A Feasible Model-Based OPC Algorithm Using Jacobian Matrix of Intensity Distribution Functions

Ye Chen[a], Kechih Wu[b], Zheng Shi[a] and Xiaolang Yan[a]

[a]Institute of VLSI Design, Zhejiang University Hangzhou, 310027, China
{chenye, shiz, yan}@vlsi.zju.edu.cn
[b]Anchor Semiconductor, Inc. 1090 Kifer Road, Suite 200 Sunnyvale, CA 94086
kechihwu@anchorsemi.com

ABSTRACT

The correction accuracy of a model-based OPC (MB-OPC) depends critically on its edge offset calculation scheme. In a normal MB-OPC algorithm, only the impact of the current edge is considered in calculating each edge offset. As the k1 process factor decreases and design complexity increases, however, the interaction between the edge segments becomes much larger. As a result, the normal MB-OPC algorithm may not always converge or converge slowly. Controlling the EPE is thus become harder. To address this issue, a new kind of MB-OPC algorithm based on MEEF matrix was introduced which is also called matrix OPC. In this paper, a variant of such matrix OPC algorithm is proposed which is suitable for kernel-based lithography models. Comparing with that based on MEEF matrix, this algorithm requires less computation in matrix construction. Sparsity control scheme and RT reuse scheme are also used to make the correction speed be close to a normal one while keeping its advantages on EPE control.

Keywords: OPC, Convergence, Matrix, Low-k1 lithography

1. INTRODUCTION

Model-based OPC (MB-OPC) algorithms usually can be divided into two steps, namely fragmentation step and edge offset calculation step. In fragmentation step, edges in layout are broken into small segments, an evaluation point and an offset direction are chosen for each edge segment. In edge offset calculation step, edges are moved along the offset direction to fill the gaps between original patterns and simulation results. The correction accuracy of an MB-OPC algorithm depends on both steps. As the k1 process factor decreases and design complexity increases, the interactions between edge segments become much larger. In this case, normal edge offset calculation method may face convergence problem,[1 2 3] which means that some segments in the layout are moved to their proper positions very slowly or even not moved to the right positions at all. Figure 1 shows an example in which the line end is not convergent at all. Figure 2 shows another example, the edge position oscillates between each two iterations and converges very slowly.

To improve OPC convergence, a new kind of MB-OPC algorithm was introduced which takes into account the interaction between the edges by using the mask error enhancement factor (MEEF) matrix.[4] Although better convergence and tight EPE control has been reported, matrix OPC algorithms have not been widely adopted in production due to the performance penalty.

In this paper, we propose a new algorithm, suitable for kernel-based lithography models, that will significantly speed up the matrix OPC process. The key aspects of the new algorithm are as follows:

- evaluating intensity distribution function instead of EPE function
- evaluating Jacobian matrix of intensity distribution functions instead of MEEF matrix
- controlling the sparsity of Jacobian matrix to reduce the time spent both on generating Jacobian matrix and solving the linear equations
- cutting layout into small pieces to deal with the super-linear complexity of solving linear equations

Optical Microlithography XX, edited by Donis G. Flagello, Proc. of SPIE Vol. 6520, 65204C, (2007)
0277-786X/07/$18 · doi: 10.1117/12.711763

