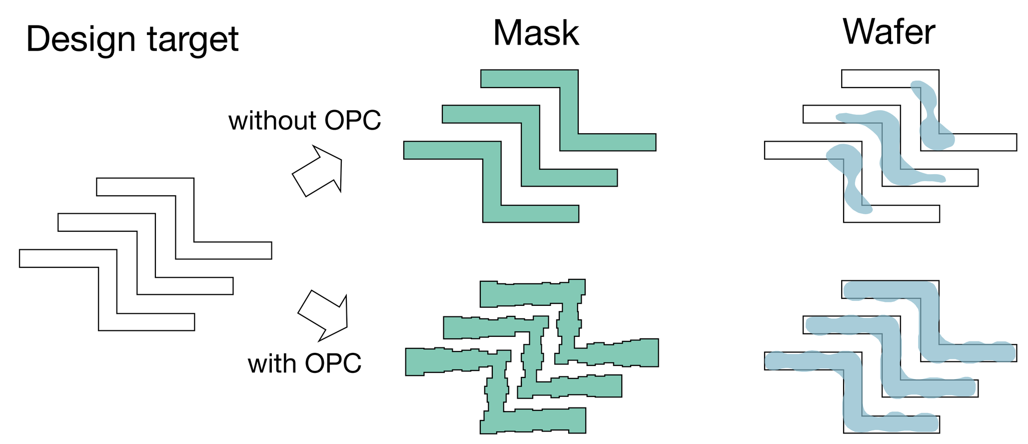
GAN-OPC: Mask Optimization with Lithography-guided Generative Adversarial Nets

Haoyu Yang, Shuhe Li, Yuzhe Ma, Bei Yu and Evangeline F. Y. Young

Department of Computer Science and Engineering
The Chinese University of Hong Kong

Backgrounds

Lithography Proximity Effects



- What you see $\neq$ what you get.
- Diffraction information loss.
- RET: Optical Proximity Correction (OPC), Scatter Bar and Multiple Patterning Lithography.

Classic OPC

Requires iterative call of lithography simulation engine.

- Model/Rule-based OPC [Kuang+,DATE'15][Awad+,DAC'16] [Su+,ICCAD'16]
  1. Fragmentation of shape edges;
  2. Move fragments for better printability.
- Inverse Lithography [Gao+,DAC'14][Poonawala+,TIP'07] [Ma+,ICCAD'17]
  1. Efficient model that maps mask to aerial image;
  2. Continuously update mask through descending the gradient of contour error.

Machine Learning OPC

Masks are updated segment-by-segment and cannot be done in one inference step. [Matsunawa+,JM3'16][Choi+,SPIE'16] [Xu+,ISPD'16][Shim+,APCCAS'16]

1. Edge fragmentation;
2. Feature extraction;
3. Model training.

Preliminaries

Lithography Model

- SVD Approximation of Partial Coherent System [Cobb,1998]

$$\mathbf{I} = \sum_{k=1}^{N^2} w_k |\mathbf{M} \otimes \mathbf{h}_k|^2. \tag{1}$$

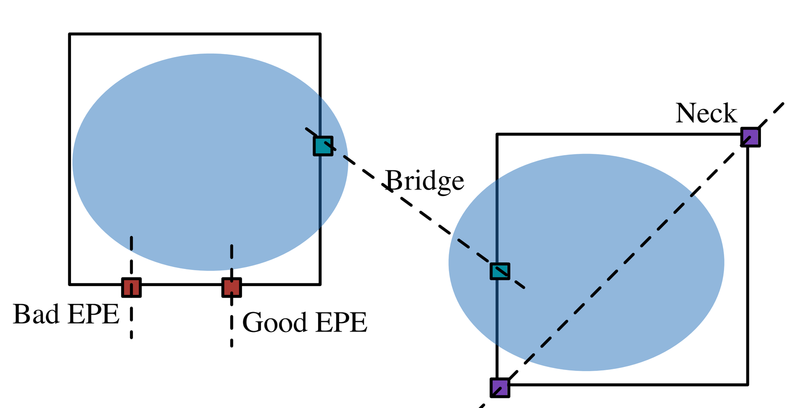
- Reduced Model [Gao+,DAC'14]

$$\mathbf{I} = \sum_{k=1}^{N_h} w_k |\mathbf{M} \otimes \mathbf{h}_k|^2. \tag{2}$$

- Etch Model

$$\mathbf{Z}(x,y) = \begin{cases} 1, \text{ if } \mathbf{I}(x,y) \geq I_{th}, \\ 0, \text{ if } \mathbf{I}(x,y) < I_{th}. \end{cases} \tag{3}$$

Lithography Defects



- EPE measures horizontal or vertical distances from given points (i.e. OPC control points) on target edges to lithography contours.
- Neck detector checks the error of critical dimensions of lithography contours compared to target patterns.
- Bridge detector aims to find unexpected short of wires.

