ILILT: Implicit Learning of Inverse Lithography Technologies

Anonymous Authors[1]

Abstract

Lithography, transferring chip design masks to the silicon wafer, is the most important phase in modern semiconductor manufacturing flow. Due to the limitations of lithography systems, Extensive design optimizations are required to tackle the design and silicon mismatch. Inverse lithography technology (ILT) is one of the promising solutions to perform pre-fabrication optimization, termed mask optimization. Because of mask optimization problems' constrained non-convexity, numerical ILT solvers rely heavily on good initialization to avoid getting stuck on sub-optimal solutions. Machine learning (ML) techniques are hence proposed to generate mask initialization for ILT solvers with one-shot inference, targeting faster and better convergence during ILT. This paper addresses the question of *whether ML models can directly generate high-quality optimized masks without engaging ILT solvers in the loop*. We propose an implicit learning ILT framework: ILILT, which leverages the implicit layer learning method and lithography-conditioned inputs to ground the model. Trained to understand the ILT optimization procedure, ILILT can outperform the state-of-the-art academic ILT solver, significantly improving efficiency and quality.

1. Introduction

Lithography is the most important phase in semiconductor manufacturing flow, which transfers chip design patterns onto the silicon wafer. However, due to the limitations of lithography systems, there are usually fetal manufacturing gaps between the design and the patterns on the wafer. Extensive prior fabrication design optimizations, termed *mask optimization*, are required to compensate for the manufacturing loss (Figure 1).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

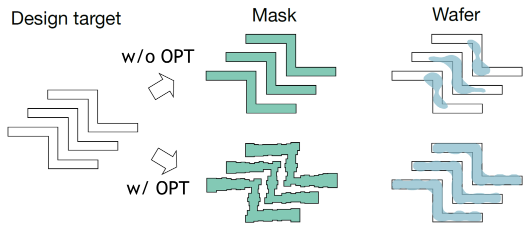
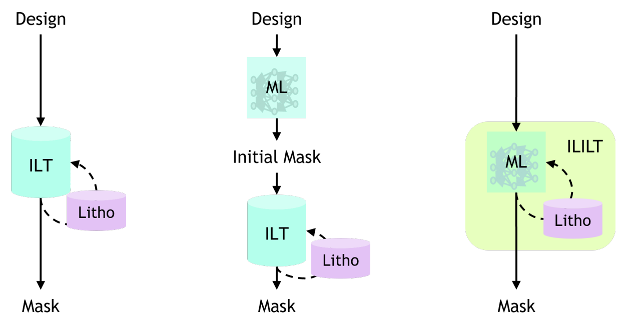


Figure 1. Limitations of lithography system requires design optimization to be correctly fabricated on the silicon wafer.

Inverse lithography technology (ILT), as a numerical solution, has shown great promise for mask optimization on advanced-node lithography (Gao et al., 2014; Yu et al., 2021; Braam et al., 2019; Pang et al., 2019). As shown in Figure 2(a), an ILT solver requires frequent calls of lithography simulator, which mathematically approximate the lithography fabrication behavior. The forward path computes the wafer image corresponding to the current mask and we measure the error between the wafer image and the original design. In the backward path, the error will be passed back through gradient method to update the mask. However, the non-convexity of mask optimization problems poses a great challenge to numerical ILT solvers, that they rely heavily on good mask initialization and can easily get stuck at sub-optimal solutions.

Recent research into machine learning (ML) techniques for lithography have made ILT solvers more efficient. As shown in Figure 2(b), ML can provide better optimization starting points that significantly reduce ILT iterations and yield better convergence (Yang et al., 2018; Chen et al., 2020; Jiang et al., 2020). GAN-OPC (Yang et al., 2018) is the earliest deep learning solution for inverse lithography, where a conditional generative adversarial network is proposed for initial mask generation with litho-guided pre-training. DAMO (Chen et al., 2020) improves on this work by replacing the standard generator in GAN-OPC with nested UNet and ResNet bottleneck layers (Zhou et al., 2019; Isola et al., 2017; He et al., 2016), yielding a direct output of high-resolution masks. Jiang et al. (Jiang et al., 2020) also integrates the explicit gradient calculation of litho-guided pre-training and achieves better training convergence and mask quality. Very recently, Zhao et al. (Zhao et al., 2022) develops the AdaOPC framework that is able to learn from



(a) ILT (b) ML-Aided (c) ILILT

Figure 2. Working scheme of ILT solvers. (a) Standard ILT solver that iteratively updates the mask from design till convergence. (b) Common ML approach that uses ML to produce an initial mask followed by standard ILT procedure. (c) The ILILT that uses ML to imitate the standard ILT procedure.

layout geometry and identifies whether a pattern can be optimized by the machine learning engine. This further increase the applicability of ML for mask optimization solutions.