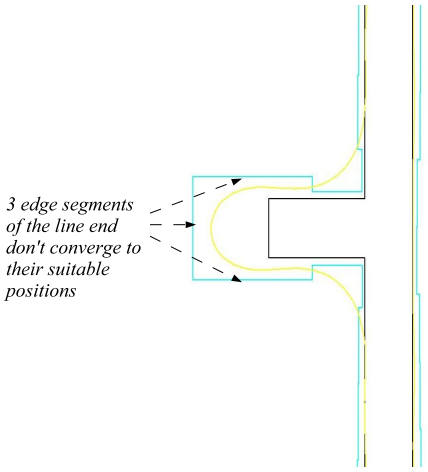


Figure 1. Edge segments don't converge after OPC

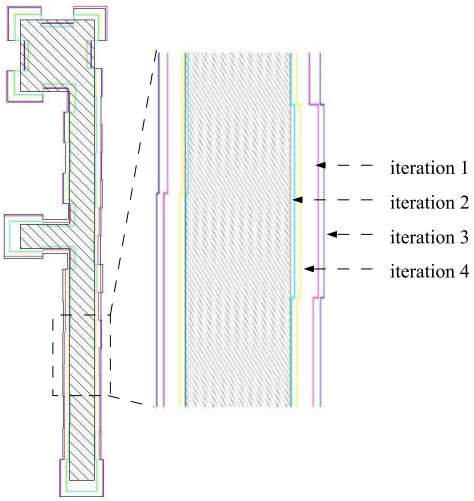
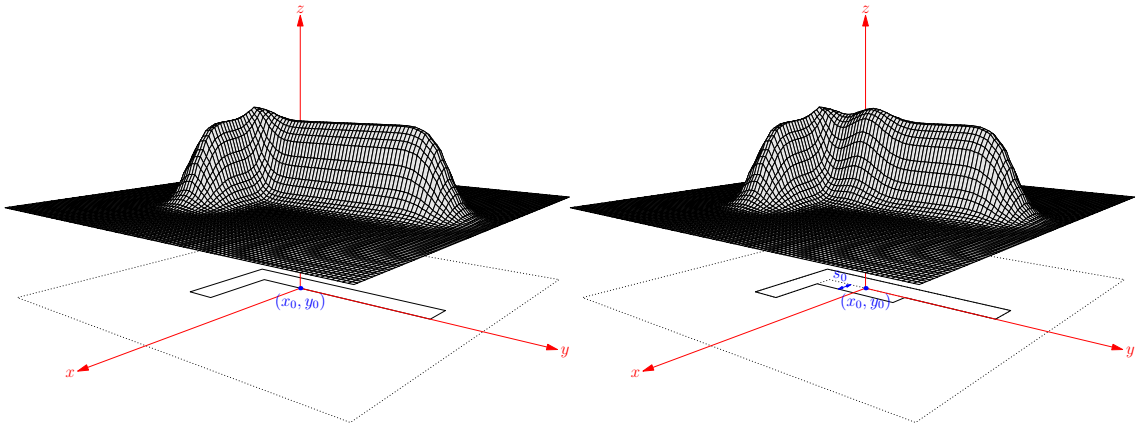


Figure 2. OPC result oscillates, segments converge very slowly



(a) (b)

Figure 3. Generalized intensity distribution function

- reusing information from neighbor areas to reduce the size of linear equations and get good initial values

Section 2 describes the methodology. Experimental results are presented in Section 3 followed by the conclusions in Section 4.

2. METHODOLOGY

2.1. Generalized intensity distribution function

Figure 3(a) shows the intensity distribution $I(x, y; mask)$ as a function of wafer coordinates and mask patterns. During OPC process, modification of mask patterns consists of the offsets of all the edge segments, which

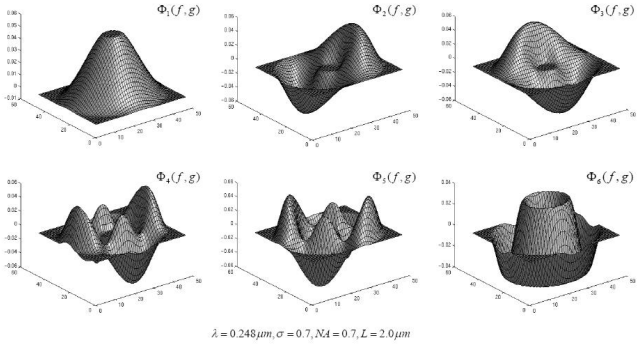


Figure 4. Typical kernel shapes in an optical lithography model

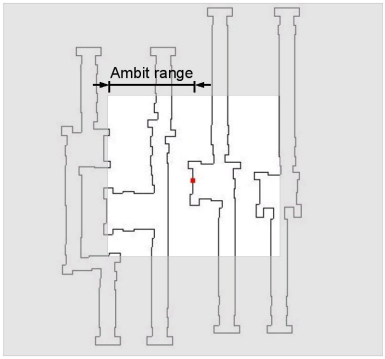


Figure 5. "Ambit" window

also change the intensity distribution. For the rest of the paper, we will refer the intensity distribution function with edge-shift denotation as the generalized intensity distribution function. For example, the intensity distribution in Figure 3(b) is denoted as $I(x, y; s_0)$.