Inverse Lithography

The main objective in ILT is minimizing the lithography error through gradient descent.

$$E = ||\mathbf{Z}_t - \mathbf{Z}||_2^2, \tag{4}$$

where $\mathbf{Z}_t$ is the target and $\mathbf{Z}$ is the wafer image of a given mask.
Apply translated sigmoid functions to make the pixel values close to either 0 or 1.

$$\mathbf{Z} = \frac{1}{1+\exp[-\alpha \times (\mathbf{I} - \mathbf{I}_{th})]}, \tag{5}$$

$$\mathbf{M}_b = \frac{1}{1+\exp(-\beta \times \mathbf{M})}. \tag{6}$$

Combine Equations (1)–(6) and the analysis in [Poonawala,TIP'07],

$$\begin{aligned}\frac{\partial E}{\partial \mathbf{M}} = & 2\alpha\beta \times \mathbf{M}_b \odot (1 - \mathbf{M}_b) \odot \\ & (((\mathbf{Z} - \mathbf{Z}_t) \odot \mathbf{Z} \odot (1 - \mathbf{Z}) \odot (\mathbf{M}_b \otimes \mathbf{H}^*)) \otimes \mathbf{H} + \\ & ((\mathbf{Z} - \mathbf{Z}_t) \odot \mathbf{Z} \odot (1 - \mathbf{Z}) \odot (\mathbf{M}_b \otimes \mathbf{H})) \otimes \mathbf{H}^*).\end{aligned} \tag{7}$$

Mask can then be updated by descending the gradient derived in Equation (7),

$$\mathbf{M} = \mathbf{M} - \gamma \frac{\partial E}{\partial \mathbf{M}}. \tag{8}$$

Generative Adversarial Nets (GAN)

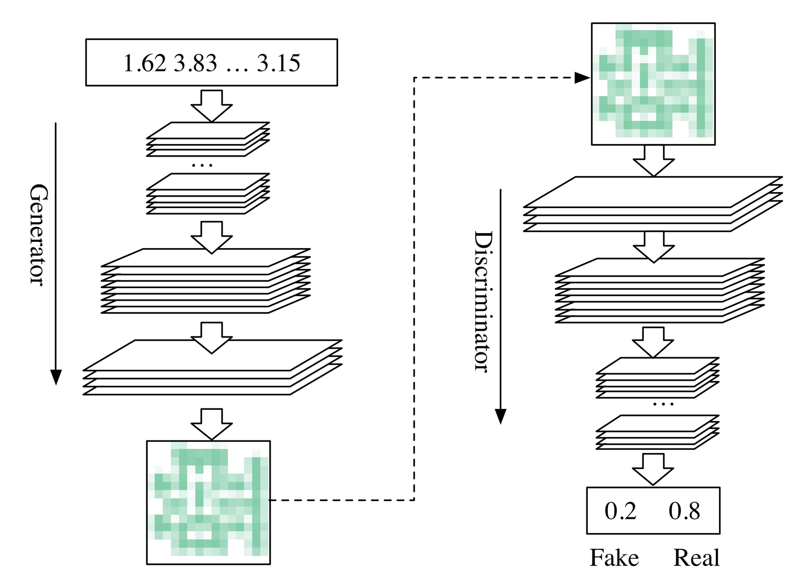
GAN Basis

- $\mathbf{x}$: Sample from the distribution of target dataset; $\mathbf{z}$: Input of G
- Generator $G(\mathbf{z};\theta_g)$: Differentiable function represented by a multilayer perceptron with parameters θ_g.
- Discriminator $D(\mathbf{x};\theta_d)$: Represents the probability that $\mathbf{x}$ came from the data rather than G.

1. Train D to maximize the probability of assigning the correct label to both training examples and samples from G.
2. Train G to minimize $\log(1 - D(G(\mathbf{z})))$, i.e. generate faked samples that are drawn from similar distributions as $p_{data}(\mathbf{x})$.

