A Novel Intensity Based Optical Proximity Correction Algorithm with Speedup in Lithography Simulation

Peng Yu and David Z. Pan
The Department of Electrical and Computer Engineering
The University of Texas at Austin
Email: yupeng@cerc.utexas.edu and dpan@ece.utexas.edu

***Abstract*—It is important to reduce the Optical Proximity Correction (OPC) runtime while maintaining a good result quality. In this paper, we obtain a better formula, which theoretically speeds up the widely used method, Optimal Coherent Approximations (OCA's), by a factor of $2\times$. We speed up the OPC algorithm further by making it intensity based (IB-OPC), because it requires much less intensity simulations than the conventional Edge Placement Error (EPE) based OPC algorithms. In addition, the IB-OPC algorithm, which uses the efficiently computed sensitivity information, converges faster than the EPE based OPC. Our IB-OPC experimental results show a runtime speedup of up to $15\times$ with a comparable result quality as of the EPE based OPC.**

I. Introduction

Optical Proximity Correction (OPC) algorithm modifies mask shapes to compensate the optical proximity effect due to the subwavelength lithography printing. The OPC algorithms can be classified as polygon based OPC [1] and pixel based inverse lithography [2]. We consider the first category in this paper, since it is the widely adopted method in the industry.

All the existing published polygon based OPC algorithms are based on Edge Placement Error (EPE) [1], which stands for the difference between the printed contour and the target contour. The simple EPE based OPC algorithm [1] has been later improved using Mask Error Enhancement Matrix (MEEM) [3], where MEEM is the EPE sensitivity matrix with respect to changes in mask shapes.

It is well known that OPC is a very time consuming process, which could take days. Slow OPC runtime can affect the turnaround-time (TAT) adversely. Therefore, it is important to reduce OPC runtime while maintaining a good OPC result quality.

The runtime is conventionally improved from two aspects. The first is from fine tuning the OPC algorithms, the OPC algorithm parameters, and the recipes (e.g. segmentation and tagging). A few new convergence schemes were proposed to improve the convergence rate [4]. A neural network was proposed to give a better initial guess which results in lower number of OPC iterations [5]. The runtime implications of some OPC parameters have been shown in [6], which pointed out the potential trade-off between various parameters for runtime. Optimized multi OPC recipes have also been proposed for SoC applications to reduce runtime and improve OPC accuracy [7].

The second is from parallel computation and dedicated hardware. Multiprocessing and multithreading have been used for OPC on linux workstation clusters [8]. A hardware-accelerated computational lithography platform have also been built [9]. IBM has developed software for IC design and DFM software with the IBM's BlueGene supercomputer [10].

However, none of the above approaches have fundamentally improved the lithography modeling nor the OPC algorithm itself. In this paper, we improve the OPC runtime in both of these aspects. We found that the widely used Optimal Coherent Approximations (OCA's) formulation [11] can be simplified by using the symmetric properties of the lithography imaging systems, which gives us a runtime speedup of up to $2\times$ without any loss in accuracy. We also observe that making the intensity on the target contour equal the intensity threshold is the equivalent of making EPE zero. Based on this observation, we propose an intensity based OPC algorithm, which is demonstrated be faster than the EPE based method, because it requires less intensity simulation, which takes the majority of the OPC runtime.

The main contributions of this paper are as follows:

- We derive a formula which could reduce the simulation runtime by $2\times$ without any accuracy loss, by using the symmetric properties in the lithography systems.
- Our intensity based OPC (IB-OPC) algorithm using the sensitivity information converges faster than the conventional EPE based method.
- We provide an efficient method for the intensity sensitivity computation.
- The IB-OPC algorithm achieves a significant runtime speedup, while maintaining the OPC result quality.

The rest of this paper is organized as follows. In Section II, we review the lithography imaging model and show our new simulation formula. In Section III, we review the segmentation of masks into the parametrized representation and describe the fast intensity sensitivity computation. Section IV shows the IB-OPC formulation and the algorithm, a variant of the Newton method, which uses the intensity sensitivity information. Section V shows the experimental results. Section VI concludes this paper.

1-4244-1382-6/07/$25.00 ©2007 IEEE

II. Speedup in Lithography Imaging Simulation

We first review the lithography imaging model and then show an improved simulation formula by using some common lithography system symmetric properties. The speedup is justified for practical cases with errors.

