RL-OPC: Mask Optimization With Deep Reinforcement Learning

Xiaoxiao Liang, Yikang Ouyang, Haoyu Yang, Bei Yu, *Senior Member, IEEE*, and Yuzhe Ma, *Member, IEEE*

***Abstract*—Mask optimization is a vital step in the VLSI manufacturing process in advanced technology nodes. As one of the most representative techniques, optical proximity correction (OPC) is widely applied to enhance printability. Since conventional OPC methods consume prohibitive computational overhead, recent research has applied machine learning techniques for efficient mask optimization. However, existing discriminative learning models rely on a given dataset for supervised training, and generative learning models usually leverage a proxy optimization objective for end-to-end learning, which may limit the feasibility. In this article, we pioneer introducing the reinforcement learning (RL) model for mask optimization, which directly optimizes the preferred objective without leveraging a differentiable proxy. Intensive experiments show that our method outperforms state-of-the-art solutions, including academic approaches and commercial toolkits.**

***Index Terms*—Design for manufacturing, mask optimization, reinforcement learning (RL), optical proximity correction (OPC).**

I. Introduction

THE CONTINUOUS shrinking of geometries in the modern VLSI has become a great challenge for manufacturing. Therefore, various resolution enhancement techniques (RETs) have been developed. As one of the most representative RETs, optical proximity correction (OPC) is vital to improving printability and attracts great attention from both academia and industry.

OPC aims to rectify the pattern to compensate for distortions resulting from the diffraction effect of the light during lithography. Fig. 1 depicts the effectiveness of OPC on a simple via pattern. It can be observed in Fig. 1(a) that directly applying the target pattern as the mask pattern, i.e., without conducting OPC, leads to a large distortion on the printed contour. Fig. 1(b) shows OPC can significantly improve the fidelity of the printed contour. Existing OPC solutions can be classified into several categories, including rule-based OPC [1], model-based OPC [2], [3], [4], inverse lithography technique (ILT) [5], [6], and machine learning (ML)-based OPC [7], [8], [9], [10], [11]. Rule-based OPC employs empirical correction rules as correction guidance. However, its efficiency and dependency on prior knowledge and empirical data are challenged by the extreme scaling of modern VLSI design. Conventional model-based OPC refines the mask iteratively based on lithography simulation. The edges of a pattern are fractured into segments that are moved inward or outward. Kuang et al. [2] proposed to optimize the edge placement error (EPE) and mask robustness through multistage optimization. Su et al. [3] built a fast OPC flow that adaptively optimizes the tradeoff between EPE and process variations.

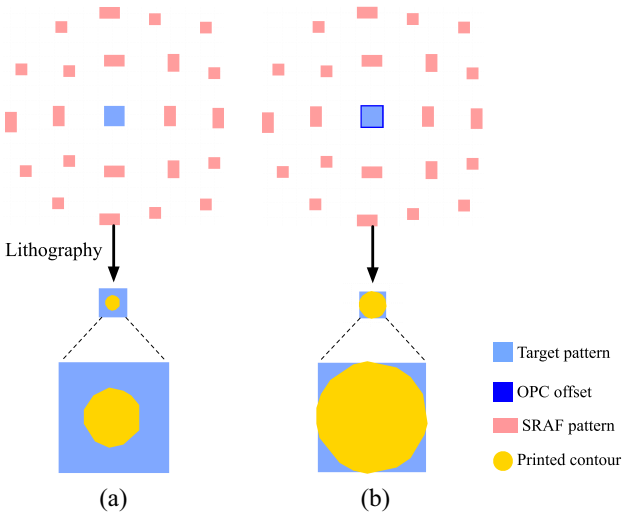

Fig. 1. (a) Directly applying target pattern as mask pattern leads to large distortion on the printed contour. (b) OPC significantly improves the fidelity of the printed contour.

To further increase the solution space, ILT-based methods treat mask optimization as an inverse imaging problem, which conducts numerical optimization on the pixels of a mask image. Conventional ILT approaches [5], [6] consider the design target and process window simultaneously to achieve high printability. However, the scalability of ILT is a major concern due to the intensive computation in the forward lithography simulation and backward gradient computation, which constrains ILT to be applied only to a portion of design patterns. Hence, model-based OPC and ILT are complementary in main-stream OPC solutions.

Manuscript received 20 January 2023; revised 21 May 2023 and 30 July 2023; accepted 4 August 2023. Date of publication 29 August 2023; date of current version 26 December 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62204066; in part by the Guangzhou-HKUST (GZ) Joint Funding Program under Grant 2023A03J0013; and in part by the Research Grants Council of Hong Kong, SAR, under Project CUHK14208021. This article was recommended by Associate Editor I. H.-R. Jiang. *(Corresponding author: Yuzhe Ma.)*

Xiaoxiao Liang, Yikang Ouyang, and Yuzhe Ma are with the Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511458, China (e-mail: yuzhema@hkust-gz.edu.cn).

Haoyu Yang is with the Design Automation Research Group, Nvidia, Austin, TX 78717 USA.

Bei Yu is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, SAR.

Digital Object Identifier 10.1109/TCAD.2023.3309745

The advancement of ML techniques has catalyzed the research of exploring efficient and intelligent solutions in VLSI design and manufacturing [12], including mask optimization. Existing learning-based OPC methods mainly involve discriminative models and generative models. Jiang et al. [7] constructed an OPC acceleration framework and a mask printability prediction model, which predicts EPE during the OPC workflow and guides the edge movements. Generative models [9], [10] mainly leverage generative adversarial networks (GANs) to generate the desired mask for a given target pattern, which may also serve as a superior initialization for conventional OPC engines. Chen et al. [10] developed DAMO to handle high-resolution input. In general, learning-based methods are dependent on a labeled dataset and are trained in a supervised manner based on a loss function. However, several issues exist in these two aspects. Regarding the dataset, since the ML models are trained on a dataset that contains design patterns and corresponding masks from a specific OPC engine (mostly a commercial tool), their performance is constrained by the dataset quality. Since the principle of supervised learning is to *approximate* a function that perfectly fits the dataset, although the ML model is trained perfectly, the OPC results are still not expected to reliably surpass the results in the original dataset. Regarding the loss function, learning-based methods rely on a differentiable objective to perform gradient descent. Considering that the primary metrics, e.g., EPE and process variation band (PV band), are calculated in a discrete manner and cannot be directly applied in gradient propagation, learning-based methods leverage proxy objectives for approximation, including intersection over union [10] and squared L_2 error [9]. However, guidance from the aforementioned proxy objectives aims to minimize the pixel-wise difference between the generated mask (or contour) and the reference mask (or contour), which does not align with the primary metrics perfectly and hence leads to a quality gap of the final OPC results.

There are multiple objectives to be considered in mask optimization tasks, including wafer image quality and mask robustness against process variation. To optimize the two objectives, existing model-based OPC approaches generally split the optimization procedure into different subroutines for different evaluation metrics. Thus, the optimized result of the former stage is easily perturbed by the latter modifications. For instance, [2] involves two successive stages of EPE and PV band minimization. However, the solution space of the latter stage, i.e., PV band optimization, is limited because of potential new EPE violations (EPEVs), with which another round of EPE refixing should be launched.

On the other hand, ILT-based methods generally perform multiobjective optimization by integrating the evaluation metrics into the objective function. For instance, the cost function of [6] is designed as the weighted summation of EPE and PV band. Standard PV band measurement requires multiple lithography simulations to obtain printed images under all process conditions and involves Boolean operations like XOR among all printed images. However, this is not applicable to ILT since ILT requires a continuous objective function. Alternatively, conventional ILT methods [6] measure PV band by the sum of the image difference between different printed images and the target image to approximate the PV band, which may result in PV band inaccuracy. Meanwhile, since ILT suffers from intensive computational overhead, for efficiency improvement, the PV band is measured among simulation results under a small set of representative process corners. Hence, the PV band approximation and the pruned process corners both lead to potential inaccuracy for PV band measurements, and thus ruin the PV band optimization in ILT.

