A Robust Approach for Process Variation Aware Mask Optimization

Jian Kuang, Wing-Kai Chow and Evangeline F.Y. Young
Department of Computer Science and Engineering
The Chinese University of Hong Kong, Shatin, NT, Hong Kong
{jkuang, wkchow, fyyoung}@cse.cuhk.edu.hk

***Abstract*—As the minimum feature size continues to shrink, whereas the wavelength of light used for lithography remains constant, Resolution Enhancement Techniques are widely used to optimize mask, so as to improve the subwavelength printability. Besides correcting for error between the printed image and target shape, a mask optimization method also needs to consider process variation. In this paper, a robust mask optimization approach is proposed to optimize the process window as well as the Edge Placement Error (EPE) of the printed image. Experiments results on the public benchmarks are encouraging.**

I. Introduction

Optical proximity correction (OPC) is a major resolution enhancement technique, in which the edges of the features are moved to compensate for distortions. Conventional OPC assumes nominal process parameters without considering process variations due to the problem difficulty and the long running time of lithography simulations across process windows. However, process window awareness is important and ignoring OPC impacts of process variations can cause erroneous timing, power and yield. It is thus important to consider process variations during OPC. Some significant sources of variation include dose and focus. In the recent ICCAD contest on this problem [2], the process window is measured by running lithography simulations at different dose and focus corners on the mask and taking the XOR of all the contours.

In this paper, we work on this OPC problem with process window awareness. There were different mask optimization methods, including pixel based approaches like [6], as well as edge based approaches like [1]. A mask optimization process mainly has two steps: (i) simulation to predict the image of the mask, and (ii) optimization of the mask to correct for error between the target and simulated image. Here, not only the error between the simulated image and the target shape, we also aim at minimizing the difference between the images when produced under different process parameters. Our work is based on benchmark layouts from advanced technology nodes and lithographic models derived from actual technology parameters to simulate the image of a mask on a wafer provided by ICCAD Contest [8]. We also evaluate our Process Variation driven OPC methodology using the standard metrics proposed in the contest. The major contributions of this work are: (i) We propose an integrated flow to co-optimize process window and printability of masks. (ii) The proposed flow consists of 3 well-designed stages that systematically optimize a mask effectively in a short running time, and our results outperform the most updated work.

The rest of this paper will be organized as follows. In section II, we will formulate the problem and discuss preliminaries on this problem. Our algorithm will be presented in section III. The experimental results will be presented in section IV followed by a conclusion at the end.

II. Problem Formulation

In this problem, we are given a layout with $N \times N$ pixels, and the output is a mask solution with $N \times N$ pixels such that the wafer image of the mask has the best quality. The primary objective is to minimize the number of Edge Placement Error (EPE) violations because it means the actual fidelity of the image, and the minimization of the Process Variation Band (PVBand) also needs to be considered as this represents the sensitivity of the mask to process variation. The image is produced with a lithography simulator that uses industrial optical models and resist model for the M1 layers.

This work was supported in part by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China, under Project CUHK14209214.

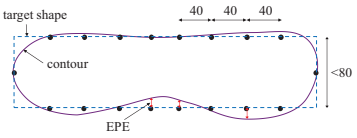

Fig. 1. Illustration for EPE Measurement.

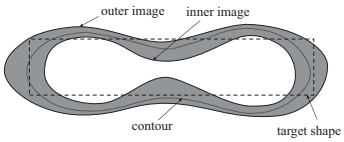

Fig. 2. Illustration for PVBand (the Dark Part).

A. Edge Placement Error (EPE)

EPE is the geometric displacement of the image contour from the target edge of the layout polygon. In our implementation, the limit of such displacement is set as 15nm (=15 pixels). For effective computation, we need to designate a set of points at which the EPE violation will be measured. First of all, this EPE check will not be done for short corners because in lithography, corner images are allowed to be round in shape as long as no electrical violations are caused and thus some amount of EPE will be incurred at corners anyway. Following the convention in [8], EPE checks will be performed at 40nm intervals and the points of EPE checks will be generated as follows: (1) Every polygon will be processed edge by edge. (2) EPE is measured perpendicular to the edge. (3) If the length of an edge does not exceed 80nm (80 pixels), EPE will be checked at the center of the edge only. (4) If the length of an edge exceeds 80nm, the first point from both ends will be located at 40nm from the endpoints. Other points will be generated at 40nm intervals until the points from the two endpoints cross each other. An example is shown in Fig. 1.