However, existing solutions only exploit limited ML in the ILT solver and still require significant intervention by traditional numerical solvers. For example, GAN-OPC and its variations can only learn a design-to-mask mapping, which cannot be used to fix poor-quality masks; AdaOPC (Zhao et al., 2022) has to conduct full numerical optimizations on complicated patterns. In this paper, we try to address the question of **whether an ML model can directly generate high quality optimized masks without engaging ILT solvers in the loop**. We believe that the key limitation of existing methods is that the trained ML models are not grounded to any actual lithography condition during inference, therefore training-inference distribution shift can not be recognized and corrected. We argue that **ML models grounded on lithography estimation of intermediate masks can make an ML-based ILT framework more effective.**

We propose the ILILT (Figure 2(c)) framework to formulate mask optimization as an implicit layer learning problem, trained to optimize the mask towards an equilibrium state (Bai et al., 2019). Like a standard ILT solver that iteratively updates masks from a design, ILILT iteratively modifies masks from a design but with much fewer steps and each step runs significantly faster because no gradient computing is required. This process produces a number of intermediate masks which ground the ML model with intermediate wafer images computed by lithography models.

ILT As An Implicit Layer. Unlike common neural network layers that explicitly define specific computation graphs supporting forward and backward propagation, an implicit layer models a joint condition of its input and output (Jittorntrum, 1978). The nature of ILT makes it a representative example of an implicit layer, where the input is a chip design and the output is the corresponding optimized mask that results in a similar wafer pattern compared to the input after the lithography process. Inspired by the implicit layer learning approach, we design ILILT with a weight-tied sequence model, making it possible to learn an optimization trajectory from training design-mask pairs. To make the trajectory learning grounded with physics, we also integrate ILILT with fast lithography estimator that can implicitly apply lithography conditions on top of the optimization procedure. After training, ILILT is able to perform mask optimization on a given design and reaches a stable state in a few iterations. We will later show how ILILT outperforms a state-of-the-art academic ILT engine with technical details and supporting experiments on public chip design benchmarks.

The reminders of the paper are organized as follows: Section 2 covers basic terminologies and related works of mask optimization; Section 3 discusses and analyzes the technique details of the ILILT framework; Section 4 presents the experimental results supporting the effectiveness of ILILT and Section 5 concludes the paper.

2. Preliminaries

2.1. ILT Solver

Mask optimization tries to find a mask design such that the remaining pattern on the silicon wafer after the lithography process is as close as possible to the desired shapes. This can be formulated as a constrained optimization problem:

$$\min_{\boldsymbol{M}} l = d(\boldsymbol{Z}, \boldsymbol{Z}^*), \tag{1}$$

where,

$$\boldsymbol{Z} = f_l(\boldsymbol{M}), \tag{2}$$

where $\boldsymbol{Z}$ and $\boldsymbol{Z}^*$ represent the wafer image and the chip design, respectively, $\boldsymbol{M}$ corresponds to the mask to be optimized, f_l represents the lithography process which is highly non-convex (see appendix for more details) and d is a measurement indicating the differences between $\boldsymbol{Z}$ and $\boldsymbol{Z}^*$. Inverse lithography techniques (Gao et al., 2014) try to solve Problem (1) in a numerical way by repeating the following steps until convergence:

1. Forward: $\boldsymbol{Z}_t = f_l(\boldsymbol{M}_t)$.
2. Backward: $\boldsymbol{M}_{t+1} = \boldsymbol{M}_t - \lambda \dfrac{\partial l}{\partial \boldsymbol{Z}_t} \dfrac{\partial \boldsymbol{Z}_t}{\partial \boldsymbol{M}_t}$.

where λ is some hyper-parameter that controls how the mask image is updated during the optimization procedure. There are also variations of ILTs (Yu et al., 2021; Jiang et al., 2020; Gao et al., 2014), but they mostly share similar mask-update concepts.

2.2. Implicit Layers

Most neural network layers or computational graphs explicitly define a function $\boldsymbol{y} = f(\boldsymbol{x})$, which is differentiable and can be back-propagated via the chain rule. An implicit layer, instead of specifying how to compute the layer's output from the input, specifies the conditions that we want the layer's output to satisfy, separating the procedure to compute the layer from the layer definition itself. It has been used in deep equilibrium models, Neural ODEs, and differentiable optimizations.

Definition 2.1 (Implicit Layer)**.** Implicit layers try to find $\boldsymbol{y}$ such that $f(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{C}$, which defines joint conditions between $\boldsymbol{x}$ and $\boldsymbol{y}$ (Jittorntrum, 1978).

A fixed-point layer is an example of the implicit layer method, which directly computes the fixed point of a nonlinear transformation, i.e., the solution to the nonlinear system which models the convergent state of a recurrent sequence.