In order to address these issues in mask optimization for modern VLSI design and manufacturing, we propose reinforcement learning (RL)-OPC, a mask optimization framework based on RL, for simultaneous optimization of mask quality and robustness. RL-OPC aims to train an agent which interacts with and learns from the environment and performs automatic optimization for mask correction based on Q-learning [13]. For each movable edge of the target pattern, the agent extracts the local pattern feature and determines its moving direction, and then receives a reward from the environment by performing lithography simulation to justify the effectiveness of this move and learns from these experiences, i.e., the Q-value of each action for a specific local pattern. Once the agent is trained, it can determine how to correct the mask by sequentially moving all the edges of an input pattern. RL-OPC has several advantages over discriminative OPC models and generative OPC models. On the one hand, the proposed RL-OPC does not rely on any dataset as a reference, which can potentially achieve superior qualities on OPC results. On the other hand, RL-OPC can be trained without introducing a differentiable proxy metric in OPC but can directly optimize discrete OPC objectives. Compared with the aforementioned model-based methods, RL-OPC simultaneously minimizes EPE and PV band by building a multiobjective RL framework. Compared with ILT, RL-OPC involves more precise PV band measurement which utilizes the standard measuring strategy and enumerates more intensive process corners. Moreover, it should be noted that RL-OPC is a good complement to all existing learning-based OPC methods. Using the generated mask of existing learning-based methods as the initialization of RL-OPC leads to substantial improvement in mask quality and runtime. We conduct intensive experiments to validate the effectiveness of RL-OPC and demonstrate its superiority in the quality of the generated masks. Overall, our main contributions can be summarized as follows.

1) We propose an RL-based OPC framework that tackles mask optimization in modern VLSI design and manufacturing.
2) A deep Q-learning approach is developed to directly optimize the nondifferentiable objectives, i.e., EPE and PV band, which is more effective than conventional generative learning models that utilize a proxy differentiable objective such as L_2 loss for mask optimization.
3) A multiobjective RL framework is built to simultaneously minimize EPE and PV band by branching the network for two individual and parallel tasks and aggregating their Q-values to generate the optimal decision.
4) We demonstrate and validate that RL-OPC can be further enhanced by using a pretrained OPC learning

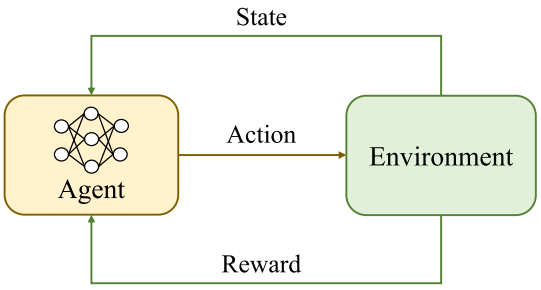

Fig. 2. General RL framework.

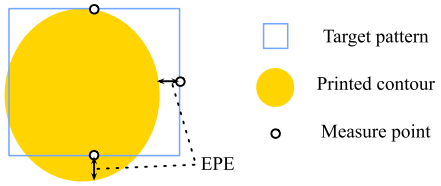

Fig. 3. Example of EPE measurement.

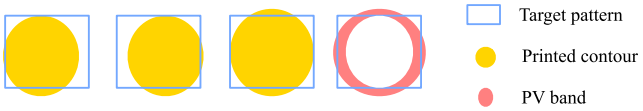

Fig. 4. Example of PV band measurement.

model to generate an initial mask as the input to boost efficiency.

5) Experimental results demonstrate that RL-OPC outperforms state-of-the-art learning-based OPC methods and commercial tools with more than 61% and 24% improvement in reducing the contour distortion, respectively, along with mask robustness enhancement by 2.9% and 2.4%. Moreover, we also demonstrate the potential of integrating existing OPC learning models with RL-OPC, achieving 40% mask quality improvement and 1.58× speedup.

The remainder of this article is organized as follows. Section II discusses the preliminaries, including a brief illustration of Q-learning and evaluation metrics, followed by the problem formulation. Our proposed algorithm will be explained in Section III. Section IV demonstrates our implementation details, experimental results, and quantitative comparisons, followed by a conclusion in Section V.

II. Preliminaries

A. Reinforcement Learning and Q-Learning

RL investigates how intelligent agents attempt to optimize a goal function by interacting with its environment and thus making sequential decisions. RL involves a set of optimization instances named *state* S, and a set of *actions* A per state. The agent transitions from state s to state s' by performing an action $a \in A$, and receives a *reward* $r(s, a)$ from RL environment as evaluation, as demonstrated in Fig. 2. The action selection model is called *policy* π, and the RL agent aims to learn a policy that maximizes accumulative reward.

Q-learning is an RL algorithm that learns the scores of each action a corresponding to the given state s, and the score is called Q-value, which is denoted as $Q(s, a)$.

The mathematical formulation of $Q(s, a)$ is as follows:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t r_t | s_0 = s, a_0 = a\right] = r(s,a) + \gamma \sum_{a_i \in A} \pi\left(a_i | s'\right) * Q^{\pi}\left(s', a_i\right) \quad (1)$$

where $\pi(a|s)$ refers to the probability of action a is decided by policy π with input s, and γ is the discount factor. For clarity, we omit the policy notation π in the following sections. Since it is impossible to enumerate all possible trajectories for the accurate accumulated reward expectation, in practice, the Q-value expectation is generally approximated as follows [14]:

$$Q(s,a) = r(s,a) + \gamma \max_{a'} Q\left(s', a'\right) \quad (2)$$

where s' indicates the next state. Therefore, the Q-value is updated by

$$Q(s,a) = Q(s,a) + \alpha\left[r(s,a) + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a)\right] \quad (3)$$

where α is the learning rate. In this article, we adopt the deep Q-learning method, which employs a deep neural network as the Q-value approximator [15]. Upon receiving state s as input, the network outputs the estimated Q-value $Q(s, a)$ w.r.t each possible action a.

B. Evaluation Metrics

Before giving the problem definition, we first introduce the evaluation metrics in the OPC problem.

Definition 1 (EPE): EPE refers to the vertical or horizontal misalignment, i.e., Manhattan distance from the lithography contour to the desired contour of the target pattern, as shown in Fig. 3. Each via will be processed edge by edge, and the measure points are placed at the center of each edge. EPE value is measured as the perpendicular distances between the lithography contour and desired edges at corresponding measure points.

Definition 2 (PV Band): The PV band is measured as the area between the outermost printed edge and the innermost printed edge among all process conditions, as shown in Fig. 4. It is used to evaluate the robustness of a mask against process variations.

C. Problem Definition

With these evaluation metrics, we aim to optimize mask quality and mask robustness against process variations through mask optimization.

Problem 1: Given a target design, the objective of mask optimization is to generate the corresponding lithography mask so that the distortion of the corresponding printed contour evaluated by EPE and the mask robustness under process variations evaluated by the PV band are optimized.

III. Methodology

A. RL Environment

A mask optimization flow includes subresolution assist feature (SRAF) insertion, OPC, and lithography compliance check. In RL-OPC, the SRAFs have been generated in all cases. An RL agent is trained based on given cases, which can sequentially determine the offsets of edges and optimize the corresponding masks.

The key concepts of RL-OPC formulation in our implementation are illustrated below.

1) *State (s):* A state s represents a local window centered at a via pattern, which consists of the patterns within the local window, including via patterns and SRAFs.
2) *Action (a):* An action a refers to an index and direction of the next moving edge. In our problem, the dimension of the action space is 12, which is spanned by four edges per via {left, right, top, bottom} and three possible moving directions {inner, outer, unchanged}. The moving distance is 1 nm by default per action, which can be naturally extended to adopt larger steps.
3) *Reward (r):* Reward r corresponds to the mask quality and robustness improvement after applying action a to an instance at state s. In RL-OPC, the mask quality improvement is represented by EPE and PV band reduction.
4) *Transition* (s_t, a_t, r_t, s_{t+1})*:* The next state s_{t+1} is generated when action a is applied on state s_t. With the corresponding reward r_t, a transition (s_t, a_t, r_t, s_{t+1}) includes the mapping relationship to be studied among states, actions, and rewards. A transition buffer is maintained to store the transitions.