B. Process Variation Band (PVBand)

The process window is measured as the area in nm^2 of the XOR region between the image contour obtained in two extreme conditions. Two simulations were used, one at nominal focus and +2% dose and the other one at defocus and -2% dose. The defocus condition is simulated with the defocus kernels provided by [8]. After simulating under these two conditions, we will perform an XOR operation between the two image contours, generating the PVBand. An example to illustrate the meaning of PVBand is shown in Fig. 2. One important goal of this work is to minimize this PVBand area.

C. Simulation Model

Two models are needed to generate a wafer image from a given mask – an optical model and a resist model. The optical model produces an aerial image that is a distribution of light intensity at the wafer plane. Theoretically, it is produced by convolving the mask (M) with a set of optical kernels (h) that represent the singular

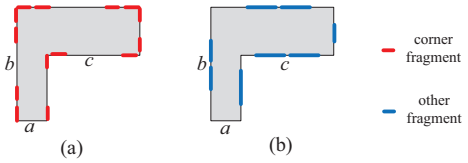

Fig. 3. Illustration for the Fragmentation Process.

value decomposition of the optical system. In practice, this is done by taking Fourier transform of the mask, followed by a weighted multiplication with the kernels in the frequency domnain:

$I = F(M) = \sum_{i=1}^{K} \rho_k (h_k \otimes M)(h_k \otimes M)*$

where the * operator represents complex conjugate and K is the total number of kernels. Then, the resist model is applied to the aerial image. This models how the photo sensitive resist materials respond to the aerial image intensity to produce patterns on the wafer. A simple constant threshold resist model is used so that any intensity above the threshold (I_{th}) causes the photoresist to be removed while any intensity level below results in photoresist remaining:

$$\begin{aligned} R(x,y) &= 1 \quad \text{if } I(x,y) \geq I_{th} \\ &= 0 \quad \text{otherwise} \end{aligned}$$

Note that there are two important parameters for a simulation model: dose and focus. With variations on these two parameters, we can simulate images to reflect process variation in practice. PVBand can be calculated through different simulation conditions.

III. The Proposed Algorithm

A. Overview

With the input layout polygons, we first segment the edges of the polygons into fragments (section III-B). Based on the segmentation, we will insert assist features to improve the printability (section III-C). Then, there are three stages: stage 1 will improve the intensity difference (section III-D), stage 2 will improve the EPE (section III-E) and the last stage will optimize the PVBand area (section III-F).

B. Fragmentation

Fragmentation is a step to break the edges of the input polygons into smaller ones such that we can have more freedom to move the edges to optimize printability. Fragmentation has a significant impact on the performance. If too many fragments were used, the algorithm will be very slow. If we have too few fragments movable independently, the solution space will be too limited to obtain a high-quality mask. Therefore a good segmentation method should have a good trade-off between running time and robustness.

Our fragmentation method is an adaptive one from [3]. Instead of segmenting edges into smaller ones with the same length, we first produce fragments near corners (called *corner fragment*) which should be shorter because finer grain control is needed around corners. We determine the number of corner fragments n_c on an edge according to the edge length:

$$\begin{aligned} n_c &= 0 \quad \text{if } l < l_1 \\ &= 2 \quad \text{if } l1 \leq l < l_2 \\ &= 4 \quad \text{otherwise} \end{aligned}$$

where l_1 and l_2 are two parameters. For example, in Fig. 3(a), the length of edge a is smaller than l_1, so there are no corner fragments on it. The length of edge b is larger than l_2, so there are 4 corner fragments on it. Note that the lengths of the corner segments on the same edge are the same.