A. Lithography Imaging — Hopkins Equation

The latent image intensity in the photoresist, commonly treated 2-dimensionally by phenomenological models in OPC softwares, is given by the Hopkins equation [12],

$$\mathcal{I}(\boldsymbol{k};z)=\iint \mathcal{T}(\boldsymbol{k}+\boldsymbol{k}',\boldsymbol{k}';z)\mathcal{F}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{F}^*(\boldsymbol{k}')\mathrm{d}^2\boldsymbol{k}'. \quad (1)$$

$\mathcal{F}(\boldsymbol{k})$ is the mask transmission function $F(\boldsymbol{r})$ in the frequency domain, where $\boldsymbol{k}$ denotes a point in the frequency domain and $\boldsymbol{r}$ denotes a point in the spatial domain. $\mathcal{I}(\boldsymbol{k};z)$ is the chemical latent image in the frequency domain at z defocus. $\mathcal{T}(\boldsymbol{k},\boldsymbol{k}';z)$ is the *transmission cross coefficient* (TCC), given by

$$\mathcal{T}(\boldsymbol{k}',\boldsymbol{k}'';z)=\mathcal{G}(\boldsymbol{k}'-\boldsymbol{k}'') \times\iint \mathcal{J}_{\mathrm{O}}^{-}(\boldsymbol{k})\mathcal{K}(\boldsymbol{k}+\boldsymbol{k}';z)\mathcal{K}^*(\boldsymbol{k}+\boldsymbol{k}'';z)\mathrm{d}^2\boldsymbol{k}, \quad (2)$$

The meaning of the symbols are described below:

- $\mathcal{G}(\boldsymbol{k})$ is the diffusion kernel, written as
$$\mathcal{G}(\boldsymbol{k})=e^{-2\pi^2d^2k^2}, \quad (3)$$
which corresponds the diffusion of the latent image during the post-exposure-bake (PEB), where d is the diffusion length and $k=|\boldsymbol{k}|$.
- $\mathcal{J}_{\mathrm{O}}^{-}(\boldsymbol{k})$ is the illumination function. We illustrate some commonly used ones in Figure 1.

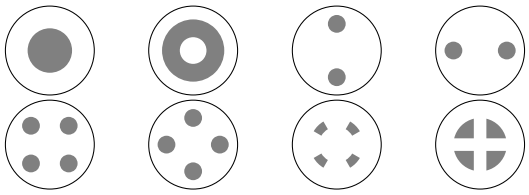

Figure 1. Commonly used illumination schemes. The radii of the outer circles are 1. $\mathcal{J}_O^-$ is a constant over the gray regions.

- $\mathcal{K}(\boldsymbol{k};z)$ is the projection system transfer function. Assuming a circular pupil, $\mathcal{K}(\boldsymbol{k};z)$ can be written as
$$\mathcal{K}(\boldsymbol{k};z)=\begin{cases} e^{i2\pi z\sqrt{1-k^2}} & k<1 \\ 0 & \text{otherwise}\end{cases}. \quad (4)$$
- The superscript * denotes the complex conjugation operation.

B. Improving the Simulation Speed Using Symmetries

It is well known that

$$\mathcal{G}(\boldsymbol{k})\in\mathbb{R},\ \mathcal{J}_{\mathrm{O}}^{-}(\boldsymbol{k})\in\mathbb{R} \text{ and } \mathcal{K}(\boldsymbol{k})\in\mathbb{C}, \quad (5)$$

where $\mathbb{R}$ is the set of all real numbers and $\mathbb{C}$ is the set of all complex numbers. This is confirmed in (3), Figure 1 and (4).

In addition, we have Property 1, which is true not necessarily only to the forms in (3), Figure 1 and (4).

Property 1.

$$\mathcal{G}(\boldsymbol{k})=\mathcal{G}(-\boldsymbol{k}), \quad (6)$$
$$\mathcal{J}_{\mathrm{O}}^{-}(\boldsymbol{k})=\mathcal{J}_{\mathrm{O}}^{-}(-\boldsymbol{k}), \quad (7)$$
$$\mathcal{K}(\boldsymbol{k})=\mathcal{K}(-\boldsymbol{k}). \quad (8)$$

Remark. (6) *is true if the diffusion is rotational invariant.* (7) *is true as long as the symmetrical illumination schemes are used (not necessary limited to the ones in Figure 1.* (8) *is true as long as there are no odd order aberrations.*

With the Property 1, we can prove the following property of TCC.