In RL-OPC, an agent continuously interacts with its environment in sequential decision making over a limited number of discrete iterations. At the beginning of each iteration, the agent receives a state s_t containing a window centered at a target pattern. It then selects action a_t from a set of possible actions referring to its policy π in the form of a neural network parameterized by θ, which approximates the Q-value of the actions. After committing the action, the target pattern is modified. Then, the agent receives the next state s_{t+1}, while the reward r_t is obtained by calculating the EPE and PV band reduction after lithography simulation. This procedure continues until the maximum number of steps is achieved. Since the agent iteratively fetches the next state s_{t+1} and rewards r_t in interaction with the environment, the continuous update of the state forms a trajectory

$$s_0 \xrightarrow{a_0} (s_1, r_1) \xrightarrow{a_1} \dots \xrightarrow{a_{T-1}} (s_T, r_T) \tag{4}$$

with the maximum number of steps T allowed. The goal of the agent is to maximize its long-term reward

$$G_t = \sum_{k=0}^{T} \gamma^k r_{t+k} \tag{5}$$

where γ is a decay factor.

B. RL-OPC Framework

Fig. 5 reveals an overview of our RL-OPC framework. A deep convolutional neural network parameterized by θ_π is adopted as the approximation for the Q-function. Given a target design pattern, the agent updates each edge of the pattern iteratively and sequentially. Our framework, respectively, performs the following steps in each iteration: 1) encode local features of each state s_t; 2) the agent maps the input state to the next action a_t, from which the next state s_{t+1} is generated; 3) feed state s_{t+1} into lithography simulator, and a reward r is computed; 4) push collected transitions (s_t, a_t, r, s_{t+1}) into replay buffer; and 5) update model θ_π, and repeat steps 1)–5).

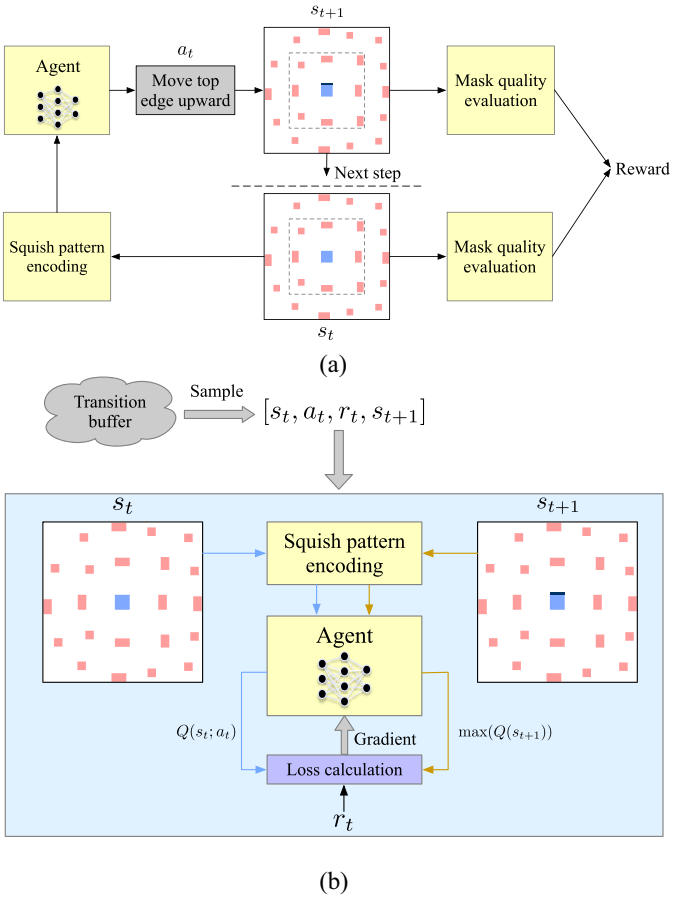

Fig. 5. (a) Forward process of RL-OPC. (b) Training illustration of RL-OPC.

Feature Encoding: After SRAF insertion, a local window centered at a via is extracted with all the other patterns inside, including surrounding SRAFs and other via patterns, as its local feature. The window size is set to 1500 nm × 1500 nm. Intuitively, we can directly transform the extracted window into an image and use the pixel-based representation as the feature, which has also been applied in previous works on lithography modeling [16] and OPC [9], [11]. However, such a large image may lead to huge computation overhead and memory consumption, impacting the efficiency of training and inference.

To alleviate this issue, we leverage a more compact layout representation. Squish pattern has been demonstrated to be a promising compact representation of the layout pattern in various manufacturability problems [17]. As demonstrated in Fig. 6, the encoded pattern consists of one binary matrix indicating the placement and two vectors indicating distance. The layout clip is first split into grids by superposing scan lines on shape edges.

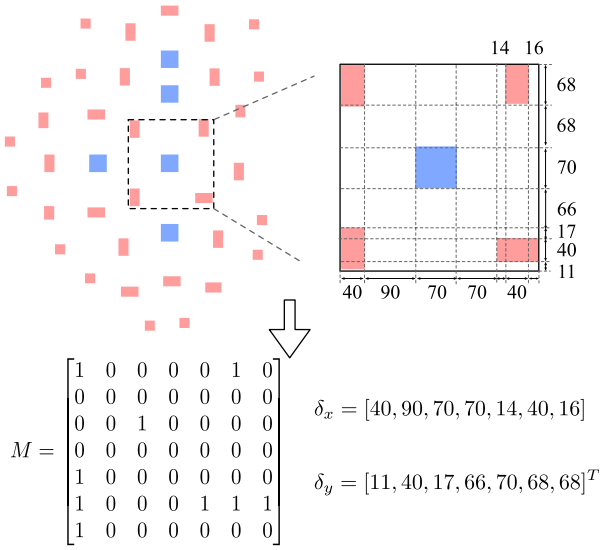

Fig. 6. Example of encoding binary pattern into squish pattern. The resulting binary matrix indicates relative placement, and vectors indicate distance.

The grid is then turned into matrix M, where entries with nonzero values indicate that the grid contains pattern geometry. Regarding the representations of the two pattern types: via and SRAF, there are two possible solutions for their encoding strategy: encode into different categories [10], [16] or treat identically [18], [19]. The first strategy considers the distinct lithographic effects of the two types of patterns, while the second solution simulates the mask in the real lithography process, where they both appear as cutouts on the same mask. While the two representations are both practical, we choose to treat the two types of patterns identically in RL-OPC. Hence, the entries in M are assigned 1 if the corresponding grids contain via or SRAF patterns, and otherwise 0.

The two vectors δ_x and δ_y indicate the horizontal and vertical grid sizes, respectively. For example, in Fig. 6, the horizontal grid sizes are $\{40, 90, 70, 70, 14, 40, 16\}$ nm, and the vertical grid sizes are $\{11, 40, 17, 66, 70, 68, 68\}$ nm, both start from the lower left corner. Note that the dimension of the input to the agent should remain unchanged for any local window, and the naive squish pattern representation cannot be utilized since the dimension of M is undetermined, although window sizes are identical. To meet the requirement, we further convert the squish pattern into an adaptive squish pattern [17], which is used as the feature representation of the state s and fed to the agent. To restrict the input dimension, more scan lines are added until the desired dimension of M is met, with δ_x and δ_y being scaled and duplicated accordingly. To determine the locations of new scan lines with low variances attained in grid sizes δ, the problem can be formulated mathematically as follows:

$$\begin{aligned} \min_{s} \quad & \left\|\delta'\right\|_\infty \\ \text{s.t.} \quad & \delta'_i = \delta_i / s_i \quad \forall i \\ & s_i \in \mathbb{Z}^+ \quad \forall i \\ & \sum_i s_i = d. \end{aligned} \tag{6}$$

The geometry information before and after scaling is denoted as δ and δ', the scaling and duplicating operation is encoded in s, and the desired number of scan lines in one direction is denoted by d. The underlying idea is to split the large grids in the original squish representation until M reaches the satisfactory dimension of $d_y \times d_x$. To formulate a standard input for a convolutional neural network, the scaled geometry information vectors δ'_x and δ'_y are duplicated for d_y and d_x times, respectively, forming two $d_y \times d_x$ matrices. Combined with M, the three matrices with identical resolution are concatenated with the dimension of $3 \times d_y \times d_x$.

Decision: With encoded state s_t as input, the network outputs a vector containing 12 elements, corresponding to 12 possible actions: move 1-nm inward, remain unchanged, and move 1-nm outward for four edges. The decision procedure appears as follows:

$$a_t = \arg\max_a Q(s_t, a|\theta_t) \tag{7}$$

where θ_t is the parameters of the network on the t-th step. The selected action a_t is then applied for mask updating.