Then, for each edge, we will segment the remaining part. According to the length of the remaining part, we will divide it into a number of fragments evenly (Fig. 3(b)), such that the length of each fragment is within a predefined range (l_{low}, l_{high}).

C. Assist Feature Insertion

Sub-resolution assist features (SRAFs) are commonly employed in industry, not only to reduce EPE, but also to improve process window. Particularly, assist features are essential for isolated features (e.g., feature a in Fig. 4).

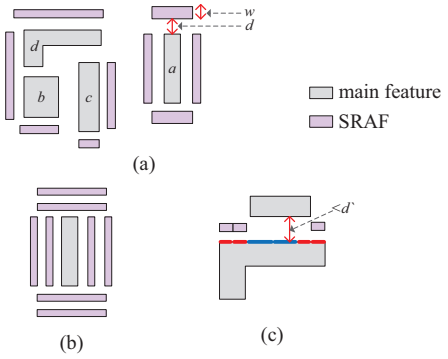

Fig. 4. Illustration for Assist Feature Insertion.

Our assist feature insertion is rule-based. We assume one level (Fig. 4(a)) instead of two levels (Fig. 4(b)) of assist features. The first rule is about the most important parameters: (1) the width w of the assist features; (2) the distance d between an assist feature and a main feature. If w is too large, unwanted image may be resulted from the assist features; if w is too small, the effect will not be enough. Similarly, a good value for d should be chosen in such a way that it is small enough to improve printability and to reduce the process window effectively, but is not too small to extend the main features in an undesirable way.

To find a good set of parameters, we first decide the maximum allowable value for w and the minimum allowable value for d to avoid unwanted images. Then, we simulate the image with different feasible w and d under normal condition, and select the one with the best printability. PVBand is left for later consideration, because (i) otherwise, the problem will be too difficult and too many simulations will be needed and (ii) the main features are not optimized yet.

The second rule is about the main features that need assist features. We only add assist features to a main feature x when there is no other features within a certain distance d' from x. To check this, for each fragment (obtained from our segmentation process), we will compute the distance of it from the nearest feature, and a corresponding assist feature with the pre-defined w and d will be added when this distance is larger than d'. An example can be found in Fig. 4(c).

D. Intensity Difference Optimization (Stage 1)

The input mask can be a very "bad" one in such a sense that the printed image is far from the desired image, and many EPE violations exist. To reduce violations fast at the beginning, we will use intensity difference to guide the fragment movement. For each segment obtained from the fragmentation process, we will take its middle point as the *control point*. We found from experiments that this simple way of choosing the control point is sufficiently effective. The EPE of a segment is 0 if and only if the intensity at the control point of the segment I_i matches the given intensity threshold I_{th} [7]. So we want to make

$F(M + \boldsymbol{D}) = \boldsymbol{I_{th}}$.

where $\boldsymbol{D}$ is the vector of moving distances of each segment. That means we want to minimize

$cost(\boldsymbol{D}) = (F(M + \boldsymbol{D}) - \boldsymbol{I_{th}})^2 = \sum_i (I_{th} - I_i)^2$

This problem is solving a difficult non-linear function. Our strategy is to solve it iteratively, and in each iteration add a small correction $\delta\boldsymbol{D}$ to $\boldsymbol{D}$ such that

$\boldsymbol{D'} = \delta\boldsymbol{D} + \boldsymbol{D}$

Numerical methods, e.g. [7] can be used to approximately get the best direction of $\delta\boldsymbol{D}$, denoted as $\boldsymbol{E}$, to make the fastest decrease of $cost(\boldsymbol{D})$.