Property 2.

$$\mathcal{T}(\boldsymbol{k}',\boldsymbol{k}'')=\mathcal{T}(-\boldsymbol{k}',-\boldsymbol{k}''). \quad (9)$$

Using Property 2 and the fact

$$\mathcal{T}(\boldsymbol{k}',\boldsymbol{k}'')=\mathcal{T}^*(\boldsymbol{k}'',\boldsymbol{k}'), \quad (10)$$

we have

$$\mathcal{T}(\boldsymbol{k}',\boldsymbol{k}'')=\mathcal{T}^*(-\boldsymbol{k}'',-\boldsymbol{k}'). \quad (11)$$

Therefore, TCC's real and imaginary parts satisfy

$$\mathcal{T}_{\text{real}}(\boldsymbol{k}',\boldsymbol{k}'')=\mathcal{T}_{\text{real}}(-\boldsymbol{k}'',-\boldsymbol{k}') \quad (12)$$

and

$$\mathcal{T}_{\text{imag}}(\boldsymbol{k}',\boldsymbol{k}'')=-\mathcal{T}_{\text{imag}}(-\boldsymbol{k}'',-\boldsymbol{k}'). \quad (13)$$

By using the fact that the mask transmission function $F(\boldsymbol{r})$ is real for commonly used masks, such as binary mask (BIM) or phase shift mask (PSM) with the phases of 0° and 180°, we can easily obtain the following property of $\mathcal{F}(\boldsymbol{k})$.

Property 3.

$$\mathcal{F}(\boldsymbol{k})=\mathcal{F}^*(-\boldsymbol{k}). \quad (14)$$

By using (13) and Property 3, we can prove that $\mathcal{T}_{\text{imag}}$ does not contribute to the image and we have the following Theorem.

Theorem 1. *The aerial image can be computed by the* Reduced Hopkins Equation*,*

$$\mathcal{I}(\boldsymbol{k})=\iint \mathcal{T}_{real}(\boldsymbol{k}+\boldsymbol{k}',\boldsymbol{k}')\mathcal{F}(\boldsymbol{k}+\boldsymbol{k}')\mathcal{F}^*(\boldsymbol{k}')\mathrm{d}^2\boldsymbol{k}'. \quad (15)$$

It has been derived from (1) that the image can be computed by

$$I(\boldsymbol{r})=\sum_{n=0}^{p'-1}\sigma_n\left|Q_n'\ast\ast F\right|^2, \quad (16)$$

where $**$ is the convolution operator and Q'_n's (called kernels) are complex functions [13]. Using Property 2 and Theorem 1, we can get a similar formula without $|\cdot|$,

$$I(\boldsymbol{r}) = \sum_{n=0}^{p-1} \sigma_n (Q_n ** F)^2, \tag{17}$$

where Q_n's are real functions. Note that (17) can not be derived directly from (1) in general, because $\mathcal{T}$ does not satisfy

$$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}(-\boldsymbol{k}'', -\boldsymbol{k}') \tag{18}$$

in general. The choices of p and p' are determined by the error requirements. With the same error requirements, it can be proved that (17) gives us a $2\times$ speedup when there are no aberrations ($p = p'$ in this case).

C. Considering Errors in Practice

Practically, there are always errors in lithography imaging system, which may make Property 1 not valid. But we can separate $\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'')$ into two parts as

$$\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') = \mathcal{T}_{\text{sym}}(\boldsymbol{k}', \boldsymbol{k}'') + \mathcal{T}_{\text{anti}}(\boldsymbol{k}', \boldsymbol{k}''), \tag{19}$$

where

$$\mathcal{T}_{\text{sym}}(\boldsymbol{k}', \boldsymbol{k}'') = \frac{\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') + \mathcal{T}(-\boldsymbol{k}', -\boldsymbol{k}'')}{2} \tag{20}$$

and

$$\mathcal{T}_{\text{anti}}(\boldsymbol{k}', \boldsymbol{k}'') = \frac{\mathcal{T}(\boldsymbol{k}', \boldsymbol{k}'') - \mathcal{T}(-\boldsymbol{k}', -\boldsymbol{k}'')}{2}. \tag{21}$$

Similar to Theorem 1, we only need the real part of $\mathcal{T}_{\text{sym}}(\boldsymbol{k}', \boldsymbol{k}'')$ and the imaginary part of $\mathcal{T}_{\text{anti}}(\boldsymbol{k}', \boldsymbol{k}'')$ for the computation of $\mathcal{I}(\boldsymbol{k})$.