2.2. Problem definition

Suppose an already fragmented layout has a total of n elements with evaluation points of $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ and the offsets of these segments are denoted as $s_1, s_2, \ldots, s_n$. The generalized intensity distribution function is $I(x, y; s_1, s_2, \ldots, s_n)$. The objective of the OPC process is to generate a set of $(s_1, s_2, \ldots, s_n)$ so as to reduce the edge placement error (EPE) of each segment to zero. This is equivalent to making the value of generalized intensity distribution function at each evaluation point equal to its threshold:

$$I(x_i, y_i; s_1, s_2, \ldots, s_n) = Tr_i \tag{1}$$

where Tr_i is the threshold value at location (x_i, y_i). For all the segments, a system of equations can be built as:

$$\begin{cases} I(x_1, y_1; s_1, s_2, \ldots, s_n) - Tr_1 = 0 \\ I(x_2, y_2; s_1, s_2, \ldots, s_n) - Tr_2 = 0 \\ \ldots \\ I(x_n, y_n; s_1, s_2, \ldots, s_n) - Tr_n = 0 \end{cases} \tag{2}$$

$Tr_1, Tr_2, \ldots, Tr_n$ are all the same when a constant threshold resist (CTR) model is used.[5] However, they are functions of nearby patterns and the aerial image properties, such as intensity, intensity slope and so forth when variable threshold resist (VTR) model is used.[6]

2.3. Solving the equations

Least square solution of Equation 2 is the best estimation of offsets which can cancel out distortion the most. Plenty of algorithms can be used to solve the nonlinear equations. For example, a Newton's method makes each iteration as:

1. solving the system of linear equations

$$J(s_k)d_k = -(I(s_k) - Tr(s_k)) \tag{3}$$

where $s_k, d_k \in R^n$, $J(s_k)$ means the Jacobian matrix of Equations 2 at s_k.

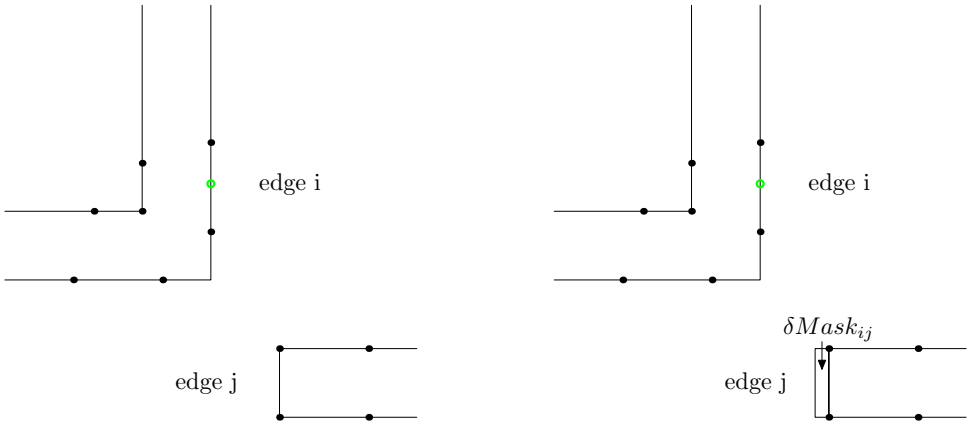


Figure 6. Jacobian matrix evaluation: (1) Calculate f_{ki} (2) Perturb edge j, calculate δf_{kij} (3) Calculate $J_I(i,j)$ using Equation 8

2. updating offsets

$$s_{k+1} = s_k + d_k \tag{4}$$

The most time-consuming parts of solving the problem are:

- evaluation of the function
- evaluation of the Jacobian matrix
- solving the linear system

2.4. Intensity distribution function evaluation

The optical lithography model used in OPC applications is usually based on a group of convolution kernels. The point intensity at (x, y) can be calculated by:

$$I(x,y) = \sum_k |\Phi_k \otimes Mask|^2 \tag{5}$$

where Φ_k are convolution kernels and "Mask" are mask patterns in the "ambit" window of point (x, y). Figure 4 shows a group of convolution kernels. Figure 5 shows an example of "ambit" window. Patterns in the "ambit" window contribute to the point intensity, while those outside don't.[7]

2.5. Jacobian matrix evaluation

Jacobian matrix of Equation 2 is defined in Equation 6.

