Moreau-ILT: Efficient Inverse Lithography via Exploiting Weak Convexity

Chaofan Wang, Ziyang Yu†, Su Zheng, Zixiao Wang, Yifan Shi, Bei Yu†
The Chinese University of Hong Kong

Abstract

As semiconductor manufacturers continue to pursue smaller critical dimensions, Inverse Lithography Technique (ILT) has emerged as a new standard in mask optimization. However, the high computational demand of ILT limits its applications in high-volume manufacturing, making *efficiency* a central topic of ILT research. To address this issue, we develop Moreau-ILT, a numerical ILT framework leveraging Moreau-envelope to decompose the weakly-convex differentiable ILT objective into a series of regularized sub-problems for accelerated convergence. Our method reduces ILT turnaround time by more than 3× compared with state-of-the-art numerical ILT solvers, without sacrificing mask quality, nor requiring data-driven warm start.

1 Introduction

Optical proximity correction (OPC) is a cornerstone of computational lithography in modern semiconductor manufacturing. As feature sizes continue to shrink in sub-wavelength domains, optical diffraction and process variation increasingly distort the transferred wafer image, making purely geometric mask design inadequate. Consequently, physics-aware mask synthesis through computational lithography remains essential for maintaining print fidelity and process window at advanced technology nodes. [1–3].

Among existing OPC paradigms, inverse lithography technique (ILT) is particularly attractive because it formulates mask synthesis as an inverse optimization problem under lithography imaging models, thereby exploring a substantially larger solution space than conventional rule-based or edge-based corrections [1]. As a result, ILT can achieve superior wafer fidelity and process window performance, which is especially important for aggressive patterning scenarios and curvilinear mask optimization [4–6]. Meanwhile, recent advances in curvilinear mask infrastructure and multi-beam mask writing are making such solutions increasingly practical for manufacturing deployment, further strengthening the case for ILT in high-volume production flows [7, 8].

ILT's superior mask optimization capability comes at the cost of massive computation. As a result, the ILT turnaround time, which often spans to the order of days for full-chip ILT, remains the main obstacle to its broader adoption [1–3, 7, 9]. Existing efforts to accelerate ILT mainly follow two directions. The first direction is data-driven acceleration, which learns either a direct design-to-mask mapping or a strong initialization for subsequent numerical refinement, as exemplified by GAN-OPC [10], Neural-ILT [11], DevelSet [12], and L2O-ILT [13]. The second direction is computational acceleration, which reduces the cost of each optimization step through GPU implementation, level-set parameterization [5], or multi-level / down-sampled lithography simulation [6, 14].

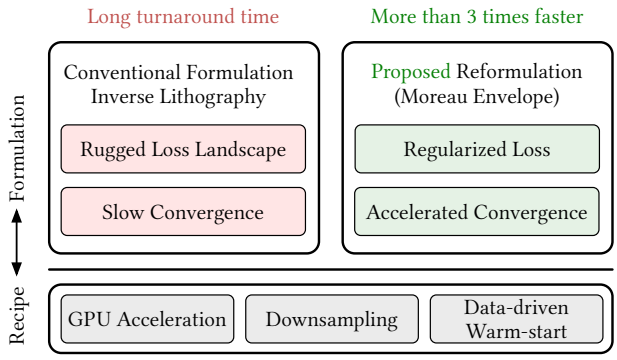


Figure 1: Illustration of efficient ILT approaches. Prior work improves the ILT recipe under conventional ILT formulations, whereas Moreau-ILT orthogonally reformulates the ILT to further boost ILT efficiency.

However, these two approaches, while demonstrating substantial improvement in the turnaround time of ILT, still leave an important gap. Data-driven methods typically require substantial offline training and large training corpora, and their effectiveness may degrade under training–inference distribution shift or under lithography conditions that are insufficiently represented during training. In practice, many of them still rely on a downstream numerical solver for final correction [3]. Numerical coarsening and down-sampling, on the other hand, can reduce the per-iteration computational cost, but they primarily make each step cheaper rather than reducing the optimization difficulty induced by the non-convex ILT loss landscape, and consequently remain limited by the heavy number of iterations before convergence.

This observation motivates a complementary question: *can ILT be accelerated by exploiting structural properties of the optimization objective itself, other than learning a better starting point or lowering the cost of each iteration*? Although differentiable ILT is intrinsically non-convex, the negative curvature introduced by the sigmoid mask and resist relaxations is uniformly bounded from below. This exposes a useful structure in the ILT loss landscape that has been largely overlooked in prior ILT acceleration methods. Rather than treating the ILT loss landscape as an arbitrary non-convex black box, we leverage its weak-convex structure to redesign the optimization process itself.

Building on this insight, we develop *Moreau-ILT*, a GPU-based numerical ILT framework that combines a Moreau-envelope-based reformulation with a stabilized 2nd-order inner solver. The reformulation converts the original ILT optimization into a sequence of regularized sub-problems while preserving the solution set of the

† Corresponding Authors
This work is supported by The Research Grants Council of Hong Kong SAR (No. RFS2425-4S02 and T46-415/25-R).

original problem. This reformulation redistributes the numerical workload in a favorable way: the outer Moreau objective, while still being non-convex, remains implicit and is updated in a single step with negligible cost, while the expensive iterative computation is concentrated on regularized inner sub-problems whose negative curvatures are suppressed and are therefore accelerable by efficient curvature-aware optimizers.

Leveraging this benefit, our framework demonstrates significant gains in terms of ILT turnaround time. On the ICCAD-2013 benchmark, Moreau-ILT achieves superior mask quality while reducing per-mask turnaround time by more than 3× over both state-of-the-art numerical ILT solvers and fast data-driven warm-start ILT solvers. Notably, our framework is in its essence complementary to existing ILT optimization approaches, including data-driven warm-start, and can be integrated into general differentiable ILT pipelines.

The main contributions of our work are:

- **Structural insight for ILT acceleration:** We identify weak convexity as a previously underexploited structural property of the differentiable ILT objective, and show that it can be leveraged through a Moreau-envelope-based reformulation for accelerated convergence.
- **Efficient ILT framework:** We develop Moreau-ILT, a GPU-based numerical ILT framework that combines the Moreau-envelope reformulation with a stabilized 2nd-order inner solver to enable efficient ILT optimization.
- **SOTA experimental results:** On the ICCAD-2013 benchmark [15], Moreau-ILT achieves better or matched mask quality than state-of-the-art numerical and data-driven warm-start ILT methods, while reducing turnaround time by more than 3×. The superiority robustly carries over to LithoBench[16], a larger mask optimization dataset.
- **A Flexible Toolbox.** Our Moreau-envelope reformulation is, in its essence, orthogonal to existing ILT efficiency-enhancing methods, and can be flexibly integrated into general ILT pipelines, supplementing the ILT design toolbox for future research.

The remainder of this paper is organized as follows. Section 2 provides background on differentiable inverse lithography. Section 3 presents the Moreau-ILT framework. Section 4 and Section 5 present the experimental results and conclusion, respectively.