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]. \tag{9}$$

GAN Architecture



GAN-OPC

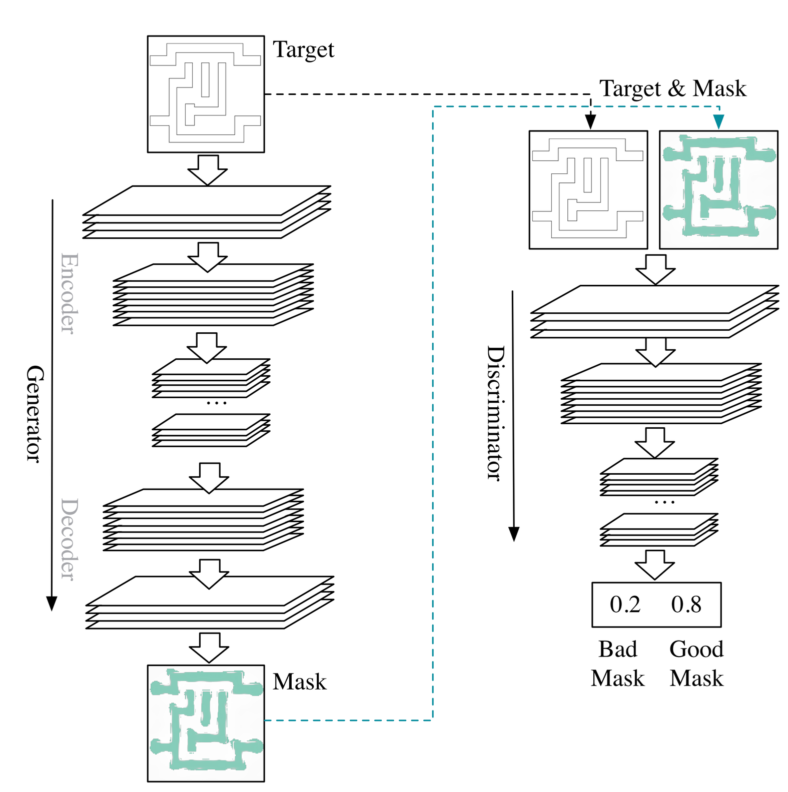
Generator Design

- Auto-encoder based generator which consists of an encoder and a decoder subnets.
- An encoder is a stacked convolutional architecture that performs hierarchical layout feature abstraction.
- A decoder operates in an opposite way that predicts the pixel-based mask correction with respect to the target.

Discriminator Design

- Take target-mask tuple as inputs: $(\mathbf{Z}_t, \mathbf{G}(\mathbf{Z})_t)$ or $(\mathbf{Z}_t, \mathbf{M}^*)$.

GAN-OPC Architecture



GAN-OPC Training

Based on the OPC-oriented GAN architecture in our framework, we tweak the objectives of $\mathbf{G}$ and $\mathbf{D}$ accordingly,

$$\max \mathbb{E}_{\mathbf{Z}_t\sim\mathcal{Z}}[\log(\mathbf{D}(\mathbf{Z}_t, \mathbf{G}(\mathbf{Z}_t)))], \tag{10}$$

$$\max \mathbb{E}_{\mathbf{Z}_t\sim\mathcal{Z}}[\log(\mathbf{D}(\mathbf{Z}_t, \mathbf{M}^*))] + \mathbb{E}_{\mathbf{Z}_t\sim\mathcal{Z}}[1 - \log(\mathbf{D}(\mathbf{Z}_t, \mathbf{G}(\mathbf{Z}_t)))]. \tag{11}$$

In addition to facilitate the training procedure, we minimize the differences between generated masks and reference masks when updating the generator as in Equation (12).

$$\min \mathbb{E}_{\mathbf{Z}_t\sim\mathcal{Z}}||\mathbf{M}^* - \mathbf{G}(\mathbf{Z}_t)||_n, \tag{12}$$

where $||\cdot||_n$ denotes the l_n norm. Combining (10), (11) and (12), the objective of our GAN model becomes

$$\begin{aligned}\min_{\mathbf{G}} \max_{\mathbf{D}} & \ \mathbb{E}_{\mathbf{Z}_t\sim\mathcal{Z}}[1 - \log(\mathbf{D}(\mathbf{Z}_t, \mathbf{G}(\mathbf{Z}_t))) + ||\mathbf{M}^* - \mathbf{G}(\mathbf{Z}_t)||_n^n] \\ & + \mathbb{E}_{\mathbf{Z}_t\sim\mathcal{Z}}[\log(\mathbf{D}(\mathbf{Z}_t, \mathbf{M}^*))].\end{aligned} \tag{13}$$

The generator and the discriminator are trained alternatively as follows.