Reward Calculation: As is demonstrated in Fig. 5(a), next state s_{t+1} is generated after applying action a_t to s_t. The corresponding EPE and PV band of state s_{t+1} are then obtained from lithography simulation. Our reward function is designed as follows:

$$\begin{aligned} r_t^{\text{EPE}} &= \frac{|\text{EPE}_t| - |\text{EPE}_{t+1}| + c}{|\text{EPE}_{t+1}| + \varepsilon} \\ r_t^{\text{PVB}} &= \frac{\text{PVB}_t - \text{PVB}_{t+1}}{\text{Area}_{\text{vias}}} \\ r_t &= r_t^{\text{EPE}} + \beta r_t^{\text{PVB}}. \end{aligned} \tag{8}$$

EPE_t and PVB_t refer to the EPE and PV band value of the layout at step t, respectively. $\text{Area}_{\text{vias}}$ refers to the total area of all vias in the input layout. c and ϵ are two predefined constants, and β denotes the weight coefficient. We design the reward function in a normalized temporal difference manner to restrict the reward range, thus benefiting model convergence. Meanwhile, since the lower bound of EPE is zero, a constant c is added to the numerator of the first term to keep the reward accumulated when the agent reaches the local optimum with a small EPE, thus encouraging convergence. The transition (s_t, a_t, r_t, s_{t+1}) is then pushed into the replay buffer for network θ_π updating.

Agent Training: The network updating procedure is depicted in Fig. 5(b). Similar to the standard training strategy in Q-learning, a replay buffer is maintained to record the historical data, i.e., past transitions. After inference, with the new transition stored, we randomly sample a batch of transitions (s_i, a, r, s_{i+1}) over the whole buffer.

As suggested by the Bellman equation, the target Q-value regarding the state–action pair at the sampled step is computed as

$$y = r + \gamma \max_{a'} Q\left(s', a'; \theta\right) \tag{9}$$

where γ is the discount factor. Based on the expected Q-value y, a gradient descent step could be performed over θ_t by

$$\theta' = \theta + \alpha \nabla_\theta (y - Q(s, a; \theta))^2. \tag{10}$$

Algorithm 1: RL-OPC Framework

Input: θ_0: Initial Q-network parameters; M: initial mask; $\tilde{M}$: real-time updated mask; γ: discount factor; T: maximum number of steps; T_B: number of steps for filling replay buffer.
Output: θ: Q-network parameters. Optimized mask.

Replay buffer $B \leftarrow \{\}$;
$N \leftarrow$ Number of vias in M;
$EPE_0, PVB_0 \leftarrow$LithoSim(M);
for $i \leftarrow 0$ **to** N **do**
  Get state s_0^i from the initial mask;
  for $t \leftarrow 0$ **to** T_B **do**
    Randomly select an action a_t;
    $s_{t+1}^i, r_t \leftarrow$ `EnvInteract`$(s_t^i, a_t, \tilde{M})$;
  Put $(s_t^i, a_t, r_t, s_{t+1}^i)$ to B;
$\tilde{M} \leftarrow M$;
for $t \leftarrow 0$ **to** T **do**
  for $i \leftarrow 0$ **to** N **do**
    $a_t \leftarrow \theta_t(s_t^i)$ by eq (7);
    $s_{t+1}^i, r_t \leftarrow$ `EnvInteract`$(s_t, a_t, \tilde{M})$;
  Put $(s_t^i, a_t, r_t, s_{t+1}^i)$ to B;
  Randomly sample $(s_{t'}, a_{t'}, r_{t'}, s_{t'+1})$ from B;
  Update Q network θ by eq. (9), eq. (10);
return $\tilde{M}, \theta$;
Def `EnvInteract`$(s_t, a_t, \tilde{M})$**:**
  Modify the mask $\tilde{M}$ by a_t;
  Get state s_{t+1} from $\tilde{M}$;
  $EPE_{t+1}, PVB_{t+1} \leftarrow$LithoSim($\tilde{M}$);
  $r_t \leftarrow$Calculate reward by eq. (8);
  return s_{t+1}, r_t;

The network is updated once a new transition is added to the replay buffer.

The learning and inferring procedure is presented in Algorithm 1. The agent first randomly samples action regarding each via, after which lithography simulation is performed. The agent then collects the transitions to fill the replay buffer, and samples transitions from the buffer to update the policy network. In the inferring stage, the agent processes each single via by generating T_E sequential actions, and the mask is real-time modified and evaluated once any action is decided. The agent processes all vias in a cyclical order. Since the mask is up-to-date, when encoding the local feature of each via, the historical modifications on its neighboring vias will be considered.

C. Multiobjective RL

Recall that in DQN, a deep neural network is employed as the policy, which takes the state as input and outputs Q-values w.r.t all possible actions. Thus, a feature extractor is employed, and the encoded feature vector is fed into a projection head to predict the Q-values. Since the target is to minimize EPE and PV band, a straightforward way is to integrate the goals into the reward function [20]. Hence, the Q-value consists of the expected accumulative reward of both EPE and PV band and is computed in a coarse-grained manner. However, physically EPE and PV band indicate the distortion of printed contour and the mask robustness, respectively. Specifically, target Q-value considering both EPE and PV band may result in the difficulty of network training a single projection head since identical network parameters are enforced to adapt to distinct tasks.

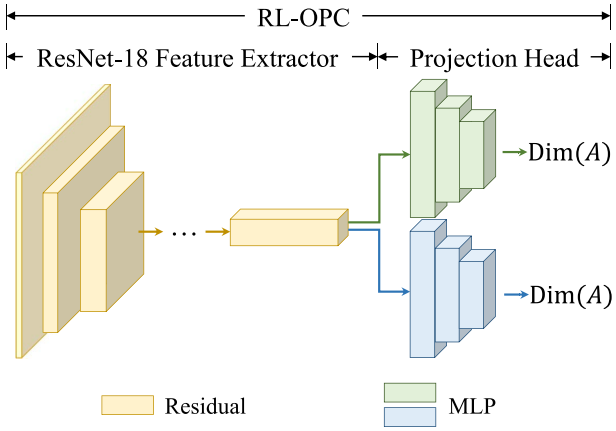

Fig. 7. Network structure with shared feature extractor and two projection heads. Dim(A) denotes that the output dimension is identical to the action space.

Hence, in addition to dual-optimization in the reward function (8), we further propose to optimize the two objectives in a more fine-grained manner.

Motivated by some multitask learning approaches that employ a shared feature extractor and distinct task-specific layers [21], [22], [23], in RL-OPC, a backbone-branching approach is performed to minimize the two objectives simultaneously, with their individual preference for different objectives being considered. As demonstrated in Fig. 7, following a shared feature extractor, we employ two independent projection heads to accommodate the two predictive tasks, i.e., EPE and PV band. Hence, the network outputs two Q-value vectors, each with the same dimension of action space, and represents the Q-values w.r.t two objectives corresponding to all actions. We then aggregate the vectors to the final Q-value as follows:

$$Q(s,a) = w_1 Q_{\text{EPE}}(s,a) + w_2 Q_{\text{PVB}}(s,a) \tag{11}$$

where $Q_{\text{EPE}}(s,a)$ and $Q_{\text{PVB}}(s,a)$ separately refer to the outputs of the two projection heads, and w_i is a coefficient factor for weight adjustment. $Q(s,a)$ is then utilized for decision and network updating, as illustrated in (7) and (10). Thus, the feature analysis of the two distinct tasks is decomposed from the backbone level for better adaptation.

D. ML-Based Initialization

If we inspect the principle of RL-OPC, it can be noted that RL-OPC can be further enhanced by taking a good initial mask, which can be generated by a preprocessing OPC

TABLE I
NETWORK PARAMETERS OF EMPLOYED PROJECTION HEAD. FC REFERS TO A FULLY CONNECTED LAYER

Name	Input Dim	Output Dim
FC1	512	512
FC2	512	128
FC3	128	12

model. We train a prediction model to generate coarsely optimized masks to validate the idea, which we utilized as the initial layer pattern for RL-OPC. Note that it is an optional step in the entire flow, and it can also be replaced by other approaches, including ML-based methods, traditional model-based methods, or human expertise. We employ a convolutional neural network θ_{pre} and adopt the same adaptive squish pattern representation as described in Section III-B as the model input to train a simple prediction model in a supervised learning manner. We generate the OPC results from a commercial tool of an EDA vendor, which are used as the labels for training. For each edge of the via pattern, we encode their offsets into signed integers y that indicate the direction and distance of deviation. Hence, the label of every single via appears as a vector with a dimension of four, referring to the expected offsets of the four edges. The loss function is the conventional mean-square error that is formulated as

$$\mathcal{L} = \left\| y_{\text{pre}} - f\left(s; \theta_{\text{pre}}\right) \right\|_2^2. \tag{12}$$

Before RL launches, the original layout is first fed into this preprocessing model to generate coarsely optimized results, which are then used as the initialization of the RL process.