2 Preliminaries

2.1 Forward and Inverse Lithography

Forward Lithography. In this work, we adopt the Hopkins' method [17, 18] as the projection optics model in our forward lithography pipeline. The Hopkins' method models partially coherent imaging using a truncated **s**um **o**f **c**oherent **s**ystems (SOCS). We use a minimal threshold-based resist model for our lithography simulation: a pixel is exposed if and only if its aerial intensity exceeds a threshold $I_{\text{th}} > 0$, and the exposed area is developed pixel-wise. The printed image is therefore:

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

We adopt the transmission cross coefficients (TCC) defining the SOCS, and the threshold settings from [15].

Inverse Lithography Technique. Given a desired printed image $\boldsymbol{Z}_T$, inverse lithography technique (ILT) seeks a binary mask $\boldsymbol{M}$ that reproduces $\boldsymbol{Z}_T$ under the forward lithography process [1]. In practice, ILT is formulated as an optimization problem over several mask quality metrics.

We use three standard metrics. The **L2 loss**, $L_2 = \|Z(\boldsymbol{M}) - \boldsymbol{Z}_T\|_2^2$, measures the area mismatch between the printed image and the target image. The **process variation band (PVB)**, $\|Z_{\max}(\boldsymbol{M}) - Z_{\min}(\boldsymbol{M})\|_2^2$, measures the difference between printed patterns at process corners. Following [15], we use ±2% dose variation, implemented by multiplying the aerial intensity by factors 1.02 and 0.98. The **edge placement error (EPE)** counts the number of sampled target-edge locations whose unsigned normal distance to the nominal printed contour exceeds a threshold d_{th}. We follow [15] and sample measurement points uniformly every 40 nm on the target contour, with $d_{\text{th}} = 15$ nm.

2.2 Differentiable Inverse Lithography

Modern ILT methods generally rely on gradient-based optimization [1, 4, 8, 15, 19], which requires a differentiable reformulation of the original problem.

Surrogate Objective. Although ILT can be evaluated by L2, PVB, and EPE, EPE is usually omitted during optimization because it requires contour tracing and is much more expensive to evaluate, while L2 already captures the resemblance between the nominal printed image and the target. Following prior work [4, 8, 11, 19–21], we therefore optimize the sum of L2 and PVB:

$$\mathcal{L}(\boldsymbol{M}) = \|Z(\boldsymbol{M}) - \boldsymbol{Z}_T\|_2^2 + \|Z_{\max}(\boldsymbol{M}) - Z_{\min}(\boldsymbol{M})\|_2^2, \tag{2}$$

where $Z(\boldsymbol{M})$, $Z_{\max}(\boldsymbol{M})$, and $Z_{\min}(\boldsymbol{M})$ are computed through the Hopkins' lithography model and (1).

Differentiable Resist and Mask. To enable gradient-based optimization, we replace the hard threshold resist model in (1) with a sigmoid surrogate:

$$Z = \frac{1}{1 + e^{-\alpha_r (\boldsymbol{I} - I_{\text{th}})}}, \tag{3}$$

where $\alpha_r > 0$ controls the stiffness of the resist surrogate; we use $\alpha = 50$ throughout this work.

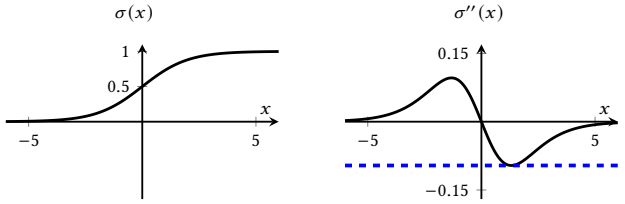
We also relax the binary mask by introducing a real-valued optimization variable $\boldsymbol{m} \in \mathbb{R}^{N \times N}$ and defining:

$$\boldsymbol{M} = \frac{1}{1 + e^{-\alpha_m (\boldsymbol{m} - T_M)}}, \ \mathcal{L}(\boldsymbol{m}) = \mathcal{L}\left(\frac{1}{1 + e^{-\alpha_m (\boldsymbol{m} - T_M)}}\right), \tag{4}$$

where $\alpha_m > 0$ and T_M are hyper-parameters. In this work, we fix $T_M = 0.5$ and sweep α_m.

3 Moreau-ILT Framework

In this section, we present the core idea and algorithmic flow of *Moreau-ILT*. First, we identify weak convexity as a useful structural property of the differentiable ILT objective. Second, based on this property, we derive a Moreau-envelope-based reformulation that is solution-wise equivalent to the original ILT problem while convexifying the inner optimization landscape. Third, we develop a practical algorithm that alternates between cheap outer synchronization steps and efficient inner solves, yielding a highly-efficient numerical ILT framework.



(a) the Sigmoid function (b) curvature of the Sigmoid function

Figure 2: Illustration for the bounded negative curvature of the Sigmoid Function. The blue dashed line is an indication to the eye, showing the sigmoid function's negative curvature is uniformly bounded from below.

3.1 Weak Convexity of ILT Loss

Weak Convexity of $\mathcal{L}(\boldsymbol{m})$. The differentiable reformulation in Equations (3) and (4) enables gradient-based optimization, but it also introduces non-convexity into the ILT loss landscape. As a result, the objective may contain local minima and/or negative curvatures, which can trap first-order methods and destabilize curvature-based solvers [22].

The key observation behind our framework is that this non-convexity is nevertheless *weak*: the negative curvature introduced by the sigmoid relaxations is uniformly bounded from below. See Figure 2 for an intuitive explanation.

Under our differentiable ILT formulation, this bounded negative curvature carries over to the loss $\mathcal{L}(\boldsymbol{m})$. A simple proof for this fact is that, since the sigmoid function saturates for large positive or negative $\boldsymbol{m}$, for an arbitrary $\epsilon > 0$, we can always find a compact subspace $\mathcal{C} = \{\boldsymbol{m} | -k \leq m_{ij} \leq k, \forall i, j \in [N]\}$ so that $\|\boldsymbol{H}(\mathcal{L}(\boldsymbol{m}))\| = \|\nabla^2 \mathcal{L}(\boldsymbol{m})\| < \epsilon$ for $\boldsymbol{m}$ outside $\mathcal{C}$. Meanwhile, the twice-continuously differentiable nature of the ILT loss $\mathcal{L}(\boldsymbol{m})$ ensures that the Hessian $\boldsymbol{H}(\mathcal{L}(\boldsymbol{m}))$ is a real symmetric matrix, and therefore its eigenvalues are real and finite. As a result, within the compact region $\mathcal{C}$, $\boldsymbol{H}(\mathcal{L}(\boldsymbol{m}))$'s eigenvalue is bounded from below. Putting it altogether, we know that $\mathcal{L}(\boldsymbol{m})$'s curvature is uniformly bounded from below for all $\boldsymbol{m} \in \mathbb{R}^{N \times N}$. This property is known as *weak convexity*.

Convexifying a Weakly Convex Objective. For a weakly convex function, adding a sufficiently strong quadratic term removes the negative curvature. i.e., there exists $p > 0$ such that

$$\mathcal{L}^*(\boldsymbol{m}) = \mathcal{L}(\boldsymbol{m}) + \frac{p}{2}\|\boldsymbol{m}\|_2^2 \tag{5}$$

is convex. A simple proof is that, since the eigenvalue of Hessian $\nabla^2 \mathcal{L}(\boldsymbol{m})$ is uniformly bounded from below, say by $\lambda_{\min}$, one can just choose $p > \max(0, -\lambda_{\min})$ so that $\nabla^2 \mathcal{L}^*(\boldsymbol{m}) = \nabla^2 \mathcal{L}(\boldsymbol{m}) + p\mathcal{I}_{N \times N}$ is positive-definite and $\mathcal{L}^*(\boldsymbol{m})$ is convex. This property is crucial for our accelerated ILT solver.