By saying the errors are small, we mean that approximately the same number of terms (p) is needed to decompose both $\mathcal{T}$ and $\mathcal{T}_{\text{sym,real}}$ and we need q ($q \ll p$) terms to decompose $\mathcal{T}_{\text{anti,imag}}$. Therefore, the runtime speedup is still

$$\frac{p}{\frac{p}{2} + \frac{q}{2}} \underset{q \ll p}{\approx} 2. \tag{22}$$

III. Segmentation and Lookup-Table Method for Intensity Sensitivity Computation

In this section, we first review how the mask shapes are segmented and manipulated by OPC algorithm. We then derive a direct and fast way to compute the intensity sensitivity with respect to shape changes, which will be used to compute $\boldsymbol{D}$ in Section IV.

There are usually only polygons in designs. In this work, we only consider rectilinear polygons, which have only horizontal and vertical edges. Our method can be extended to handle polygons with edges in other directions (e.g. $45°$). The polygon edges are usually broken into smaller parts (Figure 2(a)), called segments, which can be shifted by OPC algorithms (Figure 2(b)).

The image intensity can be computed using the conventional table lookup based method [14, 15]. A naive way of intensity sensitivity computation is to perturb the mask shapes a little and resimulate the intensity. The intensity change is proportional to the intensity sensitivity. However, this method requires two intensity simulations, which is slow. We show below an intelligent way for faster sensitivity simulations.

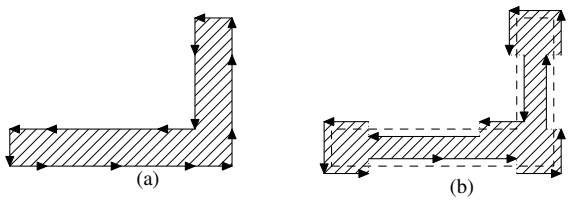

Figure 2. (a) The edges of a rectilinear polygon is segmented. (b) The segments can be shifted by OPC algorithms.

Suppose the i-th segment is in x-direction (the y-direction can be discussed similarly). The sensitivity of the image intensity with respect to the shift of the i-th segment can be derived from (17) as

$$\frac{\partial I}{\partial d_i} = \sum_{n=0}^{p-1} 2\sigma_n (Q_n ** F) \times (Q_n ** \text{boxcar}(x; a_i, b_i)\delta(y - (y_i + d_i))), \tag{23}$$

where $\text{boxcar}(\cdot;\cdot,\cdot)$ is the boxcar function

$$\text{boxcar}(x; a, b) = \begin{cases} 1, & \text{if } a \leqslant x \text{ and } x \leqslant b \\ 0, & \text{otherwise,} \end{cases} \tag{24}$$

$\delta(\cdot)$ is the Dirac delta function, a_i and b_i denote the x coordinates of the two ends of the i-th segment, and d_i denotes the segment shift.

The conventional lookup table method [14, 15] can be used to compute $(Q_n * F)$ since the convolution is a linear operator. By using the same property, we can construct another table to compute $Q_n ** \text{boxcar}(x; a_i, b_i)\delta(y - (y_i + d_i))$. In this case, only two table lookups (for the point (x, a_i) and the point (x, b_i)) are needed to compute the convolution.

We compare the runtime complexity of this new method with the naive method, which computes the intensities twice. Suppose we need to lookup the table for M times to compute the intensity. Using the naive method, we need $2M$ table lookups to get the intensity sensitivity. However, we only need $2 + M$ table lookups to get the sensitivity in the intelligent method. Usually M is much bigger than 2. Therefore, we get a speed up of $2\times$ using the new method.

IV. Intensity Based OPC Algorithm

A. Problem Formulation

The EPE based OPC tries to make EPE zero. However, the computation of EPE is expensive. In Figure 3, to compute the EPE near the tag point A, we need to simulate on the dots to find the printed contour. It may require many simulations before the contour is found. It may also need many simulations to know that the contour can not be found.

Figure 4 shows that it is equivalent to make EPE zero and to make the intensity at the target the same as the intensity threshold. This criterion enables us to simulate intensities at much less number of points (only on the tag point A in the

Figure 3. EPE computation requires many intensity simulations. The box is the target shape. To find the printed contour near the tag point A, we may need to simulate on many points.

example in Figure 3). And we have the follow OPC problem formulation.