$$J = \begin{bmatrix} \frac{\partial I_1}{\partial s_1} & \frac{\partial I_1}{\partial s_2} & \cdots & \frac{\partial I_1}{\partial s_n} \\ \frac{\partial I_2}{\partial s_1} & \frac{\partial I_2}{\partial s_2} & \cdots & \frac{\partial I_2}{\partial s_n} \\ \vdots & & & \\ \frac{\partial I_n}{\partial s_1} & \frac{\partial I_n}{\partial s_2} & \cdots & \frac{\partial I_n}{\partial s_n} \end{bmatrix} + \begin{bmatrix} \frac{\partial Tr_1}{\partial s_1} & \frac{\partial Tr_1}{\partial s_2} & \cdots & \frac{\partial Tr_1}{\partial s_n} \\ \frac{\partial Tr_2}{\partial s_1} & \frac{\partial Tr_2}{\partial s_2} & \cdots & \frac{\partial Tr_2}{\partial s_n} \\ \vdots & & & \\ \frac{\partial Tr_n}{\partial s_1} & \frac{\partial Tr_n}{\partial s_2} & \cdots & \frac{\partial Tr_n}{\partial s_n} \end{bmatrix} = J_I + J_{Tr} \tag{6}$$

First we suppose that a constant threshold model is used, so the second term $J_{Tr} = 0$.

When kernel-based lithography model is used, which has been discussed in Section 2.4, $J_I(i,j)$ in the Jacobian matrix can be calculated as below:

$$
\begin{aligned}
J_I(i,j) &= \frac{I' - I}{\delta d} \\
&= \frac{\sum_k |\Phi_k \otimes (Mask_i + \delta Mask_{ij})|^2 - \sum_k |\Phi_k \otimes Mask_i|^2}{\delta d} \\
&= \frac{\sum_k (2 \times Real((\Phi_k \otimes Mask_i) \times (\Phi_k \otimes \delta Mask_{ij})^*)) + \sum_k |\Phi_k \otimes \delta Mask_{ij}|^2}{\delta d}
\end{aligned}
\tag{7}
$$

let,

$$
\begin{aligned}
f_{ki} &= \Phi_k \otimes Mask_i \\
\delta f_{kij} &= \Phi_k \otimes \delta Mask_{ij}
\end{aligned}
$$

we get,

$$
J_I(i,j) = \frac{\sum_k (2 \times Real(f_{ki} \times \delta f_{kij}^*)) + \sum_k |\delta f_{kij}|^2}{\delta d} \tag{8}
$$

$Mask_i$ is all polygons within the "ambit" window of evaluation point (x_i, y_i). $\delta Mask_{ij}$ is a small perturbation rectangle near edge segment j. Figure 6 shows how the item $J_I(i,j)$ in the Jacobian matrix is calculated. Calculating f_{ki} has a computation complexity of $O(n_e)$ when table-lookup method is used, where n_e is the number of edge segments in the "ambit" window.[8] The value of n_e can be greater than one hundred when doing OPC, because that all edges are broken into small edge segments. $\delta Mask_{ij}$ is a single box as shown in Figure 6, which contains only 4 segments. So δf_{kij} require much less computation time than f_{ki}.

Even doing this, the generation of Jacobian matrix will require plenty of computations. This is why we choose not to describe the problem using explicitly EPE based equations. Calculating a MEEF matrix is even more expensive, since it requires to evaluating intensity profile along offset direction and search the image edge to calculate each item.[2]

When variable threshold resist (VTR) model is used, $J_{Tr} \neq 0$. Each item can be calculated by

$$
J_{tr}(i,j) = \frac{Tr' - Tr}{\delta d} \tag{9}
$$

For simple VTR models[9] which are only related to pattern density and intensity slope, the calculation can be speed up using the similar way described before. Unfortunately, when a complex VTR model is used,[10] the computation time would increase significantly due to the supplementary calculation of the extra aerial image properties such as I_{max}, I_{min} and etc., which cannot be decomposed as linear functions of mask patterns. In this case, J_I can be used as an approximated Jacobian matrix at the cost of losing some accuracy and with more iterations. Nevertheless, sparsity control and RT reuse schemes which will be introduced in next sections should still be applicable.

2.6. Sparsity Control Scheme

As shown in Figure 5, for each evaluation point, only the change of segments in the "ambit" window will affect the intensity value, the Jacobian matrix is a sparse matrix in nature. Figure 7 shows the sparsity of a Jacobian matrix. It is quite important to use a sparse linear equations solving algorithm in each Newton iteration step to let the matrix-OPC algorithm have a feasible speed for real application.