3.2 Reformulation using Moreau-Envelope

Moreau Envelope. To utilize (5) for ILT optimization, we adopt a mathemcatical tool, the Moreau envelope from proximal optimization [23]. The Moreau envelope of $\mathcal{L}(\boldsymbol{m})$ is defined as

$$g(\boldsymbol{u}) = \min_{\boldsymbol{m}} \left(\mathcal{L}(\boldsymbol{m}) + \frac{1}{2\lambda}\|\boldsymbol{u} - \boldsymbol{m}\|_2^2 \right), \tag{6}$$

where $\boldsymbol{u} \in \mathbb{R}^{N \times N}$ is an auxiliary variable and $\lambda > 0$ is a hyper-parameter. Classically, the Moreau envelope is used as a smoothing and regularization tool in optimization [24]. In our setting, it serves as a mechanism for *controlled convexification*: the quadratic term suppresses the weak non-convexity of the original ILT objective, at the expense of keeping each inner sub-problem solution close to the current reference point $\boldsymbol{u}$.

A crucial property of Moreau envelope is that the reformulated objective preserves the solution set of the original ILT problem:

$$\operatorname*{argmin}_{\boldsymbol{u}} g(\boldsymbol{u}) = \{\boldsymbol{u} = \boldsymbol{m}^* \mid \boldsymbol{m}^* \in \operatorname*{argmin}_{\boldsymbol{m}} \mathcal{L}(\boldsymbol{m})\}. \tag{7}$$

That is, minimizing $g(\boldsymbol{u})$ is solution-wise equivalent to minimizing $\mathcal{L}(\boldsymbol{m})$ [23]. This equivalence is important: it means the reformulation changes the optimization path, but not the target solution. At the same time, for suitable λ, the inner problem in Equation (6) becomes convexified and substantially more benign to optimizers than the original objective. In this way, we effectively hide $\mathcal{L}(\boldsymbol{m})$'s negative curvatures from the inner optimizer. This is an important leverage for designing a more efficient ILT solver.

Reformulation of ILT. Combining the discussions above, we propose the core of our Moreau-ILT framework: reformulation of the ILT objective. Instead of solving the original differentiable ILT problem in one shot, we repeatedly solve regularized sub-problems whose landscapes are shaped by a proximal term. Each regularized problem can be solved in a few iterations using an aggressive 2nd-order optimizer, leading to accelerated convergence of the original ILT problem. See Algorithm 1 and Figure 3.

We start from the original ILT formulation

$$\min_{\boldsymbol{m}} \mathcal{L}(\boldsymbol{m}) \tag{8}$$

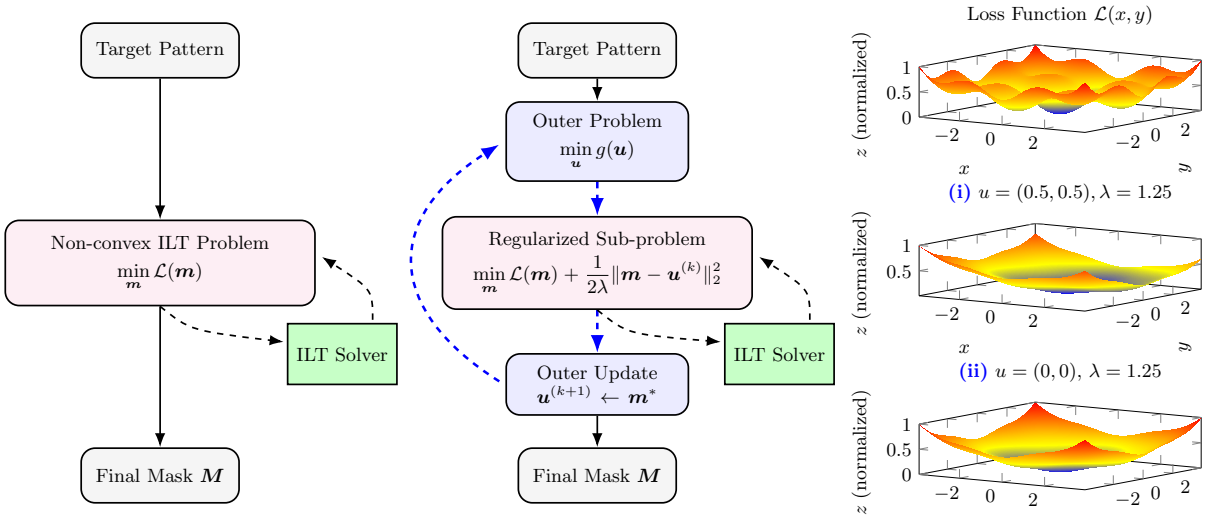
and reformulate it as the equivalent bi-level problem

$$\begin{aligned} &\min_{\boldsymbol{u}} g(\boldsymbol{u}) \\ &\text{where } g(\boldsymbol{u}) = \min_{\boldsymbol{m}} \left(\mathcal{L}(\boldsymbol{m}) + \frac{1}{2\lambda}\|\boldsymbol{u} - \boldsymbol{m}\|_2^2 \right). \end{aligned} \tag{9}$$

Here $\boldsymbol{u}$ is an auxiliary variable introduced by the Moreau reformulation, and $\lambda > 0$ is a hyper-parameter that controls the strength of the quadratic regularization. Conceptually, $\boldsymbol{u}$ serves as a moving reference point: at each stage of the optimization, we do not ask the solver to minimize the full non-convex objective globally from scratch. Instead, we ask it to improve the current solution under a proximal bias toward $\boldsymbol{u}$.

The bi-level form in Equation (9) can be solved efficiently using an alternated optimization scheme. At the k-th outer stage, we fix $\boldsymbol{u} = \boldsymbol{u}^{(k)}$ and perform several numerical optimization steps on the corresponding inner objective $f^{(k)}(\boldsymbol{m})$. This produces an updated mask variable $\boldsymbol{m}$ that approximately minimizes the current regularized sub-problem. We then fix $\boldsymbol{m}$ and update the outer variable $\boldsymbol{u}$ by minimizing $\mathcal{L}(\boldsymbol{m}) + \frac{1}{2\lambda}\|\boldsymbol{u} - \boldsymbol{m}\|_2^2$ with respect to $\boldsymbol{u}$. Crucially, this outer-level update has the closed-form solution $\boldsymbol{u}^{(k+1)} \leftarrow \boldsymbol{m}$, so the outer loop does not require any costly iterative optimization. i.e., the outer variable simply tracks the progress of the inner solver, and each new sub-problem is re-centered around the latest iterate.