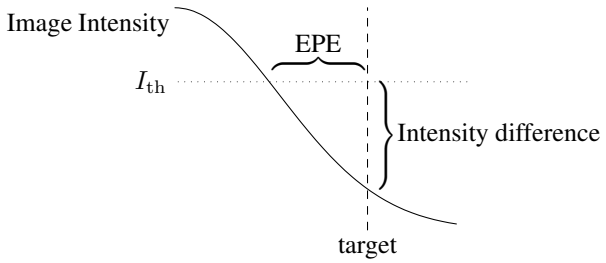

Figure 4. EPE is zero if and only if there is no intensity difference.

Formulation 1 (Intensity Based OPC Fomulation)**.** *Intensity Based OPC Algorithm tries to match the intensity on the target with the intensity threshold.*

Note that this formulation is very general — the intensity threshold is not necessarily a constant. In this paper, we demonstrate the key idea for the constant threshold model.

Suppose there are N segments, which means the mask has freedom of degree N. We need exactly N constraints to uniquely solve the shifts of the N segment vectors. We choose N tag points on the target contour, where the intensity shall match the threshold. There are many tagging strategies. In this work, we tag the central point of each segment.

Let us number the tagging points and the segments from 1 to N. I_i denote the image intensity at the i-th tag point. The threshold at the i-th tag point is denoted as $I_{\text{th}i}$. Therefore, the mathematical formulation of IB-OPC is

$$I_i(d_1, d_2, \ldots, d_N) = I_{\text{th}i} \quad i = 1, 2, \ldots, N. \tag{25}$$

Note that all $I_{\text{th}i}$ equal to a single constant for the constant threshold model.

The system of equations (25) can be written compactly in the vector form as

$$\boldsymbol{A}(\boldsymbol{x}) = \boldsymbol{b} \tag{26}$$

where $\boldsymbol{x}$ denotes the entire vector of values d_i, $\boldsymbol{A}$ denotes the entire vector of function I_i and $\boldsymbol{b}$ denotes the entire vector of values $I_{\text{th}i}$.

B. The Algorithm

In order to solve the nonlinear system of equations (26), we adapt the Newton method discussed in [16] and modify it to fit our particular problem.

Suppose $\boldsymbol{x}$ is close enough to the solution to (26) and $\boldsymbol{A}(\boldsymbol{x})$ can be Taylor expanded in the neighborhood of $\boldsymbol{x}$ as

$$\boldsymbol{A}(\boldsymbol{x} + \delta\boldsymbol{x}) = \boldsymbol{A}(\boldsymbol{x}) + \boldsymbol{J} \cdot \delta\boldsymbol{x} + O(\delta\boldsymbol{x}^2), \tag{27}$$

where $\boldsymbol{J}$ is the *Jacobian* matrix of $\boldsymbol{A}$ at $\boldsymbol{x}$. By neglecting the second and higher order terms in (27) and setting $\boldsymbol{A}(\boldsymbol{x}+\delta\boldsymbol{x}) = \boldsymbol{b}$, we can solve the correction term $\delta\boldsymbol{x}$ from

$$\boldsymbol{J} \cdot \delta\boldsymbol{x} = -\boldsymbol{A}(\boldsymbol{x}) + \boldsymbol{b}. \tag{28}$$

The solution $\delta\boldsymbol{x}$ of (28) shall be applied to $\boldsymbol{x}$ to approximate a better solution to (26). This process shall be repeated until converge. Since we do not need the exact value of $\delta\boldsymbol{x}$ in the iteration, we could approximate $\boldsymbol{J}$ by a diagonal matrix $\boldsymbol{D}$ whose diagonal terms are the same as those of $\boldsymbol{J}$. This approximation is valid since $\boldsymbol{J}$ is usually diagonal dominant in OPC applications. We end up with solving the following system of equations instead

$$\boldsymbol{D} \cdot \delta\boldsymbol{x} = -\boldsymbol{A}(\boldsymbol{x}) + \boldsymbol{b}. \tag{29}$$

Since $\boldsymbol{D}$ is a diagonal matrix, $\delta\boldsymbol{x}$ can be solved easily in (29). And this diagonal approximation also saves us from computing the full Jacobian matrix $\boldsymbol{J}$.

It is intuitive to add the correction $\delta\boldsymbol{x}$ to $\boldsymbol{x}$ as

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_{\text{old}} + \delta\boldsymbol{x}, \tag{30}$$

and to iterating until it converges. However, this method would not converge if the initial guess is not sufficiently close to the solution. In current OPC algorithms, the initial guess is usually chosen to be $\boldsymbol{x} = \boldsymbol{0}$, which could be far away from the solution. Therefore, we need to make our solution scheme converge globally even if the initial guess is far from the solution.