Definition 2.2 (Fixed-Point Layer)**.** A fixed-point layer can be mathematically given by

$$\boldsymbol{y} = g(\boldsymbol{x}, \boldsymbol{y}). \tag{3}$$

A fixed-point layer can be considered to be an infinite number of layers parameterized by the same weight $\boldsymbol{w}$ (weight-tied), where each layer can be represented as

$$\boldsymbol{y}_{t+1} = g(\boldsymbol{x}, \boldsymbol{y}_t, \boldsymbol{w}), \tag{4}$$

To train a fixed-point layer, (Bai et al., 2019) proposes an unrolling-free deep equilibrium (DEQ) method, which trains the fixed-point layer in two alternative steps: (1) solve for the fixed-point of the layer and (2) update network parameters. This process consists of repeated forward and backward computations as follows. The forward path is related to the finding of the fixed point of Equation (3) and can be solved with Newton's method. Let

$$h(\boldsymbol{x}, \boldsymbol{y}^*, \boldsymbol{w}) = g(\boldsymbol{x}, \boldsymbol{y}^*, \boldsymbol{w}) - \boldsymbol{y}^*, \tag{5}$$

and $\boldsymbol{y}^*$ can be found through the following step until convergence:

$$\boldsymbol{y}_{t+1} = \boldsymbol{y}_t - \lambda \boldsymbol{J}_t^{-1} h(\boldsymbol{x}, \boldsymbol{y}_t, \boldsymbol{w}), \tag{6}$$

where $\boldsymbol{J}_t$ is the Jacobian matrix of h at $\boldsymbol{y}_t$ and λ is the update rate.

The backward process covers the update of $\boldsymbol{w}$, which can be written as

$$\boldsymbol{w} = \boldsymbol{w} - \lambda \frac{\partial l}{\partial \boldsymbol{w}} = \boldsymbol{w} + \lambda \frac{\partial l}{\partial \boldsymbol{y}^*} \boldsymbol{J}^{*-1} \frac{\partial g}{\partial \boldsymbol{w}}, \tag{7}$$

where $\boldsymbol{J}^*$ is the Jacobian matrix of h at $\boldsymbol{y}^*$, l is a loss function that measures whether $f(x, y) \in \mathcal{C}$ satisfies the implicit layer definition, e.g. the ILT loss Equation (1). For a detailed proof of Equation (7), we refer readers to (Bai et al., 2019).

2.3. Neural Networks for ML-based Computational Lithography

There are two major groups of network architectures used in lithography applications: (1) The UNet family (Ronneberger et al., 2015; Zhou et al., 2019) and (2) The FNO family (Li et al., 2021). UNets consist of various bypass connections between convolution layer outputs. Representative works include TEMPO (Ye et al., 2020), LithoGAN (Ye et al., 2019), DAMO (Chen et al., 2020) and GAN-OPC (Yang et al., 2018). FNOs contain one or more Fourier Neural Operators along with auxiliary convolution layers. Representative works are DOINN (Yang et al., 2022b) and CFNO (Yang et al., 2022a). UNet-based models usually rely on large model capacity to learn training data distribution while FNO-based models focus on global information learning through frequency domain embedding.

3. The ILILT Framework

3.1. ILT Layer

Following the definition of implicit layer learning, we can solve the ILT equations with an implicit ILT layer defined as finding $\boldsymbol{M}$ such that

$$f(\boldsymbol{M}, \boldsymbol{Z}^*) = d(f_l(\boldsymbol{M}), \boldsymbol{Z}^*) = \epsilon, \tag{8}$$

where ϵ is some minimum value achievable for Equation (1).

Because Equation (8) is a composite of a group of non-linear functions, it is difficult to solve explicitly. Therefore, we assume Equation (8) has a solution $\boldsymbol{M}^*$ that follows the form of

$$\boldsymbol{M}^* = g(\boldsymbol{M}^*, \boldsymbol{Z}^*), \tag{9}$$

which yields a fixed-point layer representation of the ILT solution.

We start solving Equation (9) from the weight-tied approach, where we define

$$\boldsymbol{M}_{t+1} = g(\boldsymbol{M}_t, \boldsymbol{Z}^*, \boldsymbol{w}), \quad t = 1, 2, ..., \tag{10}$$

and

$$\lim_{t \to \infty} \boldsymbol{M}_t = \lim_{t \to \infty} g(\boldsymbol{M}_t, \boldsymbol{Z}^*, \boldsymbol{w}) = g(\boldsymbol{M}^*, \boldsymbol{Z}^*, \boldsymbol{w}) = \boldsymbol{M}^*. \tag{11}$$

3.2. Lithography Estimation

As we discussed previously, due to the lack of prior lithography knowledge, state-of-art AI approaches have been limited to only generate initialization for numerical ILT solvers. This motivates us to build lithography estimation inside g to ground the AI model.

We rewrite Equation (10) as :

$$\boldsymbol{M}_{t+1} = g(\boldsymbol{M}_t, \boldsymbol{Z}_t, \boldsymbol{Z}^*, \boldsymbol{w}), \ \ t = 1, 2, ..., \tag{12}$$

where $\boldsymbol{Z}_t = f_l(\boldsymbol{M}_t)$ is the wafer image of $\boldsymbol{M}_t$ at time stamp t. Equation (12) formulates the core of the ILILT framework. A virtue of Equation (12) is the embedded condition of the lithography estimator that implicitly tells g the physical meaning of each step.