The reformulated ILT workflow is summarized in Algorithm 1. We initialize the auxiliary variable from a uniform starting point and then alternate between inner updates on the regularized objective



(a) Conventional ILT (b) Moreau-ILT (c) Landscape Illustration

Figure 3: Comparison between conventional ILT and Moreau-ILT. (a) Conventional ILT directly optimizes the original non-convex loss $\mathcal{L}(\boldsymbol{m})$; (b) Moreau-ILT reformulates the problem into an outer loop over the auxiliary variable $\boldsymbol{u}$ and a sequence of regularized sub-problems in $\boldsymbol{m}$; (c) Example of optimizing a non-convex loss function using the Moreau reformulation. (i) The first sub-problem, regularized with $\lambda = 1.25$, $\boldsymbol{u}^{(1)} = (0.5, 0.5)$; (ii) The second sub-problem, regularized with $\lambda = 1.25$, $\boldsymbol{u}^{(2)} = (0, 0)$. The two steps produce the minimizer of $\mathcal{L}(x, y)$.

and outer synchronizations of $\boldsymbol{u}$. After the optimization finishes, the relaxed mask is up-sampled and projected back to the sigmoid mask representation. A conceptual comparison with conventional ILT is shown in Figure 3: conventional ILT directly optimizes the original non-convex loss, whereas Moreau-ILT decomposes the task into a sequence of regularized sub-problems whose objectives are more amenable to efficient numerical optimization.

Advantage of the Reformulation. The advantage of the reformulation lies in how it reduces the iterative optimizer's exposure to the destructive negative curvatures of the ILT objective. In conventional ILT, the optimizer directly confronts the raw loss landscape $\mathcal{L}(\boldsymbol{m})$, with all of its local irregularities and negative curvatures. In Moreau-ILT, by contrast, the heavy ILT optimization loop is redirected to a sequence of inner objectives of the form

$$f^{(k)}(\boldsymbol{m}) = \mathcal{L}(\boldsymbol{m}) + \frac{1}{2\lambda}\|\boldsymbol{u}^{(k)} - \boldsymbol{m}\|_2^2. \tag{10}$$

Each $f^{(k)}(\boldsymbol{m})$ augments the original ILT loss with a quadratic regularizer centered at the current outer iterate $\boldsymbol{u}^{(k)}$. This regularizer suppresses the negative curvature of the original ILT objective and imposes a proximal structure on the inner problem, at the expense of localizing $\boldsymbol{m}$ near $\boldsymbol{u}^{(k)}$. These sub-problems inherit the original ILT objective but with part of its negative curvature suppressed, making them much more amenable to fast curvature-aware optimization. As a result, Moreau-ILT gains efficiency not by directly solving the non-convex outer problem, but by ensuring that all substantial computation is spent on the better-structured inner problems, which could be accelerated using 2nd-order methods.

The reformulated problem brings three practical benefits. First, it is **solution-wise equivalent** to the original ILT objective, so the reformulation does not alter the target optimum. Second, it replaces the original objective with a sequence of **convexified sub-problems that converge within a few iterations**, accelerating the overall ILT flow despite having multiple outer updates. Third, our framework introduces only **minimal outer-loop overhead**, because the update of the auxiliary variable is a simple synchronization step. Combining these merits, our Moreau-ILT framework proves to be a highly-efficient ILT solver that generates high-quality masks at a fraction of runtime of state-of-the-art ILT solvers, as demonstrated by our experiments in Section 4.

Note on Choice of λ. In practice, while we would like to choose a sufficiently small $\lambda > 0$ so that $1/\lambda$ is strong enough to eliminate the non-convexity of the sub-problems, we need to balance between the convergence speeds of the inner and outer update. For the outer update, λ is the effective step size for updating $\boldsymbol{u}$ [23]. Therefore, for very small $\lambda > 0$, the parameter $\boldsymbol{u}$ is essentially trapped near its initial value, leading to sub-optimal solutions; while for very large λ, the benefit of convexification vanishes. As a result, in practice we sweep the parameter λ to find a sweet spot as in Figure 5. Later in Section 4.3 and Figure 5, we show that our Moreau-ILT is not pathologically sensitive to λ.

3.3 Inner ILT Solver

The effectiveness of Moreau-ILT depends not only on the reformulation itself, but also on how efficiently each inner sub-problem is solved. We therefore pair the Moreau reformulation with a practical inner solver designed for fast numerical convergence on the convexified objectives.

Down-sampling and Up-sampling. Following previous numerical ILT works [8, 19], we accelerate the inner optimization by working on a down-sampled representation of the target and mask variables.

Algorithm 1 Moreau-ILT

Input: Target image $Z_T \in \{0,1\}^{N\times N}$.
Output: Optimized mask $\boldsymbol{M} \in \{0,1\}^{N\times N}$

Down-sample Z_T by scale factor $s = 8$;
$\boldsymbol{m} \leftarrow Z_T$;
$k \leftarrow 1$;
$\boldsymbol{u}^{(1)} \leftarrow 0.5 \cdot \text{ones\_like}(Z_T)$; ▷ Uniform Starting Point
for $t = 1$ to #Steps **do**
  $f^{(k)}(\boldsymbol{m}) \leftarrow \mathcal{L}(\boldsymbol{m}) + \frac{1}{2\lambda}\|\boldsymbol{u}^{(k)} - \boldsymbol{m}\|_2^2$;
  Step $\boldsymbol{m}$ to minimize $f^{(k)}(\boldsymbol{m})$; ▷ Inner Update
  if $t \equiv 0 \pmod{\#\text{Inner\_Steps}}$ **then**
    $\boldsymbol{u}^{(k+1)} \leftarrow \boldsymbol{m}$, $k \leftarrow k+1$; ▷ Outer Update
  end if
end for
Up-sample $\boldsymbol{m}$ by scale factor $s = 8$;
$\boldsymbol{M} \leftarrow \frac{1}{1+\exp[-\alpha_m(\boldsymbol{m}-T_M)]}$;
return $\boldsymbol{M}$;

In particular, we follow [8] and use a scale factor of 8. After the full optimization process is completed, the resulting mask is up-sampled to the original resolution and evaluated with the full-scale forward lithography model.

Optimizer Backbone. A gradient-based optimizer is a central component of the inner ILT loop. In this work, we study two alternatives for the optimizer: (i) a 1st-order baseline, vanilla gradient descent; and (ii) a 2nd-order curvature-aware optimizer that follows previous arts AdaHessian [25] and Sophia [26].

For the 2nd-order solver, we adopt the randomized Hessian-estimation strategy in [25], which approximates the Hessian by its diagonal using interleaved Hessian-vector products. To improve numerical stability, we follow [26] and clamp near-zero curvature values before performing the update. The resulting solver is summarized in Algorithm 2, where $\gamma > 0$, $\epsilon > 0$ are hyperparameters controlling the clamping. Here we choose $\gamma = 10^{-2}, \epsilon = 10^{-4}$. Following [25], the Hessian is updated at the first step and then every 16 global steps (i.e., the t in Algorithm 1), so the additional cost of curvature estimation is minimal, only one extra backpropagation every 16 steps. See Algorithm 2. Note that in Algorithm 2 we explicitly include the #Inner_Steps loop for bookkeeping purpose, each inner update in Algorithm 1 only calls a single step of Algorithm 2.

