DAMO: Deep Agile Mask Optimization for Full Chip Scale

Guojin Chen, Wanli Chen, Yuzhe Ma, Haoyu Yang, Bei Yu

The Chinese University of Hong Kong
Guojin Chen
The Chinese University of Hong Kong.

I am an MSc student at The Chinese University of Hong Kong (CUHK) and broadly study foundational topics and applications in machine (sometimes deep) learning in VLSI and optimization, including reinforcement learning, computer vision. I am advised by Prof. Bei Yu. I received my Bachelor Degree of Computer Science from Huazhong University of Science and Technology.
Outline

Introduction and Background

Previous work

DAMO
  Dataset Generation
  DCGAN-HD
  DLS
  DMG
  Full-chip Splitting Algo.

Results
  On our datasets
  On ISPD 2019 full-chip datasets
Outline

Introduction and Background

Previous work

DAMO
Dataset Generation
DCGAN-HD
DLS
DMG
Full-chip Splitting Algo.

Results
On our datasets
On ISPD 2019 full-chip datasets
Optical proximity correction (OPC) is a photolithography enhancement technique commonly used to compensate for image errors due to diffraction or process effects.
**Problem definition:** Given a design image $\vec{w}$, the objective of mask optimization is generating the corresponding mask $\vec{x}$ such that remaining patterns $\vec{y}$ after lithography process is as close as $\vec{w}$ or, in other words, minimizing PV Band and squared $L_2$ error of lithography images.
Our DAMO: Deep Agile Mask Optimization for Full Chip Scale

Two main steps:
OPC and Litho: DMG and DLS

Diagram showing the process from Design to Wafer with DMG, DLS, and Inverse Correction.
Outline

Introduction and Background

Previous work

DAMO
  Dataset Generation
  DCGAN-HD
  DLS
  DMG
  Full-chip Splitting Algo.

Results
  On our datasets
  On ISPD 2019 full-chip datasets
cGAN

Objective function

\[ \mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x, y}[\log D(x, y)] + \mathbb{E}_{x, z}[\log(1 - D(x, G(x, z))]. \]

4 Experimental Results

4.1 Unimodal

We trained a conditional adversarial net on MNIST images conditioned on their class labels, encoded as one-hot vectors. In the generator net, a noise prior \( z \) with dimensionality 100 was drawn from a uniform distribution within the unit hypercube. Both \( z \) and \( y \) are mapped to hidden layers with Rectified Linear Unit (ReLU) activation \[4, 11\], with layer sizes 200 and 1000 respectively, before both being mapped to a second, combined hidden ReLU layer of dimensionality 1200. We then have a final sigmoid unit layer as our output for generating the 784-dimensional MNIST samples.

For now we simply have the conditioning input and prior noise as inputs to a single hidden layer of a MLP, but one could imagine using higher order interactions allowing for complex generation mechanisms that would be extremely difficult to work with in a traditional generative framework.
OPC stage previous work: GAN-OPC

GAN-OPC flow: generator inference and ILT refinement.

![Diagram showing the GAN-OPC flow: generator inference and ILT refinement.](image-url)
Litho stage previous work: LithoGAN

The proposed framework for lithography modeling is implemented using CGAN, which has demonstrated proven success in image generation tasks. CGAN has a series of layers that progressively downsample the input until a series of layers that progressively downsample the input until a final output of the LithoGAN framework.

In this work, we train both the generator and the discriminator to improve the accuracy of the model. The generator uses ReLU, whereas the decoder uses ReLU. The discriminator is a convolutional neural network that performs classification (BN).

During training, the golden pattern is re-centered at the exact location of the car in the image. However, for the lithography application, locations of the objects in the generated images are not a major concern. For example, when trained to generate images conditioned on the input images. However, for traditional computer vision tasks, locations of the objects in the generated images are not a major concern.

The objective function for CGAN is given by:

$$\min_G \max_D \mathbb{E}_{x \sim P_{data}}[\log D(x, y)] + \mathbb{E}_{z \sim P_z}[\log(1 - D(x, G(x, z)))] + \lambda \cdot \mathbb{E}_{x, y, z}$$

where $G$ and $D$ are the generator and discriminator, respectively, $x$ is the input image, $y$ is the corresponding label, $z$ is a random noise vector, and $\lambda$ is a hyperparameter.

In our experiments, we set $\lambda = 100$. The learning rate and the momentum parameters in the Adam optimizer are set to $0.0001$ and $0.9$, respectively. The batch size is set to 4 and the number of maximum training epochs is set to 80. The weight parameter is initialized to zero.

We evaluated the performance of our model using standard metrics such as accuracy, precision, recall, and F1-score. The results are reported as the average of five runs, each with different random seeds to eliminate random performance variation. The results reported in this section are the average of the five runs.

In our experiments, we compared the performance of our model against several state-of-the-art methods. The results show that our model outperforms the competing methods by a significant margin.