A reasonable strategy is to go to some point $\boldsymbol{x}_{\text{new}}$

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_{\text{old}} + \lambda\boldsymbol{p}, 0 \leqslant \lambda \leqslant 1 \tag{31}$$

along the direction of $\boldsymbol{p} = \delta\boldsymbol{x}$ and to require $f = \frac{1}{2}\boldsymbol{F} \cdot \boldsymbol{F}$ always decrease, where $\boldsymbol{F}(\boldsymbol{x}) \equiv \boldsymbol{A}(\boldsymbol{x}) - \boldsymbol{b}$ and the factor $\frac{1}{2}$ is for later convenience. We will show below how to approximately find λ so that $f(\boldsymbol{x}_{\text{old}} + \lambda\boldsymbol{p})$ decreases.

Let us define

$$g(\lambda) = f(\boldsymbol{x}_{\text{old}} + \lambda\boldsymbol{p}). \tag{32}$$

We want to find an approximate minimum of $g(\lambda)$ ($0 \leqslant \lambda \leqslant 1$). We could approximate $g(\lambda)$ as a parabola, which requires three parameters to be uniquely determined. We could choose $g(0)$, $g(1/2)$ and $g(1)$. But it need the additional evaluation of $\boldsymbol{F}$ for $\lambda = 1/2$. To make the iteration faster, we replace it by $g'(0)$, which does not need an additional evaluation of $\boldsymbol{F}$ as we shall show below.

The derivative of $g(\lambda)$ is

$$g'(\lambda) = \nabla f \cdot \boldsymbol{p} = (\boldsymbol{F} \cdot \boldsymbol{J}) \cdot (-\boldsymbol{D}^{-1} \cdot \boldsymbol{F}). \tag{33}$$

Since we only need a fast estimate of $g'(0)$, we again approximate $\boldsymbol{J}$ as $\boldsymbol{D}$ and get

$$g'(\lambda) \approx -\boldsymbol{F} \cdot \boldsymbol{F}. \tag{34}$$

Therefore, we have

$$g'(0) = -2g(0), \tag{35}$$

which does not require an additional evaluation of $\boldsymbol{F}$.

With $g(0)$, $g'(0)$ and $g(1)$ available, we model $g(\lambda)$ as a parabola:

$$g(\lambda) \approx (g(1) - g(0) - g'(0))\lambda^2 + g'(0)\lambda + g(0) \tag{36}$$

We can easily find that its minimum is taken at

$$\lambda_{\min} = -\frac{g'(0)}{2(g(1) - g(0) - g'(0))} = \frac{g(0)}{g(0) + g(1)}. \tag{37}$$

Since we have made many approximations, we eventually need to check that $g(\lambda_{\min})$ is indeed less than $g(0)$. Otherwise, we do not accept it and the algorithm stops.

We list the IB-OPC algorithm in Algorithm 1 for completeness. This algorithm is fast for four reasons:

1) It employs our new improved lithography simulation formula (17).
2) It is intensity based, which require much less intensity simulation compared with conventional EPE based method.
3) The intelligent sensitivity computation method enables it to compute $\boldsymbol{D}$ fast.
4) Newton method makes the algorithm converge in less number of iterations.

Algorithm 1 IB-OPC algorithm

```
1: function IB-OPC
2:     x_0 ← 0 // initial mask equals target
3:     i ← 0
4:     repeat
5:         p ← D(x_i)^{-1} · (−A(x_i) + b)
6:         Compute g(0) and g(1) using (32)
7:         λ_min ← g(0) / (g(0)+g(1))
8:         if g(λ_min) ⩾ g(0) then break
9:         x_{i+1} ← x_i + λ_min p
10:        i ← i + 1
11:    until x_i = x_{i−1} // stop if nothing changes
```

V. Experimental Results

We implemented two versions of lithography simulators based on the old formula (16) and the new formula (17), respectively. And we implemented our IB-OPC algorithm. For comparison, we also implemented the current EPE based OPC algorithm (see Section 2 in [3]) and we set its parameter $C = 3.33$. All the implementations were in C++.