1st-order v.s. 2nd-order Optimizer in Moreau-ILT. Note that while the regularizer introduced in our Moreau-ILT framework suppresses the negative curvatures of $\mathcal{L}(\boldsymbol{m})$, it does not automatically guarantee better conditioning in terms of the Hessian's condition number. If an eigenvalue of $\boldsymbol{H}(\mathcal{L})$ is close to $-1/\lambda$, then the corresponding curvature of $f^{(k)}(\boldsymbol{m})$ can remain close to zero, and the sub-problem may still be ill-conditioned. For this reason, a carefully chosen optimizer is crucial for leveraging the benefit of the regularized sub-problems in Moreau-ILT. In particular, when the inner solver is a first-order method such as gradient descent, the reformulation alone does not necessarily yield faster convergence.

In Section 4.3, we further compare the empirical behavior of the 1st-order and 2nd-order inner solvers, and isolate the contribution from the Moreau reformulation and the 2nd-order solver alone. Our results show that the main benefit of Moreau-ILT appears when

Algorithm 2 Inner ILT Solver

Input: $\boldsymbol{u}^{(k)}$; Down-sampled target $Z_T \in \{0,1\}^{\lfloor N/s\rfloor\times\lfloor N/s\rfloor}$
Output: $\boldsymbol{m}$ that minizes $\mathcal{L}(\boldsymbol{m}) + \frac{1}{2\lambda}\|\boldsymbol{u}^{(k)} - \boldsymbol{m}\|_2^2$

$\boldsymbol{m} \leftarrow \boldsymbol{m}_{\text{last}}$; ▷ Initialize with last $\boldsymbol{m}$ Value
$\boldsymbol{H} \leftarrow 0, \boldsymbol{G} \leftarrow 0$;
for $t = 1$ to #Inner_Steps **do**
  $f^{(k)}(\boldsymbol{m}) \leftarrow \mathcal{L}(\boldsymbol{m}) + \frac{1}{2\lambda}\|\boldsymbol{u}^{(k)} - \boldsymbol{m}\|_2^2$;
  if solver = 1st-order **then**; ▷ Gradient Descent
    $\boldsymbol{m} \leftarrow \boldsymbol{m} - \eta \frac{\partial f}{\partial \boldsymbol{m}}$;
  end if
  if solver = 2nd-order **then** ▷ Curvature-Aware Optimizer
    if Update_Hessian **then** ▷ Update $\text{diag}(\boldsymbol{H}(\mathcal{L}(\boldsymbol{m})))$
      $\boldsymbol{H}' \leftarrow$ Hutchinson Method Approx; [25]
      $\boldsymbol{H} \leftarrow \beta_H \boldsymbol{H} + (1-\beta_H)\boldsymbol{H}'$;
    end if
    $\boldsymbol{H}_{\text{clamp}} \leftarrow (\gamma \boldsymbol{H}).\text{clamp\_min}(\epsilon)$; ▷ Clamping [26]
    $\boldsymbol{G} \leftarrow \beta_G \boldsymbol{G} + (1-\beta_G)\frac{\partial f}{\partial \boldsymbol{m}}$;
    $\boldsymbol{m} \leftarrow \boldsymbol{m} - \eta \frac{\boldsymbol{G}}{\boldsymbol{H}_{\text{clamp}}}$; ▷ Element-wise Division
  end if
end for
return $\boldsymbol{m}$;

the regularized sub-problems are paired with the curvature-aware 2nd-order solver, which is aligned with our expectation.

4 Experimental Results

We develop the Moreau-ILT solver based on the open-source differentiable inverse lithography tool, OpenILT [29]. The OpenILT tool is based on PyTorch, and utilizes GPU for accelerated lithography simulation. The experiments are performed on a single NVidia RTX 3090 GPU, aligning with previous works [12, 13, 19] for a fair comparison of turnaround-times. The TCCs for the forward lithography simulation are obtained from the ICCAD-2013 competition [15]. In this Section, we use the 2nd-order version of Moreau-ILT's inner solver, because it provides that fastest convergence speed as discussed in Sections 3.3 and 4.3.

4.1 Evaluation on ICCAD-13 Benchmark

The ICCAD-13 Benchmark. The ICCAD-13 benchmark [15] consists of 10 clips of $2\,\mu\text{m} \times 2\,\mu\text{m}$ metal-layer patterns sampled from IBM's 32 nm process node. The clips are distributed in GLP format and rendered to binary matrices of size 2048×2048 to serve as the target pattern Z_T. This data set was originally released as part of the ICCAD 2013 contest and has been widely adopted by computational lithography researchers as a golden standard for evaluating mask optimization algorithms [4, 8, 11, 13, 14, 19, 21, 27]. In Table 1 and Table 2, we compare our method against state-of-the-art (SOTA) ILT algorithms under the ICCAD-13 benchmark. The "L2", "PVB", "EPE" columns are the three mask-quality metrics defined in Section 2.1, and the "TAT" column corresponds to the turnaround-time for generating each mask clip.

Comparison with SOTA Numerical ILT Solvers. Table 1 benchmarks our method against SOTA numerical ILT baselines "MultiILT" [19], "GLS-MO" [27], and "CurvyILT" [8]. Other numerical ILT algorithms, such as "MOSAIC" [4] and "GLS-ILT" [20], are not shown

Table 1: Comparison on ICCAD 2013 Benchmarks, Part I

Benchmarks		MultiILT-exact [19]				GLS-MO [27]†				CurvyILT-300 iterations [8]‡				**Ours-150 iterations**			
ID	Area (nm^2)	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)
case1	215344	38495	47015	3	3.45	36465	46558	2	3.83	38019	44410	3	2.24	36826	44585	3	0.72
case2	169280	28173	37555	0	3.44	28280	36783	0	2.84	28763	36579	0	2.76	28418	35296	0	0.66
case3	213504	67949	69361	22	3.44	61484	77304	15	5.60	63996	70683	15	2.13	62405	69916	14	0.70
case4	82560	10307	21514	0	3.45	8745	22087	0	2.55	8611	21455	0	2.97	8613	20921	0	0.69
case5	281958	28482	49683	0	3.46	29105	48486	0	2.91	27752	47533	0	2.48	28207	47213	0	0.69
case6	286234	30334	44127	0	3.42	29890	43561	0	2.57	30055	41983	0	2.78	29659	41582	0	0.68
case7	229149	14635	36961	0	3.46	12469	36039	0	2.73	12286	33445	0	2.31	13178	33457	0	0.67
case8	128544	11194	20985	0	3.42	12121	18159	0	3.71	11006	18405	0	2.83	10971	17800	0	0.67
case9	317581	34900	54948	0	3.47	35560	54395	0	3.17	32028	54491	0	2.81	32701	53977	0	0.66
case10	102400	7266	16581	0	3.47	7604	14691	0	3.72	6983	14951	0	2.28	6814	14720	0	0.65
Average		27173.5	39873.0	2.5	3.45	26172.3	39806.3	**1.7**	3.36	25949.9	38393.5	1.8	2.56	**25779.2**	**37946.7**	**1.7**	**0.68**
Rel. Value		1.054	1.051	+0.8	4.37	1.015	1.049	**+0**	4.25	1.007	1.012	+0.1	3.24	**1**	**1**	**+0**	**1**

† GLS-MO runs on CPU, the TAT here was adopted from [27], where the algorithm was tested on an Intel Core i7-14700 CPU.

‡ Experiment is re-performed under our RTX 3090 environment, using CurvyILT's released code in [28]. The solution quality is slightly better than ISPD'25 [8].