Our approach is shown in Algorithm 1, which is an iterative method. The whole process stops when the iteration number is too large. In each iteration, we will first run the simulator to generate the

Algorithm 1 Intensity Difference Optimization

```
iterNum = 0
while true do
    if iterNum > iterThreshold then
        break
    end if
    Simulate the whole mask to get intensity at each pixel
    for each segment s do
        Calculate EPE of s
        if EPE > 15nm then
            violationNum++
            Add s into violateSegSet
        end if
    end for
    if violationNum ≤ violationThreshold then
        break
    end if
    Calculate E for the segments in violateSegSet
    δD = (1 − 1/(1 + e^{α·(iterNum/iterThreshold−1)})) · E
    for each fragment i in violateSegSet do
        if i is a corner fragment then
            move i by β · δD[i]
        else
            move i by δD[i]
        end if
    end for
    iterNum++
end while
```

image. Unlike in [7] that has to run the algorithm until convergence, we will compute the EPE for each segment after simulation, and if the number of violations in EPE is less than or equal to a predefined parameter *violationThreshold*, the process will stop. The fine grain EPE optimization will be deferred to the next stage.

If the number of violations is larger than the threshold, $\boldsymbol{E}$ will be first calculated, and the fragments that violate EPE will be moved according to $\boldsymbol{E}$. There are two parameters in the algorithm: α and β. α will control the moving distance over iterations, such that the scalar on $\boldsymbol{E}$ will decrease smoothly. Smaller α will make the changing curve of the scalar steeper. β is a scalar for the corner fragments, because they are more sensitive. In our current implementation, α is set as -3 and β is 0.8.

Some previous works like [4] have shown that performing mask optimization at defocus condition, instead of at nominal focus condition, can improve the process window of the mask. So, in the first 2 iterations of this intensity difference optimization process, we will simulate the image (Section II-C) with defocus kernels. Experimental results also confirm the improvement in PVBand.

E. EPE Minimization (Stage 2)

After Stage 1, there will usually be just a small number of EPE violations left. In the second stage, we will further move slightly the edges that violate EPE hopefully to fix all the violations. We first identify the EPE testing points (Fig. 1). As these points are regularly set on each edge, they may not overlap with our control points (centers of the segments produced by our fragmentation process). Therefore, for each testing point, we need to identify its corresponding *impact segments*, that are the segments whose movement will significantly affect the EPE and intensity at that testing point. In most cases, the impact segments of a testing point will be the segment that contains it and the segments nearby.

The optimization process is similar to that in Stage 1. The major difference is that each segment will only move 2 pixels at a time, as EPE is very sensitive to segment movement at this stage. Note that as the relationship between the mask and the image is very complicated, an EPE driven segment movement may not always improve EPE. In order to avoid local optima, we will allow the EPE violation number to be worse, as long as the increase in the number of violations is not larger than a small number k.

The limit on EPE will be tightened gradually from 15nm, until it reaches 10nm or it cannot be decreased further because the caused violations can not be fixed. The reason for this tightening process is that smaller EPE means more margin to optimize PVBand, and the benefit can be seen in the next stage.

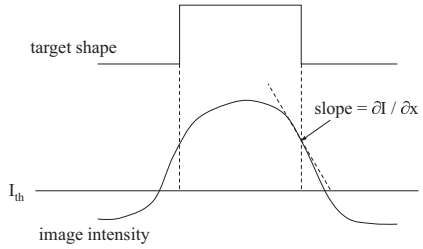

Fig. 5. Illustration for Image Slope.

F. PVBand Optimization (Stage 3)

After stage 1 and 2, either all the EPE violations are fixed, or EPE cannot be further improved. In stage 3, we will try to optimize PVBand of the mask.

First of all, as the process variation was not considered when SRAFs were inserted, we will slightly perturb w and d of the assist features near the segments that have large PVBand so as to improve the process window.

Then, similar to section III-D, we optimize the intensity difference between the two simulations under extreme conditions. There are two ways to optimize: (i) First target at the outer image (Fig. 2), and optimize the intensity of inner image towards it, i.e. to minimize $(F_{focus,+2\%dose}(M+\boldsymbol{D}) - F_{defocus,-2\%dose}(M))^2$. And then, similarly, optimize outer image towards inner image, and iterate between the two. (ii) Optimize both inner and outer images toward the target, i.e. to minimize $(F_{focus,+2\%dose}(M+\boldsymbol{D})-\boldsymbol{I_{th}})^2+(F_{defocus,-2\%dose}(M+\boldsymbol{D})-\boldsymbol{I_{th}})^2$. We will try both of the two ways and keep the best result.