Even more, an interaction radius that is typically smaller than the "ambit" can be used to control the sparsity of Jacobian matrix. Since the sparsity is roughly proportional to the square of the interaction radius, a 50 percent reduction in radius could result in a 75 percent reduction in sparsity. When the interaction radius is set to 0, only the diagonal values in the Jacobian matrix are retained, which is just the same with a conventional model-based OPC. Using this scheme, computations of evaluating Jacobian matrix and solving linear equations can be greatly reduced. The interaction radius is a parameter for trading off between correction speed and correction accuracy.

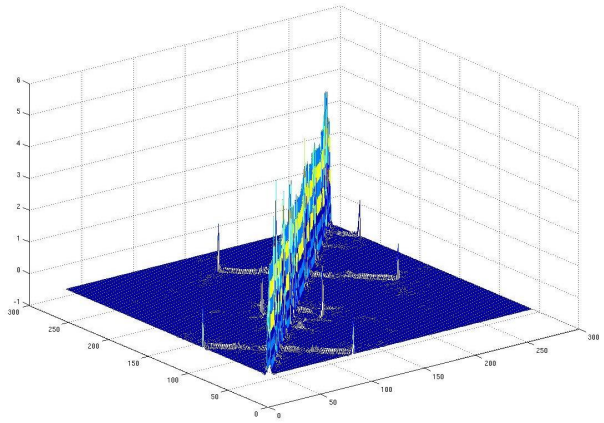


Figure 7. Sparsity of Jacobian Matrix

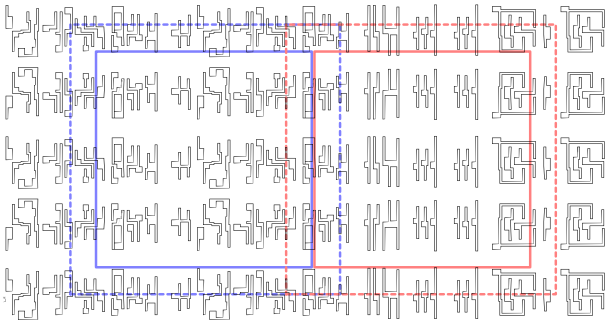


Figure 8. Cutting layout into region templates

2.7. Region Template Reuse Scheme

To control the size of Jacobian matrix, the whole layout is cut into small pieces, and OPC is performed on the pieces one by one. As shown in Figure 8, each piece of the layout, which is called as region template (RT) in this paper, contains two areas, namely the target area and the surrounding context. Both areas are corrected. However, pattern modifications in the surrounding band are not used as the final result. This operation ensures the correction result of target area. The size of the surrounding band is chosen empirically. In our tests 2~3 times of "ambit" value is enough to ensure the correctness of target areas.

When correcting each RT, the information of previously processed RTs nearby can be reused. As illustrated in Figure 9, correction is performed on RTs from left to right and from bottom to top. For each RT not on the lower bottom of the layout, its left and/or bottom neighbors are already been corrected. Two levels of reuse are possible. The first one, like the area 1 with hatch mark in Figure 9, is in the target area of corrected neighbors. Edge segments in this area need not to be corrected when processing current RT, instead they act as the environment of other patterns. This level of reuse reduces the number of variables in Equation 2. The second level of reuse is applicable to area 2 with back hatch mark in Figure 9, they are in the surrounding areas of nearby RTs. Although these pattern modifications are not final, they nevertheless provide good initial values, for speeding up the convergence.