Table 2: Comparison on ICCAD 2013 Benchmarks, Part II

Benchmarks		DevelSet-TFPS [12]				DevelSet [21]				L2O-ILT [13]				**Ours-70 iterations**			
ID	Area (nm^2)	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)
case1	215344	48687	57612	8	1.91	49142	59607	10	1.50	39636	46905	3	1.12	38394	44604	3	0.33
case2	169280	34062	49647	1	1.83	34489	52012	1	1.40	29108	37099	0	1.11	29215	35851	0	0.31
case3	213504	92025	76744	59	1.76	93498	76558	64	1.29	67263	69115	21	1.13	64422	68215	16	0.28
case4	82560	18194	27318	2	2.21	18682	29047	2	1.65	10807	20694	1	1.12	10648	21056	0	0.22
case5	281958	42158	57123	1	1.34	44256	58085	1	0.91	31909	48797	0	1.10	30539	47732	0	0.27
case6	286234	41611	51096	2	1.37	41730	53410	2	0.84	31474	45453	0	1.14	31453	42366	0	0.23
case7	229149	24265	43032	0	1.52	25797	46606	0	0.76	16942	35942	0	1.11	15044	35772	0	0.30
case8	128544	15301	22131	0	1.63	15460	24836	0	1.14	12236	19496	0	1.13	12312	18460	0	0.22
case9	317581	49843	62139	0	1.77	50834	64950	0	1.21	34849	56706	0	1.11	35916	54215	0	0.26
case10	102400	10341	20104	0	1.00	10140	21619	0	0.42	7203	15976	0	1.11	7936	15237	0	0.24
Average		37648.7	46694.6	7.3	1.63	38402.8	48673.0	8.0	1.11	28142.7	39618.3	2.5	1.12	**27587.9**	**38350.8**	**1.9**	**0.28**
Rel. Value		1.365	1.217	+5.4	5.82	1.392	1.269	+6.1	3.96	1.020	1.033	+0.6	4.00	**1**	**1**	**+0**	**1**

Table 3: Comparison on LithoBench Test Set

Benchmark	MultiILT-exact [19] Reimp†				CurvyILT-300 iterations [8]‡				**Ours-150 iterations**			
Std-Metal	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)	L2 (nm^2)	PVB (nm^2)	EPE	TAT (s)
Average	13814.2	24928.2	0.03	3.45	12481.6	20846.5	**0.0**	2.51	**12436.4**	**20732.9**	**0.0**	**0.66**
Rel. Value	1.111	1.202	+0.03	5.23	1.004	1.005	**+0**	3.80	**1**	**1**	**+0**	**1**

† Reporting results of [8]'s re-implementation.

‡ Experiment is re-performed under our RTX 3090 environment, using CurvyILT's released code in [28].

in the table because they have always been dominated by the aforementioned three newer baselines. We choose the iteration budget of our solver so that the mask quality (L2, PVB, EPE) matches the best ones among the three baselines. In this specific case, we choose the iteration budget to be 150. Later in Figure 6, we show how our ILT loss evolves with iteration count.

Under this budget, our method generated masks with the same quality as "CurvyILT-300 iterations" (CurvyILT under a budget of 300 iterations [8]), at 3.24× faster TAT. Compared with the "GLS-MO" method [27], our solver generated masks with 4.9% better PVB, while maintaining a remarkable 4.25× TAT improvement. Furthermore, compared with the exact version of MultiILT ("MultiILT-exact") [19], our method achieves 4.37× TAT improvement while attaining 5.4%, 5.1%, and 32% improvement in L2, PVB, and EPE, respectively.

Overall, compared with SOTA numerical ILT methods, our method is able to generate masks with similar/better quality under a fraction

of runtime. Therefore, Moreau-ILT demonstrates substantial improvement over previous arts, and sets a new standard for numerical ILT solvers.

Comparison with Fast Data-Driven Warm-Start ILT Solvers. In order to highlight the efficiency of our Moreau-ILT method, we benchmark our method against a category of data-driven ILT algorithms. These algorithms use machine learning methods to generate an initial guess to warm-start a numerical ILT solver, potentially boosting convergence speed.

In Table 2, we benchmark Moreau-ILT against state-of-the-art data-driven warm-start ILT solvers, "DevelSet-TFPS"[12], "DevelSet"[21], and "L2O-ILT"[13]. These solvers are specifically designed to reduce the ILT turnaround time under the help of data-and-time-consuming pretraining stage, making them unfairly strong baselines against our numerical solver. Nonetheless, we are still able to outperform these solvers in terms of mask quality and TAT, showing the superiority of our method. In this part, we also choose the iteration budget of the Moreau-ILT solver so that our generated mask quality matches the best ones among the three baselines. In this specific case, we only need 70 iterations to match the mask qualities of the baselines.

Under this budget, Moreau-ILT is able to generate masks 4× faster than "L2O-ILT"[13], while maintaining a better mask quality (2.0% better L2, 3.3% better PVB, and 24.0% better EPE). Meanwhile, compared with "DevelSet"[21] and "DevelSet-TFPS"[12], our method achieves significantly better mask quality (39.2% / 36.5% better L2, 26.9% / 21.7% better PVB, and 76.3% / 74.0% better EPE, respectively) at a favorable 3.96× / 5.82× turnaround time.

Overall, despite the unfair comparison favoring data-driven warm-start ILT methods, Moreau-ILT still generates masks with better quality within a fraction of their runtime. This highlights the high efficiency of our method. Notably, our method can be complementary to the data-driven warm-start methods. On one hand, our method is suitable for generating large datasets due to its superior mask quality and short runtime; on the other, general warm-start ILT method can be used to generate a initial solution for Moreau-ILT, potentially further boosting ILT efficiency.

Summary. Moreau-ILT's per-mask turnaround time is reduced by > 3.25× / > 3.96× compared with state-of-the-art numerical / data-driven warm-start ILT solvers at matched or better mask qualities. This demonstrates the superiority of our method, setting a new standard for efficient ILT algorithms. We visualize an optimized mask for ICCAD benchmark case 1 in Figure 4, the mask was optimized using Moreau-ILT with 150 iterations.

4.2 Evaluation on the LithoBench Dataset

To highlight the robustness of our proposed approach, we compare our method against SOTA numerical ILT methods on a newer and larger dataset, LithoBench [16]. LithoBench distributes 271 tiles of Nangate 45 nm metal layer patterns in the same form as the ICCAD-13 benchmark [15], therefore consisting of a larger-size alternative to the ICCAD-13 benchmark.

In Table 3, we benchmark Moreau-ILT against two state-of-the-art benchmarks "MultiILT-exact" and "CurvyILT-300 iterations". We did not compare against GLS-MO [27] on LithoBench, because the source code / binary of GLS-MO is not publicly available. Without adjusting any hyperparameters, Moreau-ILT achieves the best mask quality and fastest runtime among the three methods. Compared with "CurvyILT-300 iterations", our method achieves better mask qualities within 150 iterations, and finishes in 3.80× shorter wall clock time. Moreover, comparing against "MultiILT-exact", our method achieves 11.1% / 20.2% improved L2 / PVB with 5.23× faster turnaround time. The exellence on LithoBench further solidifies the superiority of Moreau-ILT as a highly efficient ILT solver.