IV. EXPERIMENTS

We implement the proposed framework in Python with the PyTorch framework on a CentOS-7 machine with an Intel i7-5930K 3.50-GHz CPU and Nvidia GeForce RTX 3090 GPU. For comparison with model-based and ML-based methods, we adopt the designs from [18], which contains a set of layouts with diverse amounts and placements of via patterns in the format of GDSII. The clips are with 2 μm $\times$ 2 μm large, and the vias are identical 70 nm $\times$ 70 nm square. For comparison with ILT, due to the lack of a unified benchmark, we adopt the designs from [24] which comprises 2048 nm $\times$ 2048 nm clips with 65 nm $\times$ 65 nm vias. Calibre from Siemens EDA is utilized for SRAF insertion. For all layouts, we sample the states by placing a 1500 $\times$ 1500 nm^2 window at the center of each via. The states are then encoded into 3 $\times$ 224 $\times$ 224 adaptive squish pattern [17]. The same operations are applied to the original layout patterns, including target via contour and identical SRAFs. Finally, the two 3 $\times$ 224 $\times$ 224 matrices are merged, and the dimension of RL-OPC network input becomes 6 $\times$ 224 $\times$ 224.

ResNet-18 [25] is employed as the feature extraction backbone, following which the preprocessing model in Section III-D utilizes an output layer with a dimension of four. Regarding the RL network, as demonstrated in Section III-C, two identical projection heads are employed for Q-value computation, consisting of a three-layer MLP. The detailed network structures of the projection head are presented in Table I.

TABLE II
RL-OPC ENHANCEMENT OVER INITIALIZATION

Design	Via #	Initialization		RL-OPC	
		EPE/via (nm)	PV band (nm^2)	EPE/via (nm)	PV band (nm^2)
Case 1	1	3.0	2937	1.0	2862
Case 2	2	2.5	5635	1.5	5579
Case 3	3	3.3	7750	2.7	7692
Case 4	3	5.7	8804	3.7	8640
Case 5	5	5.0	12373	4.2	12479
Case 6	1	3.0	2891	2.0	2912
Case 7	4	5.8	9508	4.3	9732
Case 8	4	9.8	8462	6.3	8828
Case 9	2	3.0	5720	2.5	5690
Case 10	2	4.0	5690	2.5	5614
Case 11	4	7.3	9367	4.8	9472
Case 12	2	3.5	4708	2.5	4759
Case 13	3	6.0	7636	4.3	7704
Average		5.361	2541	3.751	2555
Ratio		1.43	0.99	1.0	1.0

Adam optimizer is utilized, with the learning rate set to 10^{-4}. We adopt $L1$ loss as the loss function. After the next action is decided and applied to the current state, the modified mask is fed into the lithography simulator to evaluate its quality. The replay buffer is a first-in–first-out queue when new transitions are pushed in. Before the RL process launches, we fill the replay buffer by randomly selecting from the 12 actions and applying to each via to generate the transitions and perform offline network updating using the randomly generated replay buffer for five epochs. The maximum capacity of the replay buffer is set to 2000 transitions, and the sampling batch size while learning is set to 10. During the optimization process, while the design space exploration with temporarily increasing EPE is allowed, a reset mechanism is maintained: if the EPE value exceeds the historical best EPE for more than 60%, the mask will be reset to the best historical state. The maximum budget of RL forward steps T is a user-defined parameter depending on the users' demand on runtime, and is set to 10 in this section. Hence, RL-OPC runs for corresponding steps and outputs the best mask ever achieved within the limit. The multiobjective coefficient factors w_1 and w_2 in (11) are identically set to 1 by default, and the reward discount factor γ is set to 0.9. Hyperparameters c, ϵ, and β in (8) are, respectively, set to 1, 0.1, and 1 by default.

A. Feasibility of RL-OPC

To prove the effectiveness of RL-OPC, we first compare the initial masks generated by the preprocessing module and the RL-OPC in terms of EPE and PV band, as revealed in Table II. On average, RL-OPC reduces the EPE value by 43%, indicating the EPE drop of >1.5 nm per via.

The exploring curve of RL-OPC is revealed in Fig. 8, where the x-axis and the y-axis refer to forward steps and EPE numbers, respectively. During the optimization process, the mask with the historical best EPE will be recorded and updated, which RL-OPC outputs upon reaching the maximum step. As the figure depicts, RL-OPC explores the diverse solution space and places impressive improvements on input layers containing different pattern numbers. On average, our method takes 1.28 steps per via to optimize all cases to final states, and the corresponding runtime is less than 2 s per via.

TABLE III
COMPARISON WITH STATE-OF-THE-ART METHODS ON THE TEST DATASET. EPE, PV BAND, AND RUNTIME ARE MEASURED IN NANOMETERS (nm), SQUARE NANOMETERS (nm^2), AND SECONDS (s), RESPECTIVELY

Design	Via #	GAN-OPC [9]				DAMO [10]				Calibre				Ours			
		EPE/via	Max EPE	PVB	RT	EPE/via	Max EPE	PVB	RT	EPE/via	Max EPE	PVB	RT	EPE/via	Max EPE	PVB	RT
Case 1	1	7.0	7	2923	0.64	7.0	7	2917	0.57	4.0	4	2873	8.05	**1.0**	**1**	**2862**	1.25
Case 2	2	7.0	8	5738	0.67	14.0	14	**5433**	0.56	5.0	6	5716	8.06	**1.5**	**2**	5579	6.27
Case 3	3	5.0	7	8100	0.63	6.0	7	7975	0.57	4.0	5	7856	8.06	**2.7**	**3**	**7692**	1.26
Case 4	3	5.7	7	9024	0.68	10.7	12	9255	0.57	5.0	6	8793	8.06	**3.7**	**5**	**8640**	6.24
Case 5	5	5.8	9	13207	0.59	5.6	8	13108	0.56	4.2	**7**	12792	8.51	**4.2**	8	**12479**	5.16
Case 6	1	5.0	5	**2842**	0.62	7.0	7	2916	0.54	4.0	4	2889	8.01	**2.0**	**2**	2912	1.26
Case 7	4	9.3	10	10134	0.72	6.5	8	9902	0.54	5.5	8	10072	8.22	**4.3**	**7**	**9732**	6.26
Case 8	4	6.5	8	9339	0.74	9.8	11	8908	0.59	**5.5**	8	9326	8.18	6.3	**8**	**8828**	10.04
Case 9	2	7.0	7	5869	0.66	9.0	9	5691	0.56	5.0	7	5861	8.01	**2.5**	**3**	**5690**	1.23
Case 10	2	3.5	5	**5423**	0.64	9.5	10	5574	0.56	4.5	6	5705	8.23	**2.5**	**3**	5614	2.62
Case 11	4	5.8	7	9522	0.69	6.3	8	9882	0.58	**4.5**	**5**	9722	8.46	4.8	6	**9472**	10.17
Case 12	2	3.0	5	4956	0.65	7.0	8	4776	0.56	4.5	5	4859	7.92	**2.5**	**3**	**4759**	1.23
Case 13	3	6.0	7	7590	0.69	10.3	12	**7267**	0.57	**4.0**	**5**	7699	8.21	4.3	6	7704	5.01
Average		6.055	7.077	2630	0.24	8.111	9.308	2600	0.20	4.667	5.846	2616	2.94	**3.751**	**4.385**	**2555**	1.61
Ratio		1.614	1.614	1.029	0.15	2.163	2.123	1.018	0.12	1.245	1.333	1.024	1.83	**1.0**	**1.0**	**1.0**	1.0

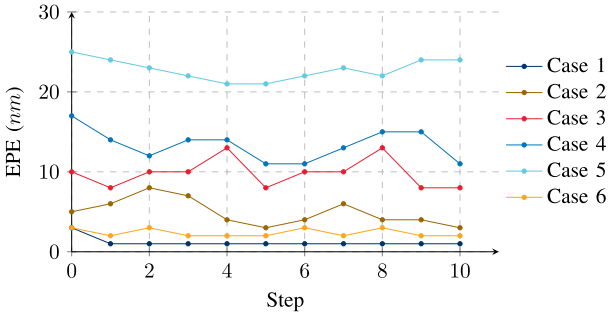

Fig. 8. EPE trajectories of six cases during the RL-OPC optimization process, corresponding to Cases 1–6 in Table III.