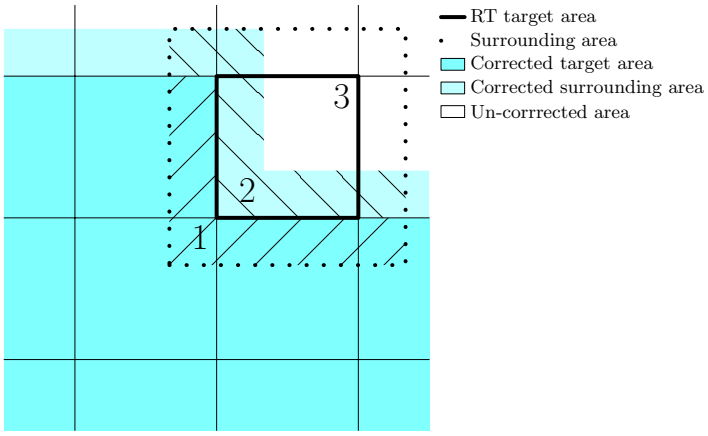


Figure 9. RT reuse scheme

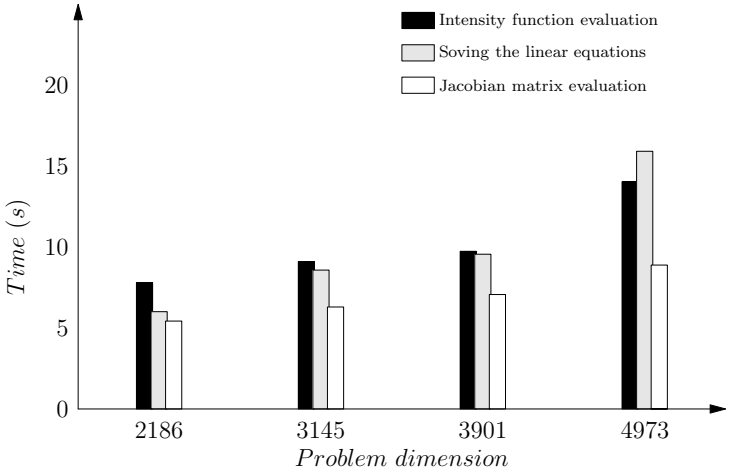


Figure 10. Runtime comparison of three kinds of operations

3. COMPARISON AND EXPERIMENTAL RESULTS

We have implemented this algorithm. Some examples of using this algorithm are shown in this section. All computations were performed on a Dell PowerEdge 1600 workstation (Xeon 1.8GHz × 2 CPU and 2GB Memory). All experiments are performed on a 90nm layout which has an area of 100um × 100um. "Ambit" value of the process model used in these examples is 1000nm.

Figure 10 compares the portion of time spent on three operations, namely evaluating the intensity function, generating Jacobian matrix and solving the sparse linear equations for a single RT. It also shows how the computing time varies with the problem dimension.

Figure 11 and Figure 12 compare the speed and the accuracy for different values of interaction radius, which is performed on a RT having size of $8000nm \times 8000nm$. In this example, OPC accuracy changes little when interaction radius decreases from "ambit" until it reaches about 200nm, where run time is reduced by about 30%. It is likely that an optimal value of interaction radius would depends on lithographic configuration and layout patterns. Thus the user should choose a suitable value for each case.

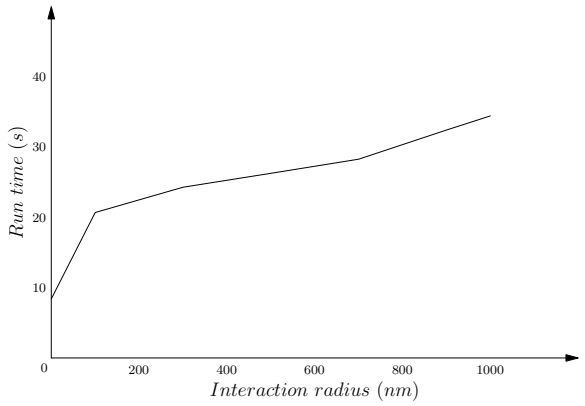


Figure 11. Run time comparison when different interaction radii are used

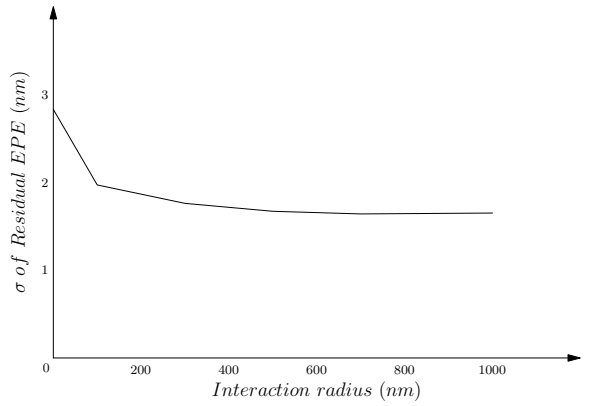


Figure 12. Standard deviation of residual EPE comparison when different interaction radii are used