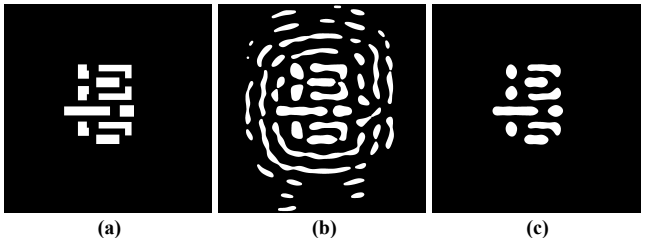


Figure 4: Visualization of Moreau-ILT-generated mask for ICCAD benchmark case 1. (a) The target design of ICCAD case 1; (b) The generated mask using Moreau-ILT-150 iterations; (c) The printed image at nominal dose.

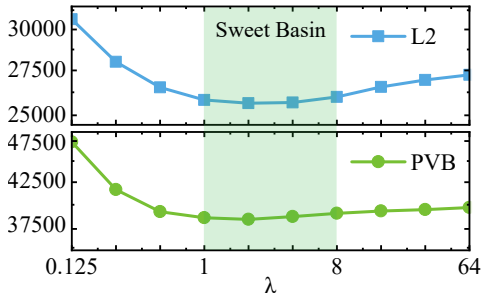


Figure 5: Sensitivity of mask quality against λ on ICCAD-13 benchmark. The L2, PVB values are in nm^2.

Table 4: Hyperparameter settings for Figure 6

Solver \ Param.	η	α_m	β_G	β_H	λ
1st-order	1.0	4.0	-	-	-
2nd-order	0.05	6.0	0.9	0.9	-
1st-order + Moreau	0.3	8.0	-	-	2.5
2nd-order + Moreau	0.03	10.0	0.9	0.9	2.5

4.3 Ablation Study

λ-Sensitivity Analysis. The hyperparameter $\lambda > 0$ is the critical parameter for our Moreau-ILT framework. It sets the strength of the Moreau regularization term and determines the tradeoff between the convergence of the sub-problems (favoring small λ) and the effective step size of the outer update (favoring large λ's). In Figure 5, we plot the optimized mask quality with respect to λ between $\frac{1}{8}$ to 64, $\eta > 0$ and $\alpha_m > 0$ are tuned to achieve best mask quality for each λ individually. For the ICCAD-13 benchmark, the sweet range of λ spans almost an order of magnitude, showing that Moreau-ILT is a robust method that does not pathologically depend on the choice of hyperparameter λ.

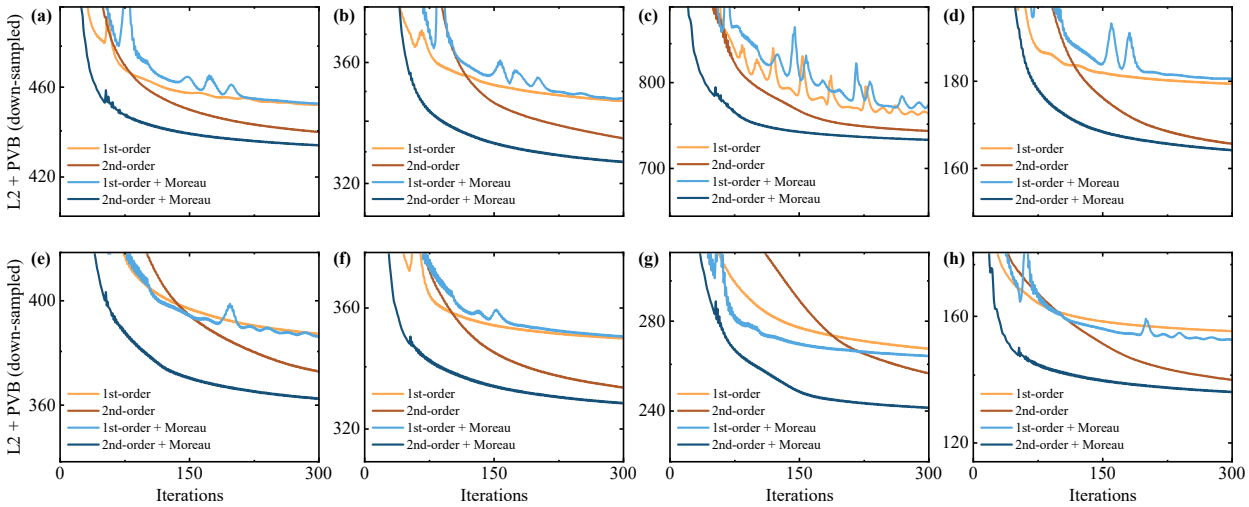


Figure 6: Traces of the down-sampled ILT loss $\mathcal{L}(\boldsymbol{m})$ during the Moreau-ILT optimization process for the first eight ICCAD-13 benchmarks. (a-h) corresponds to ICCAD-13 benchmark case 1 to case 8. Yellow/brown trace: the inner ILT solver as a standalone ILT solver, with 1st-/2nd-order optimizer as its backbone; Light blue/dark blue: Moreau-ILT with 1st-/2nd-order optimizers. Here the horizontal axis "Iterations" refers to the number of inner steps, i.e., the number of times where cycles of ILT forward + backward are performed, which roughly reflects the wall-clock time spent on the mask optimization.

Isolating the Contribution of the 2nd-Order Optimizer. In Tables 1 and 2, we demonstrated that Moreau-ILT with 2nd-order-optimizer based inner ILT solver is able to generate masks with matched or even better quality in a fraction of runtime, compared with SOTA methods. In this part, we perform an ablation study to isolate the contribution from the Moreau-reformulation and the contribution from the 2nd-order inner ILT solver.

We plot the traces of our loss function $\mathcal{L}(\boldsymbol{m})$ for the first eight ICCAD-13 benchmark cases in Figure 6. The yellow / brown lines show the trace obtained using *just* the 1st-/2nd-order inner ILT solver, without the Moreau reformulation. The light blue / dark blue traces are obtained using our Moreau-ILT solver, with 1st-/2nd-order inner ILT solver. We use grid search to determine the best hyperparameters for each solver. The best hyperparameters we use for Figure 6 is summarized in Table 4. For 1st-order + Moreau solver, we use $\lambda = 2.5$, #Inner_Steps= 10; and for 2nd-order + Moreau solver we use $\lambda = 2.5$, #Inner_Steps= 5.

Contribution from the 2nd-Order Solver: In Figure 6, the traces obtained when inner ILT solver is 2nd-order (brown / dark blue lines) consistently converge to lower losses compared with vanilla gradient descent-based solvers (blue / yellow lines). This indicates that the 2nd-order solver generally achieves better quality masks on the ICCAD-13 benchmark. However, when not combined with the Moreau reformulation, the 2nd-order solver doesn't demonstrate faster convergence compared with 1st-order methods. In Figure 6 (b) (d) (g), without the Moreau reformulation, the 2nd-order optimizer-based solver's convergence curve (brown trace) intersects with the one obtained with the 1st-order solver (yellow trace), showing that they converge to same mask quality using roughly the same number of iterations. Therefore, the 2nd-order solver alone doesn't provide much benefit in terms of turnaround time.