The following experiments were performed on a $2.8\,\mathrm{GHz}$ Pentinum-4 Linux machine. We used $1\,\mathrm{nm}$ mask grid size (scaled to wafer). We used the conventional partially coherent illumination with $\sigma = 0.7$, the numerical aperture $\mathrm{NA} = 0.8$, the wavelength $\lambda = 193\,\mathrm{nm}$. We used 6 kernels with the interaction radius of $600\,\mathrm{nm}$. The intensity threshold was 0.15. The four test patterns are compatible with the $65\,\mathrm{nm}$ technology poly layer design rules. The comparison of the OPC results and the runtimes are shown in Table I.

"# EPE" denotes the total number of EPEs that we measured (i.e., the number of tag points). "# iter" denotes the number of iterations. "RT slow" denotes the OPC runtime using (16). "RT" denotes the OPC runtime using (17). "RT per iter" denotes how much runtime each iteration takes. "$\mu_{|\mathrm{EPE}|}$" denotes the average of the EPE absolute value (post OPC). "$\sigma_{|\mathrm{EPE}|}$" denotes the standard deviation of the EPE absolute value (post OPC). The runtime improvement can be separated into three parts as

$$\begin{aligned}&\text{Total improvement}\\ &= \text{Improvement due to (17)}\\ &\times \text{Improvement due to the intensity based method}\\ &\times \text{Improvement on the number of iterations.}\end{aligned} \tag{38}$$

The three factors on the right hand side of (38) and the left hand side of (38) are shown in the last four columns in Table I, respectively. "Average" denotes the averages over the four test cases for the appropriate quantities explained above. We tried both $100\,\mathrm{nm}$ and $50\,\mathrm{nm}$ segment lengths.

As we can see from the table. The runtime improvement due to the new lithography simulation formula (17) is about 1.5, which is less than the theoretical speedup of $2\times$. This is because not all the runtime is taken by the arithmetic operations in (16) or (17). The use of (17) only improves the arithmetic operation runtime by $2\times$. Simulating only intensities give us an additional speedup of about 2.8. The reductions in the number of iterations due to the Newton method are $1.44\times$ for $100\,\mathrm{nm}$ segment lengths and $3.28\times$ for $50\,\mathrm{nm}$ segment length. As the segment length goes down, we can see the number of iterations increases much more in the EPE based algorithm than in IB-OPC. The total runtime speedup from IB-OPC can be up to $15.0\times$ for $50\,\mathrm{nm}$ segment length, which is a significant improvement. As technology scales down, we would expect the segment length be even smaller, in which case IB-OPC could give even more runtime improvement compared with the EPE-based OPC.

In addition, we clearly see that IB-OPC results are comparable with the EPE based OPC results. In fact, IB-OPC reduces average EPE error for both segment lengths, and reduces EPE standard deviation for $100\,\mathrm{nm}$ segment length, although it increases the standard deviation of $|\mathrm{EPE}|$ a little for $50\,\mathrm{nm}$ segment length.

VI. Conclusions

We have derived a new formula for the lithography simulation, which gives $2\times$ runtime speedup over the OCA's. The derivations are based on the symmetric properties of the lithography imaging systems. Conventional OPC algorithms are EPE based, which require many image intensity simulations. To reduce runtime, we have proposed the first intensity based OPC algorithm, which requires much less intensity simulations. The algorithm is further speeded up by a variant of the Newton method for fast convergence, which

Table I
OPC RESULTS AND RUNTIMES COMPARISON

Case #	# EPE	EPE based OPC						IB-OPC					RT Impr sym (×)	RT Impr per iter (×)	Iter Impr (×)	Total RT Impr (×)
		# iter	RT slow(s)	RT(s)	RT per iter (s)	$\mu_{\|EPE\|}$ (nm)	$\sigma_{\|EPE\|}$ (nm)	# iter	RT(s)	RT per iter (s)	$\mu_{\|EPE\|}$ (nm)	$\sigma_{\|EPE\|}$ (nm)				
100 nm segment length																
1	77	16	0.72	0.47	0.029	1.81	0.96	12	0.14	0.013	0.91	0.68	1.53	2.3	1.33	5.2
2	105	16	1.24	0.83	0.052	1.83	0.95	11	0.19	0.016	0.98	0.76	1.49	3.3	1.45	6.5
3	88	16	0.81	0.54	0.034	1.78	0.91	12	0.14	0.013	0.77	0.64	1.50	2.7	1.33	5.8
4	119	18	1.54	1.03	0.057	1.71	1.00	11	0.23	0.019	1.05	0.82	1.50	3.0	1.64	6.7
Average						1.78	0.96				0.93	0.72	1.50	2.8	1.44	6.0
50 nm segment length																
1	153	51	8.32	5.27	0.103	1.66	1.00	16	0.62	0.039	1.33	1.11	1.58	2.7	3.19	13.4
2	209	49	13.90	8.60	0.175	1.69	1.04	15	0.97	0.065	1.53	1.32	1.62	2.7	3.27	14.3
3	172	48	8.66	5.71	0.119	1.67	1.00	16	0.72	0.045	1.24	1.09	1.52	2.6	3.00	12.0
4	235	51	15.31	9.90	0.194	1.58	1.02	14	1.02	0.072	1.43	1.35	1.55	2.7	3.64	15.0
Average						1.65	1.02				1.38	1.22	1.57	2.7	3.28	13.7