TABLE IV
COMPARISON WITH ILT

Design	Via #	ILT [6]			Ours		
		#EPEV/via	PVB(nm^2)	RT(s)	#EPEV/via	PVB(nm^2)	RT(s)
Via 1	6	0.50	14511	2.28	0.33	14515	0.46
Via 2	8	1.13	19022	2.26	0.88	18647	1.85
Via 3	9	0.56	22038	2.26	0.33	21803	1.85
Via 4	5	0.80	12036	2.26	0.80	11906	0.69
Via 5	12	0.33	30062	2.26	0.50	29821	2.77
Via 6	7	1.14	16321	2.26	0.71	16094	2.31
Via 7	5	1.00	11895	2.26	0.60	12181	1.62
Via 8	6	1.17	14493	2.26	1.00	14429	0.46
Via 9	9	1.00	21745	2.26	0.44	21465	1.85
Via 10	8	0.13	19545	2.26	0.25	19218	1.85
Average		0.73	2422.2	0.30	**0.56**	**2401.1**	**0.21**
Ratio		1.30	1.01	1.43	**1.0**	**1.0**	**1.0**

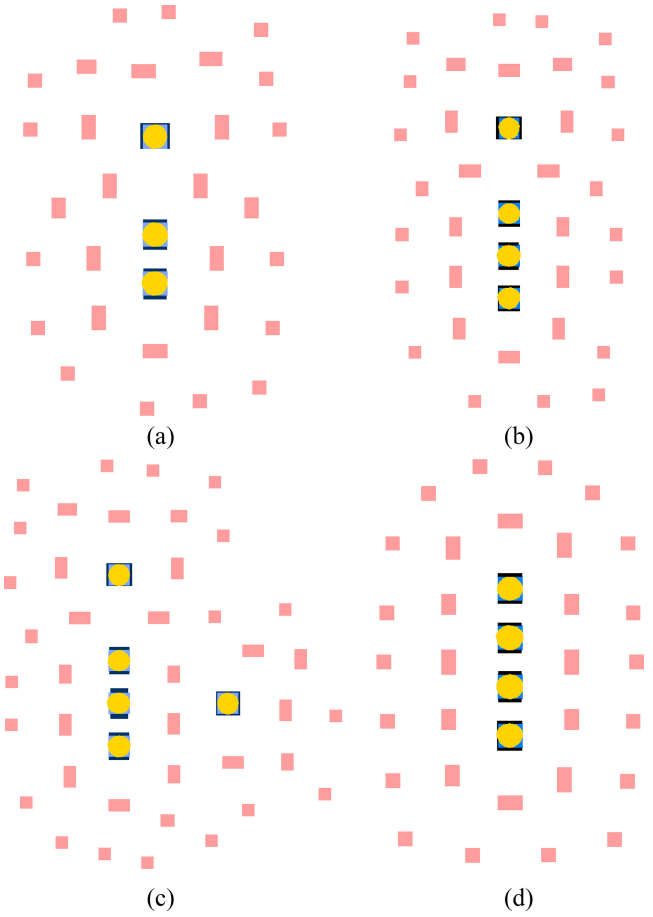

Fig. 9. Lithography simulation results of our method on multiple layouts with different numbers of vias. (a)–(d) correspond to Case 3, Case 7, Case 5, Case 8 in Table III, respectively.

The results visualization are revealed in Fig. 9. It contains SRAF patterns, target patterns, RL-OPC results, and printed contours from the simulator. The legend is identical to Fig. 1.

B. Comparison With Previous Arts

1) Model-Based and Learning-Based Methods: We compare our framework against state-of-the-art learning-based OPC methods (GAN-OPC [9] and DAMO [10]) and a commercial tool Calibre [26] that performs model-based OPC in terms of EPE and PV band. Note that RL-OPC in this section adopts initialization from the preprocessing module in Section III-D.

As is demonstrated in Table III, RL-OPC achieves the best EPE values in most test cases. Reported EPE values refer to the sum of EPEs of the entire layout. EPE/via denotes the averaged EPE values per via within each layout, and max EPE denotes the EPE on the most critical via. Specifically, our method achieves 61% improvement in averaged EPE value and 3% on averaged PV band than GAN-OPC. Compared with DAMO [10], RL-OPC can reduce the EPE by more than half

TABLE V
COMPARISON OF MULTIOBJECTIVE AGGREGATION W/WO MULTIHEAD NETWORK STRUCTURE

	Single-head		Multi-head	
	EPE/via (nm)	PV band (nm^2)	EPE/via (nm)	PV band (nm^2)
Case 1	1.0	2862	1.0	2862
Case 2	2.0	5587	1.5	5579
Case 3	2.7	7692	2.7	7692
Case 4	4.0	8724	3.7	8640
Case 5	4.4	12427	4.2	12479
Case 6	3.0	2910	2.0	2912
Case 7	4.0	9683	4.3	9732
Case 8	6.5	8823	6.3	8828
Case 9	2.5	5690	2.5	5690
Case 10	3.0	5673	2.5	5614
Case 11	4.3	9633	4.8	9472
Case 12	2.5	4759	2.5	4759
Case 13	4.3	7666	4.3	7704
Average	3.833	2559	3.751	2555
Ratio	1.022	1.002	1.0	1.0

and reduce the PV band by 2%. Moreover, compared with the commercial tool, RL-OPC outperforms Calibre in both EPE and PV band metrics by 24.5% and 2%, respectively. Overall, RL-OPC exceeds the compared methods in both metrics and balances between mask quality and runtime. For a few cases where other methods perform better, RL-OPC can serve as a complementary step over these methods, which is illustrated in Section IV-E.

2) ILT: Furthermore, we compare RL-OPC against an ILT method [6] in terms of mask quality and runtime, where an open-source implementation [27] is adopted as the baseline. Note that the EPEVs are measured in [6] instead of EPE values, which refers to the number of EPE measure points where EPE values exceed a predefined threshold. Hence, we follow the same criteria to evaluate the mask quality by EPEVs and PV band in this section, and the EPE threshold is set to 10 nm. As is revealed in Table IV, RL-OPC is capable of generating masks that are competitive with ILT. Among the ten test cases with various numbers of vias, the number of EPEVs is reduced by 30%. Meanwhile, a slight improvement in the PV band is achieved by RL-OPC, with a 43% improvement in runtime.

C. Comparison Between Strategies for Multiobjectives

We further conduct an ablative experiment to verify the effectiveness of multihead network structure in handling multiobjective tasks. Table V compares the quantitative EPE and PV band results of utilizing single-head and multihead backbone as the policy network. "Single head" suggests that the feature extractor is followed by one projection head, and the output Q-values are directly applied in the action decisions. The table reveals that the results of the multihead network structure slightly exceed those of the single-head model. In EPE evaluation, the multihead network reduces the EPE by 2.2% on average. A slight reduction can also be observed w.r.t PV band, proving the effectiveness of RL-OPC in accommodating multiple objectives.

D. Comparison on Large Layout

To prove the scalability of RL-OPC, we conduct an experiment on a large layout and compare RL-OPC and Calibre in terms of EPE, PV band, and runtime. We utilize a 15 μm $\times$ 40 μm synthesized layout that comprises 188 vias and employ the splitting algorithm from [10] to break the layout into 80 split windows in 2 μm $\times$ 2 μm. The number of vias in each split window varies from 1 to 5. For comparison, we test RL-OPC and Calibre on each split window and push the generated masks into the Calibre lithography simulator for evaluation. The numerical results are revealed in Table VI. The results are categorized by the number of vias in their input clip, and "case #" refers to the number of split windows in each category. Since the number of vias in each clip varies, the reported average results are the mean values among all vias instead of categories or clips. RL-OPC obtains competitive EPE results and a slight drop of 23 nm^2 in the PV band per via against Calibre. Regarding runtime, we utilize two Nvidia GeForce RTX 3090 GPUs, each accommodating one RL-OPC process. For a fair comparison, two Calibre processes are simultaneously launched for all clips. As Table VI depicts, RL-OPC is 85.8 s faster than Calibre on the 80 clips, which achieves 1.44$\times$ speedup on average compared with Calibre.