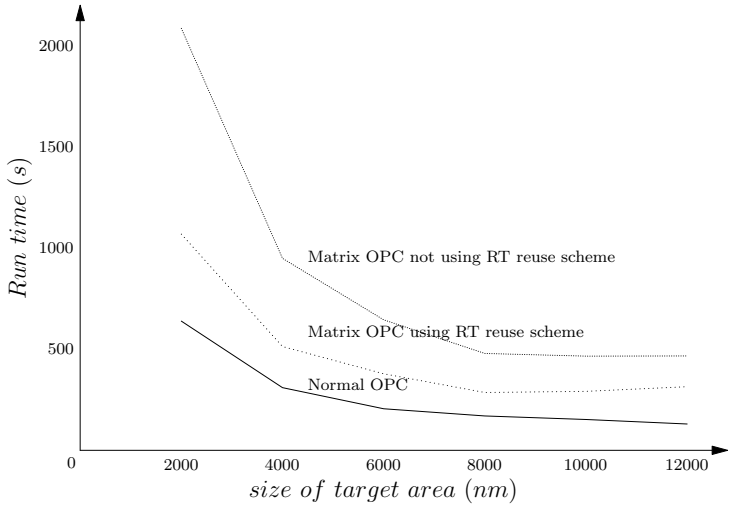


Figure 13. Speed comparison when using different sizes of target area

Figure 13 shows the correction speed versus target area size. Three types of OPC algorithms are compared, namely the normal MB-OPC (8 iterations are performed), the matrix OPC using and not using RT reuse scheme (both with an interaction radius of 200nm). The size of surrounding band is 2000nm. It shows that RT reuse effectively speeds up the matrix OPC by about 90%. The result also shows that the correction speed of using matrix OPC with RT reuse can be close to that of a normal MB-OPC. 8000nm is the optimal size of the target area for this example.

Figure 14 compares the residual EPE distributions of a single RT area (8000nm × 8000nm), one is corrected using the matrix OPC and the other is corrected using a normal MB-OPC, both with 8 iterations. It shows that the new algorithm has better EPE control. Figure 15 shows a layout corrected by this algorithm. Correction results of the 1st, 4th and final iteration are shown and compared. In this example, some segments are too close to each other after OPC. This is due to no constraint is applied on the calculated offsets here. In a real case the minimum space should be kept to be compatible with mask making rules.