To guarantee highly accurate resist patterns, the resist patterns generated by rigorous simulation are considered as the golden results. To maintain consistency, we adopt after each simulation. In other words, obtaining the golden resist pattern for each contact in a mask layout requires one model evaluation. Using LithoGAN requires one model evaluation, while the remaining 25% clips are for testing. In our experiments, we set the batch size to 4 and the number of maximum training epochs to 999.

The results reported in this section are the average of the five runs. Note that we train the CGAN and LithoGAN models separately, and the remaining 25% clips are for testing. In our experiments, we set the batch size to 4 and the number of maximum training epochs to 999. The weight parameter is initialized to zero. The learning rate and the momentum parameters in the Adam optimizer are set to $0.0001$ and $0.9$, respectively.

In such a way, the shape and the center of the resist pattern are predicted from the CNN. The resulting adjusted image is the final output of the LithoGAN framework.
Issues of previous work.

- Only initial solution, rely on the traditional ILT-model, time-consuming.
- Only targets a single shape within a clip, limited usage in general OPC tasks.
- Only small single clip, low resolution (256 × 256 pixels).
Outline

Introduction and Background

Previous work

DAMO
   Dataset Generation
   DCGAN-HD
   DLS
   DMG
   Full-chip Splitting Algo.

Results
   On our datasets
   On ISPD 2019 full-chip datasets
Solution: DAMO

- DAMO: End-to-end mask optimization framework without using traditional model.
- DCGAN-HD: High resolution cGAN model.
- Full-chip splitting algorithm for large layout.
Generate Training set
DCGAN-HD: solution for higher resolution

- High-resolution Generator
- Multi-scale discriminator
- Perceptual Loss
High-resolution Generator of DCGAN-HD

Arch.

- UNet++ with Residual blocks.
- High resolution of $1024 \times 1024$. 

DCUNet++

- High-resolution Generator
- UNet++ Backbone
- Encoder
- Residual Blocks
- Decoder
- Convolution
- Deconvolution
- Residual
Multi Scale discriminator

Arch.

- Two discriminators at different input size, $D_1$, $D_2$.
- High resolution of $1024 \times 1024$ and $512 \times 512$.
- Helps the training of high-resolution model easier.
Perceptual Loss

\[
\mathcal{L}^G_{L_p} (\bar{x}, \hat{x}) = \mathcal{L}_{L_1}(\Phi(\bar{x}), \Phi(\hat{x})) = \mathbb{E}_{\bar{x}, \hat{x}} \left[ \|\Phi(\bar{x}) - \Phi(\hat{x})\|_1 \right],
\]

(2)
DLS training

\[
\mathcal{L}_{DLS} = \sum_{k=1,2} \mathcal{L}_{cGAN}(G, D_k) + \lambda_0 \mathcal{L}_{L_0}^{G,\Phi} (y, \hat{y}).
\] (3)
DMG training

\[ \mathcal{L}_{DMG} = \sum_{k=1,2} \mathcal{L}_{cGAN}(G_{DMG}, (D_{DMG})_k) + \lambda_1 \mathcal{L}_{L_p}^{G_{DMG}, \Phi}(x, \hat{x}). \]  

(4)

\[ \mathcal{L}_{DAMO} = \mathcal{L}_{DMG} + \mathcal{L}_{DLS} + \lambda_2 \mathcal{L}_{L_1}(\hat{y}, w_r). \]  

(5)
Full-chip Splitting Algo: Coarse to Fine, DBSCAN to KMeans

Algo. detail

1. DBSCAN then KMeans++
2. Initialize the number of centroids from 1 to \( V \) to run KMeans++.
3. Every cluster contains no more than \( K \) via patterns.
4. Every via pattern must be contained in a window.
5. If (3) or (4) is not satisfied, increase the centroid number.

Algo. figure
Coarse step, DBSCAN

**Algo. detail**

DBSCAN algorithm is used to initially detect the clusters of via patterns. After the coarse step, the via patterns in a large layout will be assigned into different DBSCAN clusters.
Fine step, KMeans++

Algo. detail

Then we search every coarse cluster and run KMeans++ algorithm to find the best splitting windows.
The split chips.

Algo. detail

Every KMeans cluster belongs to a 1024 × 1024 nm² chip, whose center locates at the centroid of the KMeans cluster.
Main Contribution

- **DCGAN-HD**: we extend cGANs model by redesign the generator and discriminator for high resolution input (1024*1024), combined with a novel window-splitting algorithm, our model can handle input layout of any size with high accuracy.

- **DLS**: We build up a deep lithography simulator (DLS) based on our DCGAN-HD. Thanks to the express power of stack convolution layers, DLS is expected to conduct lithography simulation faster with similar contour quality compared to legacy lithography simulation process.

- **DAMO**: We present DAMO, a unified end-to-end trainable OPC engine that employs both DLS and DMG to conduct full-chip scale mask optimization without further fine-tune with legacy OPC engines.