TABLE VI
COMPARISON OF RL-OPC AND CALIBRE ON THE LARGE LAYOUT WITH 188 VIAS

Bench	Clip #	Calibre			Ours		
		EPE(nm)	PVB(nm^2)	RT(s)	EPE(nm)	PVB(nm^2)	RT(s)
1-via	30	38	86088	106.6	35	86159	21.3
2-via	20	72	114020	71.1	66	113684	36.9
3-via	13	92	107300	46.2	104	106554	55.0
4-via	6	54	65577	21.1	80	64435	28.8
5-via	11	196	134371	39.1	158	132337	56.3
Sum	80	452	507356	284.1	443	503169	198.3
Avg/via		2.404	2699	1.51	**2.356**	**2676**	**1.05**

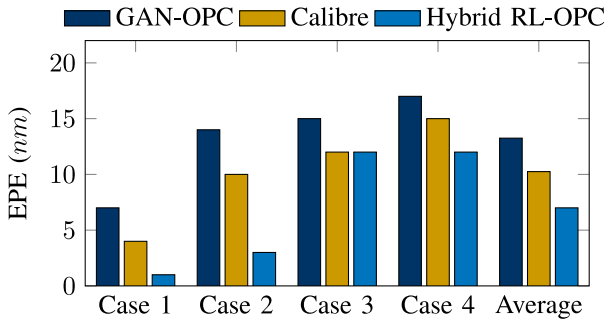

Fig. 10. Comparison of EPE results.

E. Enhanced With Existing Generative Methods

As we mentioned, RL-OPC can be further boosted with a pretrained OPC model. We integrate RL-OPC with a generative model [9], which serves as the initialization of RL-OPC. We denote this hybrid OPC model as *hybrid RL-OPC*.

The following sections analyze the enhancement in EPE, PV band, and runtime with *hybrid RL-OPC* and compare it with Calibre, as Figs. 10–12 depict.

In the figures, *GAN-OPC* and *Calibre* represent that only GAN-OPC and Calibre are utilized for mask optimization, respectively.

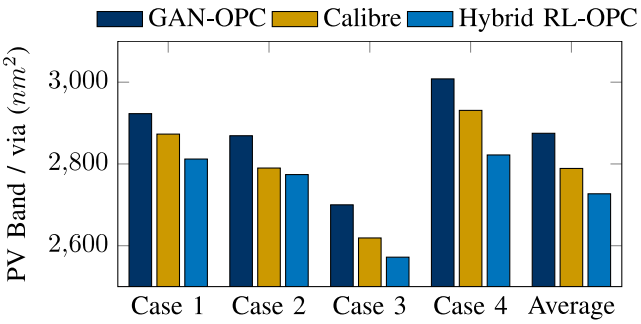

Fig. 11. Comparison of PV band results. PV band per via is revealed for clarity.

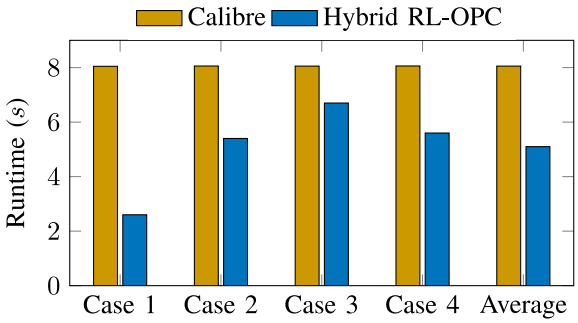

Fig. 12. Comparison of runtime.

The EPEs after launching RL-OPC are significantly reduced in all test cases. A notable decrease in EPE happens in Cases 1 and 2. On average, the integrated OPC flow reduces the EPE by 32% compared with Calibre. Since the PV band values vary due to different numbers of vias among the testcases, the PV band per via is compared in Fig. 11 for clarity. Compared to Calibre, results from *hybrid RL-OPC* are reduced in all cases, and drop by 62 nm^2 per via on average. Regarding runtime, we measure the time cost when reaching the best EPE value as the runtime for RL-OPC. As depicted in Fig. 12, *hybrid RL-OPC* can be more efficient than Calibre, which achieves 1.58× speedup with a 40% reduction on EPE in comparison with Calibre.

F. Runtime Analysis

We further conduct runtime analysis toward the RL steps in RL-OPC. Each RL step comprises four behaviors: 1) policy inference; 2) Calibre I/O; 3) lithography simulation; and 4) memory buffer updating. At the beginning of each step, the RL agent reads the input layouts and generates a modified output design, both in GDSII format. Calibre is then employed to perform lithography simulation. Note that since we lack structural data that can be directly embedded into the Calibre workflow, extra operations like I/O and preprocessing are required to accommodate the GDSII input. Hence, those extra operations are individualized as "Calibre I/O." Finally, with the evaluations from the lithography simulator, the new transition is pushed into the memory buffer and the next RL step is launched. We run RL-OPC for 20 steps, and the average runtime per step is approximately 5.46 s, with the proportions of each behavior demonstrated in Fig. 13.

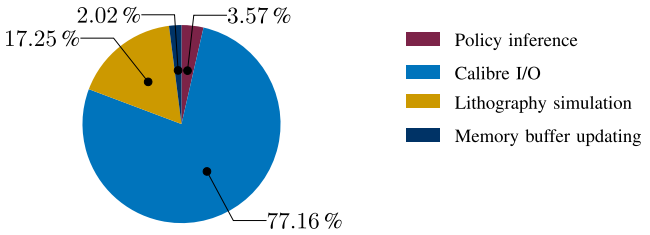

Fig. 13. Runtime of each behavior in an RL step.

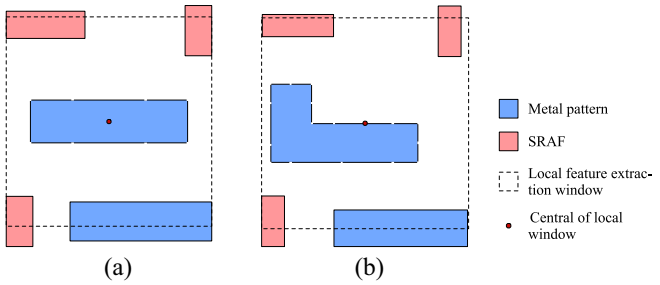

Fig. 14. Two examples of feature extraction on (a) rectangle and (b) polygon.

"Calibre I/O" occupies 77.16% of the runtime, indicating that 4.21 s out of 5.46 s are irrelevant with the RL approach. Regarding this runtime overhead, as depicted in Algorithm 1, every step in RL-OPC involves interaction with the lithography simulator, and the simulator we can access is only compatible with Calibre tool. Hence, we have to employ the GDSII file as the exchange format between RL agent and Calibre, and the runtime on I/O grows linearly since more RL steps are required for optimizing on larger layouts. Therefore, the runtime overhead by Calibre I/O can be eliminated if there is a standalone implementation of lithography simulation, with which the mask evaluation of RL-OPC can be performed internally without invoking any external tools. Currently, for fair comparison where only the runtime for RL computing and lithography simulation is involved, the precise runtime of RL-OPC should exclude "Calibre I/O," and hence is 1.25 s per step.

V. Discussion

In this section, we further discuss some future extensions of RL-OPC, including some problems that may be encountered and the potential solution. The topics include 1) extension to metal layer; 2) efficiency enhancement; and 3) parallel running on clips.

A. Metal Layer Extension

Although current RL-OPC aims at mask optimization on the via layer, it can be further extended to general patterns on the metal layer. The insight of RL-OPC is to learn the mapping from layout patterns to mask quality improvement, i.e., accumulated reward. Hence, the key to RL-OPC's adaptation to the metal layer becomes the definition of state, action, and reward on general polygons. In this section, we provide two examples of rectangles and polygons for demonstration.