The next optimization method is to optimize the slope of the image. Image slope is calculated as the intensity gradient in a direction perpendicular to the target image edge (Fig. 5). It has been reported in many works, e.g. [1], that sensitivity of an image to process variation in dose can be reduced if the image slope is increased. The reduction is effective at both nominal focus and defocus. Therefore, slope improvement can be an effective way to improve PVBand. This can also be implemented easily because the intensity slope can be computed after we get the intensity at each pixel with simulation. Instead of using a look-up table as in [1], which can be inaccurate because sensitivity of a slope to segment movement at different position can be very different, we will move each segment separately according to the result in the last iteration. For each segment, we will move in the same (opposite) direction of the last iteration if the corresponding slop at the control point is increased (decreased). However, if the movements of a segment turn out to be not improving the slop in two consecutive iterations (that means, we must have tried both directions of move), we will not move that segment further.

Another way to improve PVBand is to have the segments more regularly placed. Examples of regular and irregular segments can be seen from Fig. 6. In the experiments, we observed that a regular edge of a mask tends to have smaller PVBand area than an irregular one.

In this stage, we only move the relatively "safe" fragments with EPE smaller than 5nm, because their movement is unlikely to violate EPE. Here it can be seen that the more safe fragments EPE minimization generates, the larger solution space we can explore to optimize PVBand. But if the EPE is over minimized, it may produce a very bad PVBand. So, in the last stage, a good balance is needed and that is why the smallest limit on EPE is set as 10nm. However, after each iteration of PVBand Optimization, a small number of EPE violations may still be caused. The approach in Section III-E will be used to fix them. If the violations cannot be fixed within a certain number of iterations, this stage will stop. If they can be fixed, we will then run the simulator to get the current PVBand. Note that

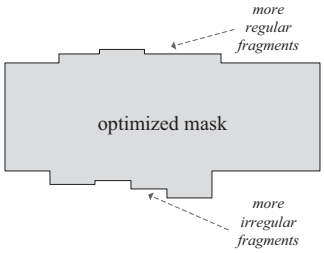

Fig. 6. Illustration for Regular and Irregular Segments.

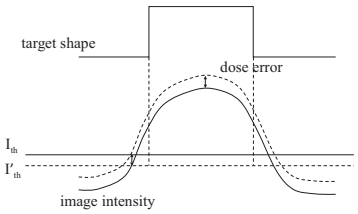

Fig. 7. Illustration for Intensity with Dose Variation.

after EPE fixing, it is possible that PVBand is worsen instead of being improved, comparing with the previous iteration. In this case, stage 3 will also stop.

Notice that in this stage, we need to run the simulator three times for each iteration: one at nominal focus with +2% dose, one at nominal focus with nominal dose, and the last one at defocus with -2% dose. In order to save running time, we try to "estimate" the image at nominal focus with +2% dose from the one obtained at nominal focus and nominal dose. We can do so because variation in dose is similar to changing the value of I_{th} in the resist model. An example is shown in Fig. 7, which illustrates that the dose error can be seen as varying the I_{th}. In our resist model, I_{th} is 0.225. By experiments, we found that simulation at nominal focus with +2% dose is very similar to simulation at nominal focus with nomial dose using $I_{th} = 0.216$. The difference between the numbers of pixels appearing in the images of the two cases is smaller than 0.3% for all the data sets and the two produced images are also almost the same. Therefore, instead of simulating with dose variation, we can approximate the image by using $I_{th} = 0.216$ on the intensity values simulated at nominal condition, which saves us one simulation in each iteration.

IV. Experimental Results

We implement the proposed optimization flow in C language, on a 3.4 GHz Linux machine with 32 GB Memory. To verify the effectiveness of our approach, we test it on the public benchmarks released in ICCAD 2013 CAD Contest, and compare our result with the most updated work [5]. We compare the results in Table I, where #EPEV represents the number of EPE violations calculated by the official checker, and Time is the wall clock time in seconds. It can be seen from the table that we can fix all violations for 7 out of 10 datasets, whereas [5] can only fix for 2 datasets. We can obtain much smaller violation numbers than [5], although our PVBand is larger. After checking the produced images, one possible reason we found is that [5] tends to produce smaller images than us. Naturally, this will result in smaller PVBand, but it also means much worse quality in terms of EPE, which has the highest priority in practice. For runtime, the machines in our experiments and in [5] have similar configurations, but we can obtain nearly 9x speedup on average.