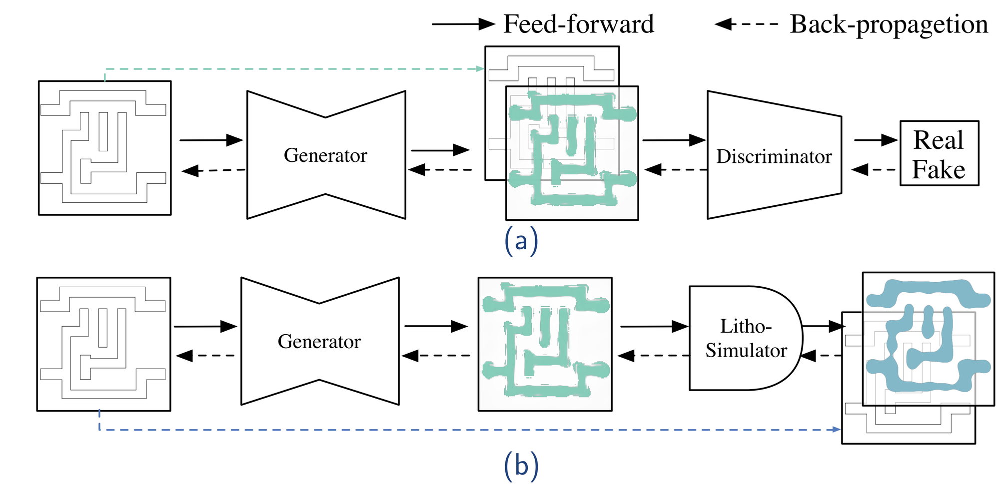
The GAN-OPC Training Algorithm

for number of training iterations **do**
  Sample m target clips $\mathcal{Z} \leftarrow \{\mathbf{Z}_{t,1}, \mathbf{Z}_{t,2}, \dots, \mathbf{Z}_{t,m}\}$;
  $\Delta\mathbf{W}_g \leftarrow \mathbf{0}, \Delta\mathbf{W}_d \leftarrow \mathbf{0}$;
  for each $\mathbf{Z}_t \in \mathcal{Z}$ **do**
    $\mathbf{M} \leftarrow \mathbf{G}(\mathbf{Z}_t; \mathbf{W}_g)$;
    $\mathbf{M}^* \leftarrow$ Groundtruth mask of $\mathbf{Z}_t$;
    $l_g \leftarrow -\log(\mathbf{D}(\mathbf{Z}_t, \mathbf{M})) + \alpha||\mathbf{M}^* - \mathbf{M}||_2^2$;
    $l_d \leftarrow \log(\mathbf{D}(\mathbf{Z}_t, \mathbf{M})) - \log(\mathbf{D}(\mathbf{Z}_t, \mathbf{M}^*))$;
    $\Delta\mathbf{W}_g \leftarrow \Delta\mathbf{W}_g + \frac{\partial l_g}{\partial \mathbf{W}_g}$; $\Delta\mathbf{W}_d \leftarrow \Delta\mathbf{W}_d + \frac{\partial l_d}{\partial \mathbf{W}_g}$;
  end for
  $\mathbf{W}_g \leftarrow \mathbf{W}_g - \frac{\lambda}{m}\Delta\mathbf{W}_g$; $\mathbf{W}_d \leftarrow \mathbf{W}_d - \frac{\lambda}{m}\Delta\mathbf{W}_d$;
end for

ILT-guided Pretraining

Why Pretraining?

- ILT and neural networks optimization share similar gradient descent procedure.
- Let the generator get prior knowledge from the lithography engine.



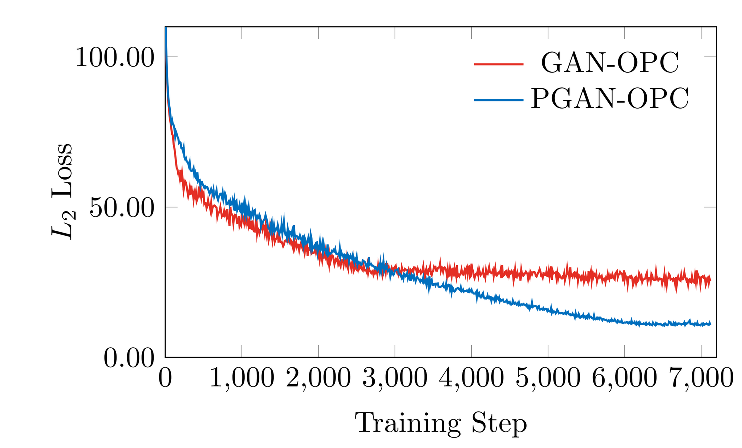
ILT-guided Pretraining

Equation (8) is naturally compatible with mini-batch gradient descent, if we create a link between the generator and ILT engine, the wafer image error can be back-propagated directly to the generator as illustrated above.