3.3. Training

Equation (12) defines a weight-tied fixed-point model that can be solved with the DEQ method. Although DEQ (Bai et al., 2019) reduces the computing overhead of sequence unrolling and back-propagation through time, there are several concerns when it's used for solving ILT problems:

- Solving $\boldsymbol{M}^*$ through Newton's method or other Root-Finding algorithms is still computationally costly considering that $\boldsymbol{M}^*$ usually has millions of entries.
- The solution error caused by the non-optimality in Newton's method will further propagate to the gradient calculation in backward DEQ, inducing additional errors.
- The DEQ backward pass still requires the computation of the gradient over lithography modeling in the term $\frac{\partial d}{\partial \boldsymbol{M}^*}$, showing no advantage over a traditional ILT solver.

To address these concerns, we developed the following training architecture that solves the ILT layer efficiently and effectively. We unroll ILILT to a fixed depth T and allow the final output $\boldsymbol{M}_T$ to be trained towards a ground truth mask $\boldsymbol{M}^*$,

$$\begin{aligned} \min_{\boldsymbol{w}} \ \ & l = \sum_{t=\lceil \frac{T}{2} \rceil}^{T} \exp[\frac{t}{T} - 1] \cdot ||\boldsymbol{M}_t - \boldsymbol{M}^*||_F^2, \\ \text{s.t.} \ \ & \boldsymbol{M}_{t+1} = g(\boldsymbol{M}_t, \boldsymbol{Z}_t, \boldsymbol{Z}^*, \boldsymbol{w}), \ \ t = 0, 1, ..., T-1, \\ & \boldsymbol{Z}_t = f_l(\boldsymbol{M}_t), \ \ t = 0, 1, ..., T. \end{aligned} \tag{13}$$

It should be noted that to encourage the mask to grow towards and stop at the golden solution, we did not adopt the cost function that directly minimizes the difference between $\boldsymbol{M}_T$ and $\boldsymbol{M}^*$. Instead, we ask intermediate masks in later unrolling stages to have a trend growing toward the ground truth. This trick allows the mask to grow smoothly in these steps while stopping at the fixed point. The exponential term is empirically chosen to regularize the mask-growing trends which are consistent with the behavior of the numerical ILT solver that only minor changes are observed in later optimization iterations. Solving Equation (13) is straightforward through back-propagation through time (BPTT).

3.4. Discussion

3.4.1. Relations to ML-Aided Solutions

For the training procedure, we observe that ILILT resembles GAN-OPC (Yang et al., 2018) in terms of the training objectives, where we also force the model to learn a golden mask generated by some conventional methods. However, ILILT advances beyond GAN-OPC works (Yang et al., 2018; Jiang et al., 2020; Zhao et al., 2022; Chen et al., 2020) because it not only learns what a good mask should look like, but also **why and how a good mask is grown** thanks to the implicit lithography condition introduced by $f_l(\cdot)$.

3.4.2. Relations to Implicit Layer Learning

The whole ILILT method is built upon the definition of the ILT layer and how it is solved. Representative solutions, e.g. DEQ (Bai et al., 2019), still requires costly computation of gradients of the objectives over physical models. ILILT makes it **even more implicit** by feeding the wafer image, target design and the mask image simultaneously into $g(\cdot, \boldsymbol{w})$ at each time stamp. Finally $g(\cdot, \boldsymbol{w})$ will automatically figure out the inherent relations between masks, designs and resist patterns.

3.4.3. Relations to Learning-to-Optimize

The ILILT system is a similar scheme to learning-to-optimize (L2O) solutions (Chen et al., 2021), particularly the algorithm unrolling approach, which unrolls a truncated algorithm into multi-stage neural networks with independent parameters. In contrast, ILILT benefits from the **weight-tied structure** that (1) significantly reduces model size for memory efficiency and (2) generalizes better (Bai et al., 2019).

3.4.4. Practical Usage Scenario

Lastly, ILILT imitates a real ILT solver during the inference runtime but with many benefits: (1) ILILT requires to query a lithography simulator in each step but updates the mask in a derivative-free manner. (2) ILILT requires significant less steps than traditional ILT solver. (3) ILILT can serve as a real solver during chip design optimization. One representative case is when a design is not well-optimized by an existing mask optimizer. Because most AI solutions only learn a design-to-mask mapping, it is not possible to use them to fix partially optimized masks. ILILT, however, does not have this limitation, and can take over from anywhere in the optimization trajectory.