computes the intensity sensitivity using an intelligent method. Our experiments have shown much improved runtimes and a good OPC result quality. We only considered constant threshold model in this work, but we will extend the algorithm to handle variable threshold model in the future.

ACKNOWLEDGMENTS

This work is partially supported by SRC, IBM Faculty Award, Fujitsu, Qualcomm, Sun, and Intel equipment and KLA-Tencor software donations. The authors would like to thank Dr. Chris A. Mack from lithoguru.com and Dr. Wang Yao from UT Austin Physics Department for helpful discussions.

REFERENCES

[1] N. B. Cobb, "Fast Optical and Process Proximity Correction Algorithms for Integrated Circuit Manufacturing," Ph.D. dissertation, University of California at Berkeley, 1998.

[2] A. Poonawala and P. Milanfar, "Mask Design For Optical Microlithography — An Inverse Imaging Problem," *IEEE Trans. on Image Processing*, vol. 16, no. 3, pp. 774–788, Mar. 2007.

[3] N. B. Cobb and Y. Granik, "Model-based OPC using the MEEF matrix," in *Proc. SPIE 4889*, Dec. 2002, pp. 1281–1292.

[4] B. Painter, L. L. Melvin, III, and M. L. Rieger, "Classical control theory applied to OPC correction segment convergence," in *Proc. SPIE 5377*, May 2004, pp. 1198–1206.

[5] W. C. Huang, C. M. Lai, B. Luo, C. K. Tsai, M. H. Chih, C. W. Lai, C. C. Kuo, R. G. Liu, and H. T. Lin, "Intelligent model-based OPC," in *Proc. SPIE 6154*, Apr. 2006, pp. 1065–1073.

[6] A. D. Dave, C. P. Babcock, S. N. McGowan, and Y. Zou, "Methods and factors to optimize OPC run-time," in *Proc. SPIE 6520*, 2007.

[7] M. Do, J. Kang, J. Choi, J. Lee, Y. Lee, and K. Kim, "Efficient approach to improving pattern fidelity with multi-OPC model and recipe," in *Proc. SPIE 6349*, Oct. 2006.

[8] N. Cobb and Y. Granik, "New concepts in OPC," in *Proc. SPIE 5377*, 2004, pp. 680–690.

[9] J. Ye, Y.-W. Lu, Y. Cao, L. Chen, and X. Chen, "System and method for lithography simulation," Patent US 7,117,478 B2, Jan. 18, 2005.

[10] G. A. Gomba, "Collaborative Innovation: IBM's Immersion Lithography Strategy for 65 nm and 45 nm Half-pitch Nodes & Beyond," in *Proc. SPIE 6521*, 2007.

[11] Y. C. Pati and T. Kailath, "Phase-shifting masks for microlithography: automated design and mask requirements," *Journal of the Optical Society of America A*, vol. 11, pp. 2438–2452, Sep. 1994.

[12] M. Born and E. Wolf, *Principles of Optics : Electromagnetic Theory of Propagation, Interference and Diffraction of Light*, 7th ed.

[13] Y. Pati, A. Ghazanfarian, and R. Pease, "Exploiting structure in fast aerial image computation forintegrated circuit patterns," *IEEE Trans. on Semiconductor Manufacturing*, vol. 10, no. 1, pp. 62–74, Feb. 1997.

[14] P. Yu, S. X. Shi, and D. Z. Pan, "Process variation aware opc with variational lithography modeling," in *Proc. Design Automation Conf.*, 2006, pp. 785–790.

[15] ——, "True Process Variation Aware Optical Proximity Correction with Variational Lithography Modeling and Model Calibration," *Journal of Microlithography, Microfabrication and Microsystems*, vol. 6, no. 3, Jul.–Sep. 2007.

[16] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992, pp. 383–385.