- Experimental results show that the proposed DAMO framework is able to output high quality lithography contours more efficiently than Calibre, which also derives $\sim 4\times$ speed-up in OPC tasks while generating masks with even better printability.
Outline

Introduction and Background

Previous work

DAMO
  Dataset Generation
  DCGAN-HD
  DLS
  DMG
  Full-chip Splitting Algo.

Results
  On our datasets
  On ISPD 2019 full-chip datasets
Results on self-generated datasets
### Table 2: Comparison with State-of-the-art on validation set

<table>
<thead>
<tr>
<th>Bench case#</th>
<th>Bench case#</th>
<th>GAN-OPC $L_2 (nm^2)$ PV Band ($nm^2$) runtime (s)</th>
<th>Calibre $L_2 (nm^2)$ PV Band ($nm^2$) runtime (s)</th>
<th>DAMO $L_2 (nm^2)$ PV Band ($nm^2$) runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-via 500</td>
<td>1464</td>
<td>3064</td>
<td>1084</td>
<td>1080</td>
</tr>
<tr>
<td>2-via 500</td>
<td>4447</td>
<td>6964</td>
<td>2161</td>
<td>2129</td>
</tr>
<tr>
<td>3-via 500</td>
<td>8171</td>
<td>11426</td>
<td>3350</td>
<td>3244</td>
</tr>
<tr>
<td>4-via 500</td>
<td>11659</td>
<td>14958</td>
<td>4331</td>
<td>4263</td>
</tr>
<tr>
<td>5-via 500</td>
<td>15773</td>
<td>18976</td>
<td>5410</td>
<td>5396</td>
</tr>
<tr>
<td>6-via 500</td>
<td>18904</td>
<td>22371</td>
<td>6647</td>
<td>5981</td>
</tr>
<tr>
<td>Average</td>
<td>10069</td>
<td>12960</td>
<td>3831</td>
<td>3682</td>
</tr>
<tr>
<td>Ratio</td>
<td>2.735</td>
<td>1.367</td>
<td>1.040</td>
<td>1.00</td>
</tr>
</tbody>
</table>

25/29
Results on ISPD 2019 datasets
### Results on ISPD 2019 datasets

#### Table 3: Comparison on ISPD 2019 full-chip splitting windows

<table>
<thead>
<tr>
<th>Bench</th>
<th>case#</th>
<th>GAN-OPC $L_2 (nm^2)$</th>
<th>GAN-OPC PV Band ($nm^2$)</th>
<th>GAN-OPC runtime (s)</th>
<th>Calibre $L_2 (nm^2)$</th>
<th>Calibre PV Band ($nm^2$)</th>
<th>Calibre runtime (s)</th>
<th>DAMO $L_2 (nm^2)$</th>
<th>DAMO PV Band ($nm^2$)</th>
<th>DAMO runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISPD-1-via</td>
<td>6116</td>
<td>2367</td>
<td>3492</td>
<td>3963</td>
<td>1073</td>
<td>2857</td>
<td>18959</td>
<td>1056</td>
<td>2848</td>
<td>3669</td>
</tr>
<tr>
<td>ISPD-2-via</td>
<td>2652</td>
<td>5412</td>
<td>7126</td>
<td>1742</td>
<td>2232</td>
<td>5670</td>
<td>7537</td>
<td>2172</td>
<td>5654</td>
<td>1591</td>
</tr>
<tr>
<td>ISPD-3-via</td>
<td>1582</td>
<td>8792</td>
<td>13047</td>
<td>1021</td>
<td>3602</td>
<td>8276</td>
<td>4494</td>
<td>3196</td>
<td>8127</td>
<td>949</td>
</tr>
<tr>
<td>ISPD-4-via</td>
<td>522</td>
<td>12395</td>
<td>15015</td>
<td>341</td>
<td>4395</td>
<td>11051</td>
<td>1692</td>
<td>4361</td>
<td>10987</td>
<td>313</td>
</tr>
<tr>
<td>ISPD-5-via</td>
<td>777</td>
<td>16526</td>
<td>19147</td>
<td>495</td>
<td>5526</td>
<td>12305</td>
<td>2537</td>
<td>4542</td>
<td>12251</td>
<td>466</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>9098</td>
<td>11565</td>
<td>1512</td>
<td>3365</td>
<td>8031</td>
<td>7043</td>
<td>3065</td>
<td>7973</td>
<td>1397</td>
</tr>
<tr>
<td>Ratio</td>
<td></td>
<td>2.968</td>
<td>1.451</td>
<td>1.082</td>
<td>1.098</td>
<td>1.007</td>
<td>5.041</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
</tbody>
</table>
Results Visualization

Visualization of DAMO model advancement on via layer:
(c) Epoch 20; (d) Epoch 40; (e) Epoch 60; (f) Epoch 80; (g) Epoch 100.
Thanks

Thank you.