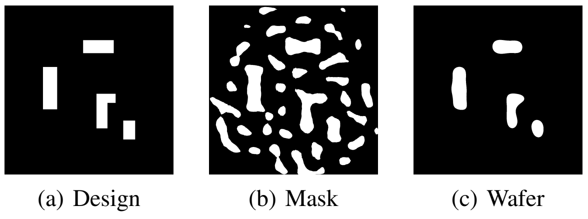


Figure 3. Dataset examples from LithoBench (Zheng et al., 2023). (a) Design image contains the patterns of original circuit devices or connectivity; (b) Mask image is the optimized design that is manufacturing-friendly; (c) Wafer image is the patterns on silicon given the mask image.

4. Experiments

4.1. The Dataset and Configurations

To demonstrate the effectiveness of the ILILT framework, we adopt the latest AI for computational lithography benchmark suite `LithoBench` (Zheng et al., 2023), where state-of-the-art mask optimization solutions are evaluated and will be used as comparison baselines. LithoBench contains over 10K design clips and each covers an area of $2\mu m \times 2\mu m$. Each data point contains a pair of chip design patterns and the ground truth masks generated by a numerical ILT solver. These design patterns and masks are rasterized with a $1nm^2$ pixel size, yielding 2000×2000 images. Example designs are displayed in Figure 3. We can observe that in real world, mask optimization does not simply finetune the design like in Figure 1. The process also produce *sub-resolution assist features* (SRAFs) surrounding the main design pattern to improve its robustness against process variations. This poses great challenge of ML models on optimization tasks.

To deploy ILILT, we reimplement the lithography simulator as a pytorch layer that can be integrated in the data pipeline. Detailed configurations of the generator used to reproduce our results are listed in Table 1. Here we take two representative backbones CFNO (Yang et al., 2022a) and UNet (Ronneberger et al., 2015) as $g(\cdot, \boldsymbol{w})$ and test their performance respectively.

4.2. Evaluation Metrics

As is typical for mask optimization tasks, we evaluate the final performance through two popular metrics: edge placement error (EPE) violations and the process variation band (PVB) area. In semiconductor manufacturing processes, the lithography conditions are not always as we expected due to systematic error and mechanical perturbations, which is the root cause of process variations. Therefore, we expect the design to be robust under different manufacturing conditions. Typically, we call the expected lithography settings and configurations *nominal condition*, and non-nominal condition will result in larger or smaller patterns on the silicon wafer. The difference between the smallest pattern and the largest pattern will form a process variation band (PVB) as in Figure 4. A good mask optimization result should have smaller number of EPE violations and smaller PVB area.

Table 1. Model Configuration.

Item	g
Structure	CFNO / Unet
Max Epoch	5
Initial Learning Rate	1.0
Learning Rate Decay Policy	cosine
Optimizer	Prodigy [13]
Weight Decay	0.0001
Loss	Equation (13)
Sequence Unrolling Depth (T)	4-8
Batch Size	4

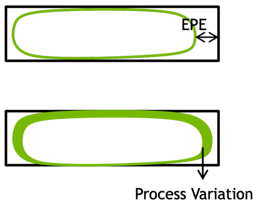


Figure 4. Lithography edge placement error and process variations.

Definition 4.1 (EPE Violation(Banerjee et al., 2013))**.** EPE is measured as the geometric distance between the target edge and the lithographic contour printed at the nominal condition. If the EPE measured at a point is greater than certain tolerance value, we call it an EPE violation.

Definition 4.2 (PVB Area(Banerjee et al., 2013))**.** This is evaluated by running lithography simulation at different process condition on the final mask solution. Once run, the total area of the process variation band will be defined as PVB Area.

4.3. Comparison with State-of-the-Art

In the first experiment, we compare the performance of ILILT with state-of-the-art mask optimizers. A quantitative comparison is listed in Table 2. Columns "EPE", "PVB" and "Time (s)" indicate the count of EPE violations, PVB Area in terms of nm^2 and optimization runtime, respectively; "AdaOPC (Zhao et al., 2022)" is the latest AI-assisted ILT engine that avoids rigorous solver runs on non-critical patterns; "ILT-GPU (Gao et al., 2014)" is a CUDA implementation of the numerical ILT solver in (Gao et al., 2014) that comes with faster iteration time and same QoR compared to the original version. "DAMO (Chen et al., 2020)" is a recent generative model for mask optimization that leverages

UNet++ and ResNet structures. Overall, the ILILT achieves the smallest EPE violation count among all other solutions with 15.7 average EPE violation for 10 test cases, while the state-of-the-art AdaOPC exhibits 24.9 average EPE violations. All methods do not show significant advantage on PVB area, and this is because the optimization flow in these solutions do not explicitly target process variations. Lastly, the ILILT already achieves better mask quality than other three solutions without the refinement from legacy ILT engine. Therefore we show significant speedup over previous methods with $10\times$ over AI-based approaches and $100\times$ over numerical methods.

