JOSAA.25.002960]
Ma, X., Li, Y.Q., 2011. Resolution enhancement optimization methods in optical lithography with improved manufacturability. *J. Micro/Nanolith. MEMS MOEMS*, **10**(2): 023009. [doi:10.1117/1.3590252]
Ma, X., Li, Y.Q., Dong, L.S., 2012a. Mask optimization approaches in optical lithography based on a vector imaging model. *J. Opt. Soc. Am. A*, **29**(7):1300-1312. [doi:10.1364/JOSAA.29.001300]
Ma, X., Li, Y.Q., Guo, X.J., *et al.*, 2012b. Vectorial mask optimization methods for robust optical lithography. *J. Micro/Nanolith. MEMS MOEMS*, **11**(4):043008. [doi:10.1117/1.JMM.11.4.043008]
Oh, Y., Lee, J.C., Lim, S., 1999. Resolution enhancement through optical proximity correction and stepper parameter optimization for 0.12-μm mask pattern. *SPIE*, **3679**:607-613. [doi:10.1117/12.354373]
Osher, S., Sethian, J.A., 1988. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, **79**(1):12-49. [doi:10.1016/0021-9991(88)90002-2]
Pang, L.Y., Dai, G., Cecil, T., *et al.*, 2008. Validation of inverse lithography technology (ILT) and its adaptive SRAF at advanced technology nodes. *SPIE*, **6924**:69240T. [doi:10.1117/12.775084]
Park, J., Park, C., Phie, S., *et al.*, 2000. An efficient rule-based OPC approach using DRC tool for 0.18 μm ASIC. Proc. IEEE 1st Int. Symp. on Quality Electronic Design, p.81-85. [doi:10.1109/ISQED.2000.838858]
Platt, J.C., 1998. Sequential Minimal Optimization: a Fast Algorithm for Training Support Vector Machines. Technical Report MsR-TR-98-14, Microsoft Research, Microsoft Inc., Redmond, WA.
Poonawala, A., Milanfar, P., 2007a. Mask design for optical microlithography—an inverse imaging problem. *IEEE Trans. Image Process.*, **16**(3):774-778. [doi:10.1109/TIP.2006.891332]
Poonawala, A., Milanfar, P., 2007b. A pixel-based regularization approach to inverse lithography. *Microelectro. Eng.*, **84**(12):2837-2852. [doi:10.1016/ j.mee.2007.02.005]
Shen, S.H., Peng, Y., Pan, D.Z., 2008. Enhanced DCT2-based inverse mask synthesis with initial SRAF insertion. *SPIE*, **7122**:712241. [doi:10.1117/12.801409]
Shen, Y.J., Wong, N., Lam, E.Y., 2009. Level-set-based inverse lithography for photomask synthesis. *Opt. Expr.*, **17**(26): 23690-23701. [doi:10.1364/OE.17.023690]
Wong, A.K.K., 2001. Resolution Enhancement Techniques in Optical Lithography. SPIE Press, Bellingham, Washington, USA, p.28. [doi:10.1117/3.401208]
Yang, E., Li, C.H., Kang, X.H., *et al.*, 2009. Model-based retarget for 45nm node and beyond. *SPIE*, **7274**:727428. [doi: 10.1117/12.814085]
Yang, Y.W., Shi, Z., Shen, S.H., 2009. Seamless-merging-oriented parallel inverse lithography technology. *J. Semicond.*, **30**(10):106002-106006. [doi:10.1088/1674-4926/30/10/106002]
Yu, P., Pan, D.Z., 2007. TIP-OPC: a new topological invariant paradigm for pixel based optical proximity correction. Proc. IEEE/ACM Int. Conf. on Computer-Aided Design, p.847-853. [doi:10.1109/ICCAD.2007.4397370]