Contribution from the Moreau Reformulation: When using 1st-order optimizer + Moreau reformulation (the light blue traces), there is no significant improvement compared with the vanilla 1st-order optimizer-based solver (the yellow traces). This aligns with our expectation in Section 3.3.

In Figure 6, the traces obtained using 2nd-order optimizer + Moreau reformulation (the dark blue lines) consistently outperform the other three settings, simultaneously achieving lower loss and faster convergence. This proves that our Moreau reformulation significant boosts the convergence of 2nd-order methods, and therefore improving turnaround time. This is also expected because the convergence of the curvature-aware optimizer strongly depends on *not* having negative curvatures, aligning with the motivation of our Moreau-envelope reformulation.

In conclusion, our ablation study shows that, 2nd-order optimizers alone does not demonstrate significant boost in the ILT efficiency. However, it is a key component in our Moreau-ILT formulation, because it utilizes the regularized loss landscape after the Moreau reformulation to accelerated the ILT optimization in a curvature-aware fashion. Our Moreau-ILT formalism, when backed up by a 2nd-order optimizer, shows significant improvement in both mask quality and turnaround time compared with other alternatives.

5 Conclusion

In this work, we present Moreau-ILT, a highly efficient numerical inverse lithography framework that generates high-quality photomasks in a fraction of turnaround time. Our optimization pipeline effectively hides the ILT objective's finite negative curvatures from gradient-based optimizers, making the ILT problem accelerable using a curvature-aware 2nd-order optimizer. Experimentally, our algorithm demonstrates more than 3× runtime improvement compared to state-of-the-art numerical ILT algorithms. Furthermore, our Moreau-envelope reformulation is complementary to previous ILT acceleration techniques, and we expect the introduction of this technique to stimulate future trends in numerical inverse lithography research.

References

[1] L. Pang, "Inverse lithography technology: 30 years from concept to practical, full-chip reality," *Journal of Micro/Nanopatterning, Materials, and Metrology*, vol. 20, no. 3, pp. 030 901–030 901, 2021.

[2] T. Cecil, D. Peng, D. Abrams, S. J. Osher, and E. Yablonovitch, "Advances in inverse lithography," *ACS Photonics*, vol. 10, no. 4, pp. 910–918, 2022.

[3] Y. Yang, K. Liu, Y. Gao, C. Wang, and L. Cao, "Advancements and challenges in inverse lithography technology: a review of artificial intelligence-based approaches," *Light: Science & Applications*, vol. 14, no. 1, p. 250, 2025.

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

[5] Z. Yu, G. Chen, Y. Ma, and B. Yu, "A GPU-enabled level-set method for mask optimization," *IEEE TCAD*, vol. 42, no. 2, pp. 594–605, 2022.

[6] S. Sun, F. Yang, B. Yu, L. Shang, and X. Zeng, "Adaptive ilt via multi-level lithography simulation," *IEEE TCAD*, 2025.

[7] L. Pang, E. V. Russell, B. Baggenstoss, M. Lee, J. Digaum, M.-C. Yang, P. J. Ungar, A. Bouaricha, K. Wang, B. Su *et al.*, "Breakthrough curvilinear ilt enabled by multi-beam mask writing," *Journal of Micro/Nanopatterning, Materials, and Metrology*, vol. 20, no. 4, pp. 041 405–041 405, 2021.

[8] H. Yang and H. Ren, "Gpu-accelerated inverse lithography towards high quality curvy mask generation," in *Proc. ISPD*, 2025, pp. 42–50.

[9] L. L. Pang, P. J. Ungar, A. Bouaricha, L. Sha, M. Pomerantsev, M. Niewczas, K. Wang, B. Su, R. Pearman, and A. Fujimura, "Truemask ilt mwco: full-chip curvilinear ilt in a day and full mask multi-beam and vsb writing in 12 hrs for 193i," in *Optical Microlithography XXXIII*, vol. 11327. SPIE, 2020, pp. 145–158.

[10] H. Yang, S. Li, Z. Deng, Y. Ma, B. Yu, and E. F. Y. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," *IEEE TCAD*, 2020.

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

[12] G. Chen, Z. Yu, H. Liu, Y. Ma, and B. Yu, "Develset: Deep neural level set for instant mask optimization," *IEEE TCAD*, vol. 42, no. 12, pp. 5020–5033, 2023.

[13] B. Zhu, S. Zheng, Z. Yu, G. Chen, Y. Ma, F. Yang, B. Yu, and M. D. Wong, "L2o-ilt: Learning to optimize inverse lithography techniques," *IEEE TCAD*, vol. 43, no. 3, pp. 944–955, 2023.

[14] Q. Wang, B. Jiang, M. D. Wong, and E. F. Young, "A2-ilt: Gpu accelerated ilt with spatial attention mechanism," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, 2022, pp. 967–972.

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

[16] S. Zheng, H. Yang, B. Zhu, B. Yu, and M. Wong, "Lithobench: Benchmarking ai computational lithography for semiconductor manufacturing," *Proc. NIPS*, vol. 36, pp. 30 243–30 254, 2023.

[17] H. Hopkins, "The concept of partial coherence in optics," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 208, no. 1093. The Royal Society, 1951, pp. 263–277.

[18] N. Cobb, "Sum of coherent systems decomposition by svd," *Berkeley CA*, vol. 94720, pp. 1–7, 1995.

[19] S. Sun, F. Yang, B. Yu, L. Shang, and X. Zeng, "Efficient ilt via multi-level lithography simulation," in *Proc. DAC*, 2023, pp. 1–6.

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

[21] G. Chen, Z. Yu, H. Liu, Y. Ma, and B. Yu, "Develset: Deep neural level set for instant mask optimization," in *Proc. ICCAD*, 2021, pp. 1–9.

[22] J. Nocedal and S. J. Wright, *Numerical optimization*. Springer, 2006.

[23] J.-J. Moreau, "Proximité et dualité dans un espace hilbertien," *Bulletin de la Société mathématique de France*, vol. 93, pp. 273–299, 1965.

[24] P. Liao, H. Liu, Y. Lin, B. Yu, and M. Wong, "On a moreau envelope wirelength model for analytical global placement," in *Proc. DAC*, 2023, pp. 1–6.

[25] Z. Yao, A. Gholami, S. Shen, M. Mustafa, K. Keutzer, and M. Mahoney, "Adahessian: An adaptive second order optimizer for machine learning," in *Proc. AAAI*, vol. 35, no. 12, 2021, pp. 10 665–10 673.

[26] H. Liu, Z. Li, D. L. W. Hall, P. Liang, and T. Ma, "Sophia: A scalable stochastic second-order optimizer for language model pre-training," *Proc. ICML*, 2024.

[27] N. Nonaka, M. Kuramochi, Y. Kohira, R. Azuma, T. Matsui, A. Takahashi, and C. Kodama, "Fast mask optimization under process variation using guided local search on quadratic programming," *IEEE TCAD*, 2025.

[28] H. Yang and H. Ren, "Nvlabs/curvyilt," https://github.com/NVlabs/curvyILT, 2025.

[29] S. Zheng, Y. Ma, B. Zhu, G. Chen, W. Zhao, S. Yin, Z. Yu, and B. Yu, "Openilt: An open-source platform for inverse lithography technique research," https://github.com/OpenOPC/OpenILT/, 2023.