4.4. Generalization of ILILT

In this experiment, we examine the generalization benefits of ILILT. We show that although ILILT does not achieve the best on other network structures but can still boost the results. We use UNet and CFNO as two base network choices and show how the mask optimization results can be improved when embedded into ILILT. Detailed results are shown in Table 3, where columns "UNet" and "CFNO" list the results using UNet and CFNO only as a design-to-mask mapping function, and columns "ILILT-*" correspond to using the specific architecture as the base optimizer network g and plugging them into ILILT flow. We can observe that with the weight-tied sequence training on UNet, the mask quality has an improvement of 8% reduced EPE violation and 4% reduced PVB Area. The benefits are even higher when we use CFNO as the base optimizer with 30% fewer EPE violations. Although the unrolled sequence induces additional runtime, the model inference time can be ignored when compared to the entire physical implementation and manufacturing flow.

4.5. ILILT Optimization Flow

The learned mask optimization process is depicted in Figure 5, where we visualize the growth of a mask along the unrolled sequence for a given design. We show that the mask reaches the best quality at the configured unrolling depth. Interestingly, the ILILT also learns a fixed-point iteration such that if we continuously feed the mask beyond the unrolling depth, the model will no longer make significant changes to the mask and converges to an EPE violation count of 13. This property brings more application scenario of the ILILT framework. *It is also worth noting that the implicit lithography model contributes to the fixed point iterations and the unrolled sequence will diverge beyond the maximum unrolling depth without the help of lithography estimator.* We can also visualize the predicted resist images during the optimization runtime and see that the lithography estimator is able to conduct satisfactory resist prediction on intermediate masks. Though Mask2 has the smallest EPE, we still select the converged mask as our outputs to avoid rigorous examination of all intermediate results. Quality fluctuation is also common in numerical ILT solvers (Gao et al., 2014; Ma et al., 2017; Yu et al., 2021).

4.6. Investigation of Unrolling Depth

In the last experiment, we discuss the effects of the maximum unrolling depth. The ILILT is trained using different max unrolling depth (T) with $T \in \{2, 4, 6, 8\}$. With PVB area being similar, we observe that the best EPE violation count occurs at T = 4 in the configuration. When $T = 2$, ILILT does not benefit too much from the weight-tied structure and exhibits larger EPE violation counts. As discussed previously, a larger T will put stronger regularization on the model and enable temporal feature learning to increase the model generality. However, we observe performance degradation when $T = 6$ and $T = 8$. This is because a longer unrolling sequence will make the training process more challenging, yielding a much slower training convergence.

5. Conclusion

In this paper, we propose a novel solution to the mask optimization problem with inverse lithography technology. We argue that existing AI-based approaches are overly reliant on conventional numerical solvers and lack forward lithography conditions, limiting the applicability of AI-based approaches to mask optimization scenarios. To further exploit the potential of AI and machine learning, we carefully analyze the characteristics of the ILT procedure and reformulate it as an implicit layer learning problem. Instead of applying the end-to-end DEQ method, we solve the ILT layer through ILILT, a weight-tied model, where a forward lithography estimator provides lithography information in the optimization procedure. Experiments demonstrate the effectiveness of ILILT learning an ILT layer with limited intervention of a traditional numerical solver. To the best of our knowledge, this is the first time an AI solution has been proposed to imitate the working scheme of a traditional ILT system. We hope this work can further motivate research into expanding the potential of AI for solving complex design automation problems. Additional investigations on production-level designs are necessary to prototype the framework to study whether QoR meets manufacturing standards.

References

Bai, S., Kolter, J. Z., and Koltun, V. Deep equilibrium models. In *Proc. NIPS*, 2019.

Banerjee, S., Li, Z., and Nassif, S. R. ICCAD-2013 CAD contest in mask optimization and benchmark suite. In *Proc. ICCAD*, pp. 271–274, 2013.

Braam, K., Selinidis, K., Hoppe, W., Cai, H., and Xiao,

Table 2. Comparison with State-of-the-Arts.

Design	AdaOPC (Zhao et al., 2022)			ILT-GPU (Gao et al., 2014)			DAMO (Chen et al., 2020)			Ours	
	EPE	PVB (nm^2)	Time (s)	EPE	PVB (nm^2)	Time (s)	EPE	PVB (nm^2)	Time (s)	EPE	PVB (nm^2)
1	22	23232	5.50	23	23329	41.15	22	23323	5.20	12	23446
2	24	26580	5.41	25	26762	48.50	26	26729	5.26	17	26565
3	24	26718	5.37	24	26720	55.92	27	26938	5.22	19	27003
4	25	27934	5.40	29	28127	70.57	36	27975	5.18	20	28388
5	30	28927	5.44	30	28925	66.89	35	28805	5.32	14	28951
6	24	26775	5.38	25	26762	55.81	30	26960	5.31	13	26959
7	28	26281	5.43	28	26453	59.47	33	26382	5.23	14	26641
8	27	29341	5.42	25	29450	54.88	32	30646	5.38	17	29463
9	23	24022	5.43	24	24053	70.62	25	24054	5.25	14	24210
10	22	21644	5.53	23	21701	37.59	24	21939	5.29	17	21902
Average	24.9	**26145.4**	5.43	25.6	26228.2	56.14	29	26375.1	5.26	**15.7**	26352.8
Ratio	1.000	**1.000**	1.000	1.028	1.003	10.337	1.165	1.009	0.969	**0.631**	1.008

Table 3. Generalization of ILILT.