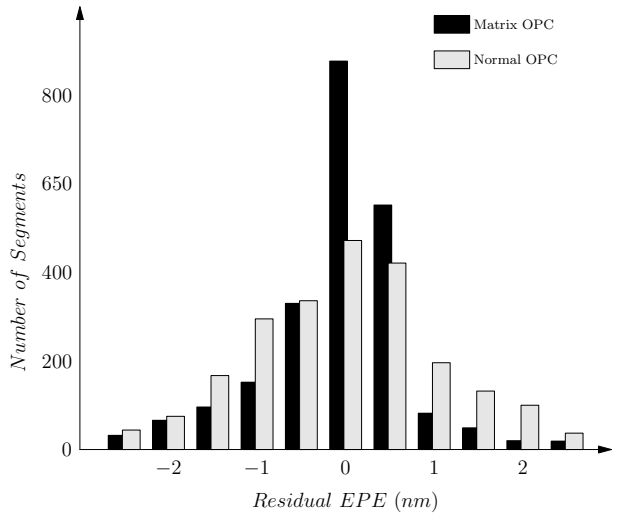


Figure 14. Residual EPE distribution after OPC

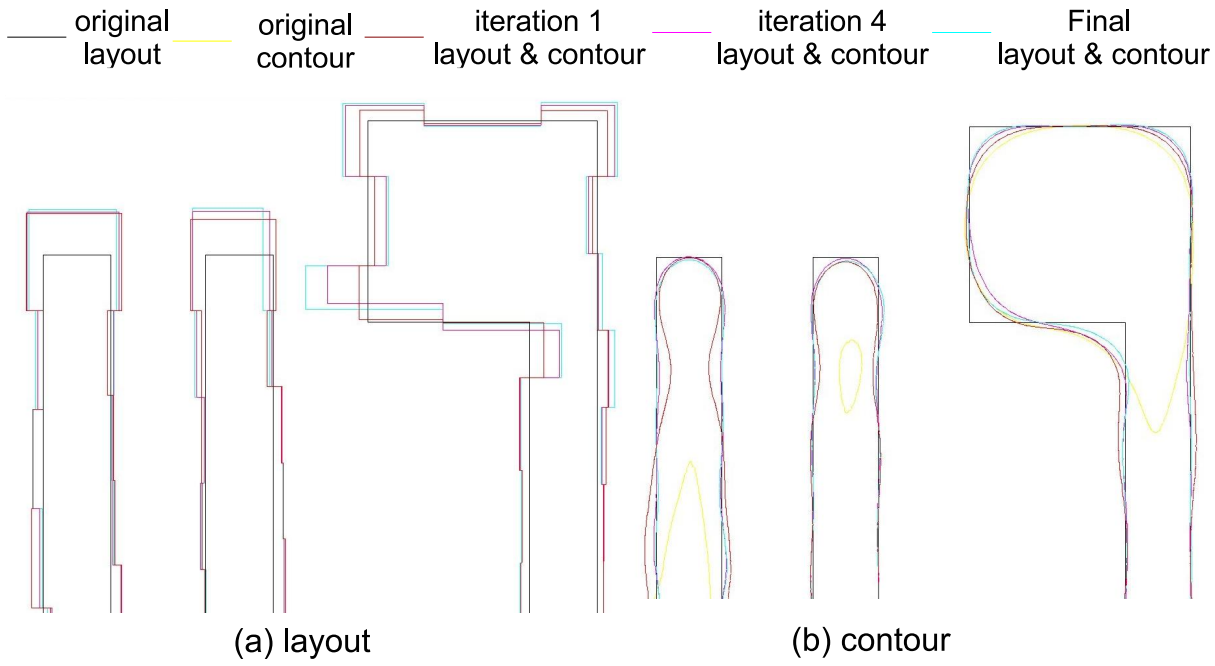


Figure 15. An example using the new algorithm

4. CONCLUSION

In this paper we present a new kind of matrix OPC algorithm based on generalized intensity distribution functions. A scheme of controlling the sparsity of Jacobian matrix using a proper interaction radius to reduce the computation is introduced. A scheme of reusing information from already corrected RTs nearby is also presented. Experiments prove the efficacy of these new methods. Results also show that the new algorithm can

achieve better EPE control while keeping the correction speed close to normal OPC algorithm.

ACKNOWLEDGMENTS

The authors would like to thank Xin Zhou from Anchor Semiconductors Inc. for fruitful discussion.

REFERENCES

1. K. Herold, N. Chen, and I. P. Stobert, "Managing high-accuracy and fast convergence in opc," *Photomask Technology 2006* **6349**(1), p. 634924, SPIE, 2006.
2. S.-H. Choi, A.-Y. Je, J.-S. Hong, M.-H. Yoo, and J.-T. Kong, "Meef-based correction to achieve opc convergence of low-k1 lithography with strong oai," *Optical Microlithography XIX* **6154**(1), p. 61540P, SPIE, 2006.
3. B. Painter, L. L. M. III, and M. L. Rieger, "Classical control theory applied to opc correction segment convergence," *Optical Microlithography XVII* **5377**(1), pp. 1198–1206, SPIE, 2004.
4. N. B. Cobb and Y. Granik, "Model-based opc using the meef matrix," *22nd Annual BACUS Symposium on Photomask Technology* **4889**(1), pp. 1281–1292, SPIE, 2002.
5. C. Dolainsky, W. Maurer, and T. Waas, "Evaluation of resist models for fast optical proximity correction," *17th Annual BACUS Symposium on Photomask Technology and Management* **3236**(1), pp. 202–207, SPIE, 1997.
6. N. B. Cobb, A. Zakhor, M. Reihani, F. Jahansooz, and V. N. Raghavan, "Experimental results on optical proximity correction with variable-threshold resist model," *Optical Microlithography X* **3051**(1), pp. 458–468, SPIE, 1997.
7. A. E. Rosenbluth, G. M. Gallatin, R. L. Gordon, W. Hinsberg, J. Hoffnagle, F. Houle, K. Lai, A. Lvov, M. Sanchez, and N. Seong, "Fast calculation of images for high numerical aperture lithography," *Optical Microlithography XVII* **5377**(1), pp. 615–628, SPIE, 2004.
8. N. B. Cobb, A. Zakhor, and E. A. Miloslavsky, "Mathematical and cad framework for proximity correction," in *Optical Microlithography IX*, *Optical Microlithography IX* **2726**, pp. 208–222, SPIE, 1996.
9. D. Fuard, M. Besacier, and P. Schiavone, "Assessment of different simplified resist models," *Optical Microlithography XV* **4691**(1), pp. 1266–1277, SPIE, 2002.
10. Y. Granik, N. B. Cobb, and T. Do, "Universal process modeling with vtre for opc," *Optical Microlithography XV* **4691**(1), pp. 377–394, SPIE, 2002.