An example of the simulation result is shown in Fig. 8, in which (a) is the target shape, (b) is the image we produced, (c) is taken from one of winners in the ICCAD contest[1], which is smaller than ours and the hotspots are marked in the figure. It is clear that the image generated by our approach is much better in terms of image fidelity, although we may have larger PVBand.

[1]As the corresponding image of [5] is not available to public.

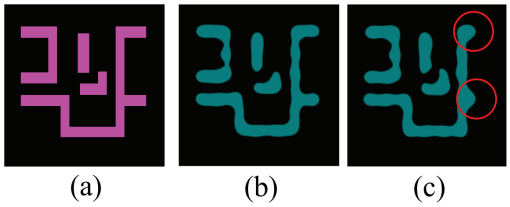

Fig. 8. Comparison of Simulated Images.

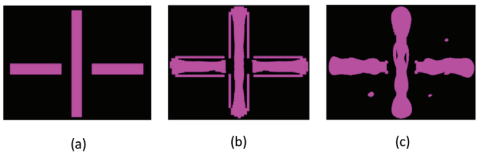

Fig. 9. Comparison of Produced Masks.

Another advantage of our method over [5] is the manufacturability of the output masks. As shown in [5], many isolated small parts of the masks and irregular masks that are very hard to be manufactured will be generated due to the nature of inverse lithography technique, whereas the masks output by our method never have this problem. An example is shown in Fig. 9, where (a) is the target, (b) is our mask, (c) is the mask from [5].

TABLE I. Result Comparisons

	[5]			Ours		
data	#EPEV	PVBand	Time	#EPEV	PVBand	Time
B1	9	56890	1707	0	66218	278
B2	4	48312	1245	0	53434	142
B3	52	84608	2523	18	146776	152
B4	3	24723	1269	0	33266	307
B5	2	56299	2167	1	65631	189
B6	1	49285	2084	0	62068	353
B7	0	46280	1641	0	51069	219
B8	2	22342	663	0	25898	99
B9	3	62529	3022	1	75387	119
B10	0	18141	712	0	18536	61

V. Conclusions

In this paper, we study the mask optimization problem to co-optimize image EPE and process window. Experiments verify our effectiveness. In the near future, we will try to extend our work to consider other process variations like overlay variation to further improve the process window in practice.

References

[1] S. Banerjee, K. B. Agarwa and M. Orshansky, *SMATO: Simultaneous Mask and Target Optimization for Improving Lithographic Process Window*, Proc. of ICCAD, 2010.

[2] S. Banerjee, Z. Li, S. R. Nassif and D. Jamsek, *ICCAD-2013 CAD Contest in Mask Optimization and Benchmark Suit*, Proc. of DAC, 2013.

[3] N. B. Cobb, *Fast Optical and Process Proximity Correction Algorithms for Integrated Circuit Manufacturing*, PhD Thesis, University of California at Berkeley, 1998.

[4] N. B. Cobb and Y. Granik, *Using OPC to optimize for image slope and improve process window*, Proc. of SPIE, 2003.

[5] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan, *MOSAIC: Mask Optimizing Solution With Process Window Aware Inverse Correction*, Proc. of DAC, 2014.

[6] F. Liu and X. Shi, *An Efficient Mask Optimization Method based on Homotopy Continuation Technique*, Proc. of DATE, 2011.

[7] P. Yu and D. Z. Pan, *A Novel Intensity Based Optical Proximity Correction Algorithm with Speedup in Lithography Simulation*, Proc. of ICCAD, 2007.

[8] ICCAD 2013 CAD Contest: http://cad_contest.cs.nctu.edu.tw/CAD-contest-at-ICCAD2013/problem_c.