Design	UNet			ILILT-UNet			CFNO			ILILT-CFNO	
	EPE	PVB (nm^2)	Time (s)	EPE	PVB (nm^2)	Time (s)	EPE	PVB (nm^2)	Time (s)	EPE	PVB (nm^2)
1	19	25281	0.16	11	23532	0.69	14	21895	0.23	12	23446
2	22	27627	0.15	25	26632	0.69	21	26899	0.22	17	26565
3	17	28963	0.16	21	26993	0.70	28	26459	0.21	19	27003
4	32	30130	0.16	32	28485	0.70	24	28804	0.23	20	28388
5	19	29293	0.15	15	28948	0.70	16	23983	0.23	14	28951
6	19	28953	0.14	19	27009	0.70	16	24123	0.21	13	26959
7	25	27445	0.14	25	26749	0.70	28	31110	0.21	14	26641
8	31	29483	0.14	21	29766	0.70	34	28565	0.21	17	29463
9	18	24360	0.17	16	24217	0.70	25	26829	0.23	14	24210
10	20	22272	0.15	20	21865	0.70	21	27003	0.22	17	21902
Average	22.2	27380.7	0.15	20.5	26419.6	0.70	22.7	26567	0.22	15.7	26352.8
Ratio	1.000	1.000	1.000	0.923	0.965	4.653	1.023	0.970	1.467	0.707	0.962

G. EUV mask synthesis with rigorous ILT for process window improvement. In *Proc. SPIE*, volume 10962, 2019.

Chen, G., Chen, W., Ma, Y., Yang, H., and Yu, B. DAMO: Deep agile mask optimization for full chip scale. In *Proc. ICCAD*, 2020.

Chen, T., Chen, X., Chen, W., Heaton, H., Liu, J., Wang, Z., and Yin, W. Learning to optimize: A primer and a benchmark. *arXiv preprint arXiv:2103.12828*, 2021.

Gao, J.-R., Xu, X., Yu, B., and Pan, D. Z. MOSAIC: Mask optimizing solution with process window aware inverse correction. In *Proc. DAC*, pp. 52:1–52:6, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proc. CVPR*, pp. 770–778, 2016.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Jiang, B., Liu, L., Ma, Y., Zhang, H., Young, E. F. Y., and Yu, B. Neural-ILT: Migrating ILT to nerual networks for mask printability and complexity co-optimizaton". In *Proc. ICCAD*, 2020.

Jittorntrum, K. An implicit function theorem. *Journal of Optimization Theory and Applications*, 25(4):575–577, 1978.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. In *Proc. ICLR*, 2021.