ILT-guided Pretraining

for number of pre-training iterations **do**
  Sample m target clips $\mathcal{Z} \leftarrow \{\mathbf{Z}_{t,1}, \mathbf{Z}_{t,2}, \dots, \mathbf{Z}_{t,m}\}$;
  $\Delta\mathbf{W}_g \leftarrow 0$;
  for each $\mathbf{Z}_t \in \mathcal{Z}$ **do**
    $\mathbf{M} \leftarrow \mathbf{G}(\mathbf{Z}_t; \mathbf{W}_g)$;
    $\mathbf{Z} \leftarrow$ `LithoSim`$(\mathbf{M})$ ▷ Equations (2)–(3)
    $E \leftarrow ||\mathbf{Z} - \mathbf{Z}_t||_2^2$;
    $\Delta\mathbf{W}_g \leftarrow \Delta\mathbf{W}_g + \frac{\partial E}{\partial \mathbf{M}}\frac{\partial \mathbf{M}}{\partial \mathbf{W}_g}$; ▷ Equation (7)
  end for
  $\mathbf{W}_g \leftarrow \mathbf{W}_g - \frac{\lambda}{m}\Delta\mathbf{W}_g$; ▷ Equation (8)
end for

Training Behavior:

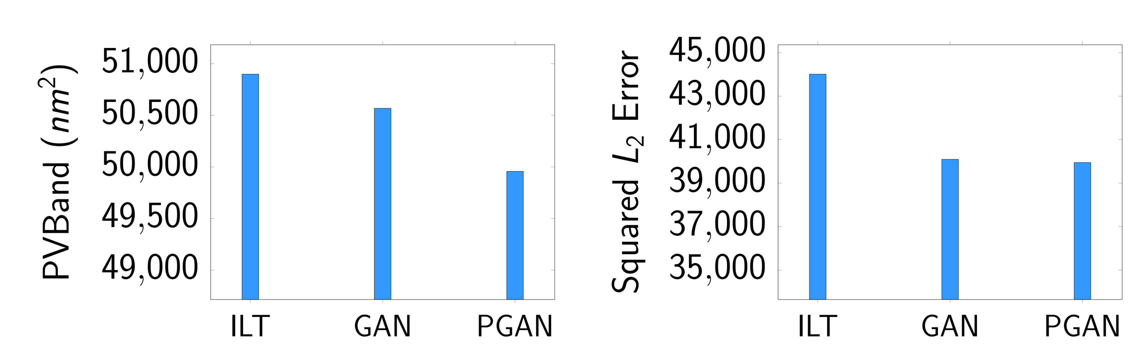


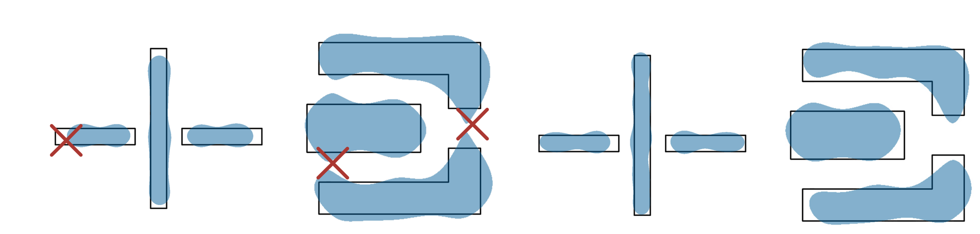
Experimental Results

The Dataset

- The lithography engine is based on the `lithosim v4` package from ICCAD 2013 CAD Contest.
- Manually generated 4000 instances based on the design specfication from existing $32nm$ M1 layouts.

Mask Optimization Results





(a) (b)

Visualizing PGAN-OPC and ILT:

(a) masks of [Gao+,DAC'14]; (b) masks of PGAN-OPC; (c) wafer images by masks of [Gao+,DAC'14]; (d) wafer images by masks of PGAN-OPC; (e) target patterns.