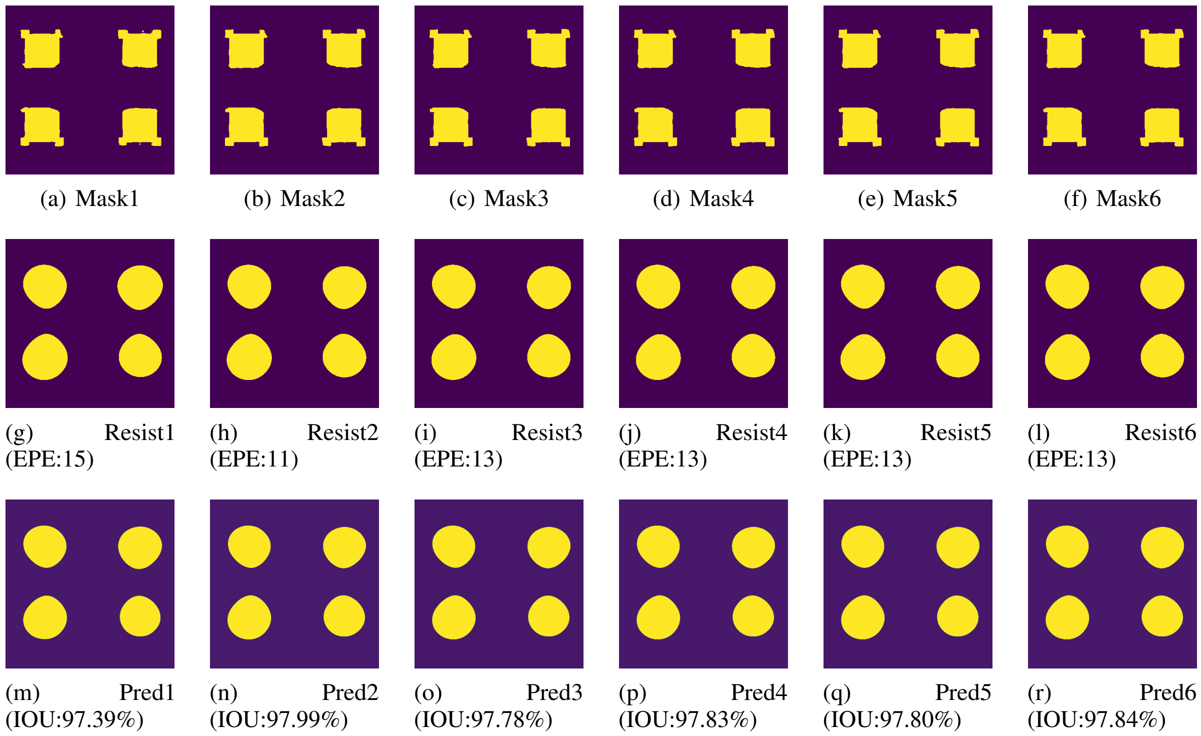


Figure 5. Visualization of the optimization trajectory beyond the maximum unrolling depth. Mask stays stable after the 4th step. (g)–(l) correspond to the simulated resist image of mask from time stamps 1–6 with EPE violation count. (m)–(r) are the lithography estimator g_l predicted resist image with the prediction error (IOU) compared to the simulated resist images.

Table 4. Effects of maximum unrolling depth.

Max Depth	2		4		6		8	
Design	EPE	PVB	EPE	PVB	EPE	PVB	EPE	PVB
1	15	23562	12	23446	13	23535	13	23634
2	25	26599	17	26565	19	26546	28	26562
3	22	26949	19	27003	21	26938	21	26839
4	30	28362	20	28388	22	28355	27	28373
5	20	28896	14	28951	21	28912	23	28882
6	21	26902	13	26959	15	26907	15	26946
7	22	26679	14	26641	17	26607	23	26612
8	23	30045	17	29463	20	29666	23	29754
9	15	24135	14	24210	13	24169	15	24206
10	20	21862	17	21902	21	21910	19	21940
Average	21.3	26399.1	15.7	26352.8	18.2	26354.5	20.7	26374.8
Ratio	1.000	1.000	0.737	0.998	0.854	0.998	0.972	0.999

Ma, Y., Gao, J.-R., Kuang, J., Miao, J., and Yu, B. A unified framework for simultaneous layout decomposition and mask optimization. In *Proc. ICCAD*, pp. 81–88, 2017.

Mishchenko, K. and Defazio, A. Prodigy: An expeditiously adaptive parameter-free learner. *arXiv preprint arXiv:2306.06101*, 2023.

Pang, L., Russell, E. V., Baggenstoss, B., Lee, M., Digaum, J., Yang, M.-C., Ungar, P. J., Bouaricha, A., Wang, K., Su, B., et al. Study of mask and wafer co-design that utilizes a new extreme simd approach to computing in memory manufacturing: full-chip curvilinear ilt in a day. In *Proc. SPIE*, volume 11148, pp. 136–151, 2019.

Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pp. 234–241, 2015.

Yang, H., Li, S., Ma, Y., Yu, B., and Young, E. F. GAN-OPC: Mask optimization with lithography-guided generative adversarial nets. In *Proc. DAC*, pp. 131:1–131:6, 2018.

Yang, H., Li, Z., Sastry, K., Mukhopadhyay, S., Anandkumar, A., Khailany, B., Singh, V., and Ren, H. Large scale mask optimization via convolutional fourier neural operator and litho-guided self training. *arXiv preprint arXiv:2207.04056*, 2022a.

Yang, H., Li, Z., Sastry, K., Mukhopadhyay, S., Kilgard, M., Anandkumar, A., Khailany, B., Singh, V., and Ren, H. Generic lithography modeling with dual-band optics-inspired neural networks. In *Proc. DAC*, 2022b.

Ye, W., Alawieh, M. B., Lin, Y., and Pan, D. Z. LithoGAN:

End-to-end lithography modeling with generative adversarial networks. In *Proc. DAC*, pp. 107:1–107:6, 2019.

Ye, W., Alawieh, M. B., Watanabe, Y., Nojima, S., Lin, Y., and Pan, D. Z. Tempo: Fast mask topography effect modeling with deep learning. In *Proc. ISPD*, pp. 127–134, 2020.

Yu, Z., Chen, G., Ma, Y., and Yu, B. A GPU-enabled level set method for mask optimization. In *Proc. DATE*, 2021.

Zhao, W., Yao, X., Yu, Z., Chen, G., Ma, Y., Yu, B., and Wong, M. D. AdaOPC: A self-adaptive mask optimization framework for real design patterns. In *Proc. ICCAD*, 2022.

Zheng, S., Yang, H., Zhu, B., Yu, B., and Wong, M. D. Lithobench: Benchmarking ai computational lithography for semiconductor manufacturing. In *Proc. NIPS*, 2023.

Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., and Liang, J. Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE transactions on medical imaging*, 39(6):1856–1867, 2019.