1) For a relatively simple pattern like a rectangle, we assume it contains n segments with a fixed segmenting strategy. The state s could be formulated by placing the feature-extracting window at the center of the rectangle and encoding the region into a squish pattern as Fig. 14(a). The dimension of the action space becomes $3n$ in this situation, spanned by n segments and three possible moving directions {inner, outer, unchanged}, with the default stride set to 1nm. The reward can still be formulated as in (8), denoting EPE and PV band reduction on the layout after applying a to s.
2) For complex polygons, we provide an example of an L-shape pattern as demonstrated in Fig. 14(b). In this situation, RL-OPC could perform fine-grained optimization on the segment level. The state could be formulated by placing the local window at the center of each segment. Hence, the extracted feature corresponds to the segment instead of the entire pattern. The action a could be defined as the moving direction {inner, static, outer} and stride of the segment being processed. The reward could be formulated by (8). In case the segment-by-segment optimization is too time costly, parallel processing could be launched on distant segments for efficiency improvement. Meanwhile, the order of segments being processed could be decided by ranking the EPE values on their nearest measure point for boosting.

Therefore, RL-OPC is adaptive to general polygons, which we leave as a future direction.

B. Further Efficiency Improvement

Similar to previous model-based OPC techniques [2], [3], the efficiency of RL-OPC could be further improved by adaptively assigning the priority of vias awaiting. We first assume that an epoch is completed when all vias are processed once. Although RL-OPC processes each via in a cyclical manner, their order within a single epoch is undetermined. Hence, the efficiency of RL-OPC could be improved by dynamically ranking the real-time EPEs of each via at the beginning of each epoch, and starting from the critical ones.

On the other hand, RL-OPC's efficiency could be improved by training an evaluation network in replacement of the lithography simulator. Hence, an actor–critic framework [28] could be formulated, where the actor generates modifications to the mask which is evaluated by the critic. In this way, the framework simulates the behaviors of conventional model-based approaches but prevents the expensive lithography simulation.

C. Parallel Running on Clips

Section IV-D mentioned that the large layout is first split into multiple clips where RL-OPC is parallel launched. One potential problem is that the patterns on the boundaries might be disturbed by its neighboring clips due to light diffraction. Apart from setting overlapping areas in splitting, one possible solution is maintaining a shared replay buffer among the neighboring clips. By real-time uploading the modifications in the neighboring clips to the replay buffer, they could be considered in updating the local RL agent. However, the inter-clip correlation is still yet to be solved, not only in RL-OPC but also in any layout-splitting-related approach, and future research is expected in this field.

VI. Conclusion

In this article, we present RL-OPC, an RL-based mask optimization framework that undertakes wafer image distortion in the manufacturing process. Different from conventional learning-based approaches like discriminative models and generative models, RL-OPC does not rely on a supervised dataset generated by other OPC engines; meanwhile, RL-OPC is compatible with nondifferentiable objectives. Experimental results show that RL-OPC outperforms learning-based approaches and a commercial toolkit. Also, we prove that RL-OPC is effective in tuning results from other methods, thus further enhancing mask qualities.

Our future research focuses on the scalability of RL-OPC in mask optimization of large-scale chips. On the one hand, a network can be trained to replace the expensive lithography simulation for EPE calculation, which can further accelerate the process. On the other hand, parallel computation techniques can be explored to simultaneously optimize multiple instances when dealing with the full-chip scale layout. We hope our work can stimulate more research in this direction.

References

[1] O. W. Otto et al., "Automated optical proximity correction: A rules-based approach," in *Proc. SPIE*, 1994, pp. 278–293.
[2] J. Kuang, W.-K. Chow, and E. F. Y. Young, "A robust approach for process variation aware mask optimization," in *Proc. IEEE/ACM Design, Autom. Test Eurpoe (DATE)*, 2015, pp. 1591–1594.
[3] Y.-H. Su, Y.-C. Huang, L.-C. Tsai, Y.-W. Chang, and S. Banerjee, "Fast lithographic mask optimization considering process variation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 35, no. 8, pp. 1345–1357, Aug. 2016.
[4] A. Awad, A. Takahashi, S. Tanaka, and C. Kodama, "A fast process variation and pattern fidelity aware mask optimization algorithm," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2014, pp. 238–245.
[5] A. Poonawala and P. Milanfar, "Mask design for optical microlithography—An inverse imaging problem," *IEEE Trans. Image Process.*, vol. 16, pp. 774–788, 2007.
[6] J.-R. Gao, X. Xu, B. Yu, and D. Z. Pan, "MOSAIC: Mask optimizing solution with process window aware inverse correction," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2014, p. 52.
[7] B. Jiang, H. Zhang, J. Yang, and E. F. Young, "A fast machine learning-based mask printability predictor for OPC acceleration," in *Proc. IEEE/ACM Asia South Pacific Design Autom. Conf. (ASPDAC)*, 2019, pp. 412–419.
[8] A. Gu and A. Zakhor, "Optical proximity correction with linear regression," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 2, pp. 263–271, May 2008.
[9] 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 Trans. Comput.-Aided Design Integr. Circuits Systems (TCAD)*, 2020, pp. 1–6.
[10] G. Chen, W. Chen, Y. Ma, H. Yang, and B. Yu, "DAMO: Deep agile mask optimization for full chip scale," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, 2020, pp. 1–9.
[11] H.-C. Shao et al., "From IC layout to die photograph: A CNN-based data-driven approach," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 5, pp. 957–970, May 2021.
[12] G. Huang et al., "Machine learning for electronic design automation: A survey," *ACM Trans. Design Autom. Electron. Syst.*, vol. 26, no. 5, pp. 1–46, 2021.

[13] C. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, May 1992.
[14] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
[15] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[16] W. Ye, M. B. Alawieh, Y. Lin, and D. Z. Pan, "LithoGAN: End-to-end lithography modeling with generative adversarial networks," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2019, p. 107.
[17] H. Yang, P. Pathak, F. Gennari, Y.-C. Lai, and B. Yu, "Detecting multi-layer layout hotspots with adaptive squish patterns," in *Proc. IEEE/ACM Asia South Pacific Design Autom. Conf. (ASPDAC)*, 2019, pp. 299–304.
[18] K. Liu et al., "Adversarial perturbation attacks on ML-based CAD: A case study on CNN-based lithographic hotspot detection," *ACM Trans. Design Autom. Electron. Syst.*, vol. 25, no. 5, pp. 1–31, 2020.
[19] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, 2018, p. 131.
[20] Q. Xu et al., "GoodFloorplan: Graph convolutional network and reinforcement learning-based floorplanning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 10, pp. 3492–3502, Oct. 2022.
[21] R. Caruna, "Multitask learning: A knowledge-based source of inductive bias," in *Proc. 10th Int. Conf. Mach. Learn.*, 1993, pp. 41–48.
[22] R. Qin, Q. Liu, G. Gao, D. Huang, and Y. Wang, "MRDet: A multihead network for accurate rotated object detection in aerial images," *IEEE Trans. Geosci. Remote Sens.*, vol. 60, pp. 1–12, 2022.
[23] D. Keshwani, Y. Kitamura, S. Ihara, S. Iizuka, and E. Simo-Serra, "TopNet: Topology preserving metric learning for vessel tree reconstruction and labelling," in *Proc. 23rd Int. Conf. Med. Image Comput. Comput. Assist. Intervent.*, 2020, pp. 14–23.
[24] Y. Ma et al., "A unified framework for simultaneous layout decomposition and mask optimization," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 5069–5082, Dec. 2020.
[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 770–778.
[26] "Calibre design solutions." 2023. [Online]. Available: https://eda.sw.siemens.com/en-US/ic/calibre-design/
[27] S. Zheng et al.. "OpenILT: An open-source platform for inverse lithography technique research." 2023. [Online]. Available: https://github.com/OpenOPC/OpenILT/
[28] V. Konda and J. Tsitsiklis, "Actor–critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12, 1999, pp. 1–7.

Xiaoxiao Liang received the B.E. degree from the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, in 2020, and the M.S. degree from the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, in 2021. She is currently pursuing the Ph.D. degree with the Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China.

Her current research interests include computer-aided VLSI design, and design for manufacturability.

Yikang Ouyang received the B.E. degree from the School of Electronics and Information Technology, Sun Yat-sen University, Guangzhou, China, in 2022. He is currently pursuing the Ph.D. degree with the Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou.

His current research interests include fast circuit modeling and graph neural networks in EDA.

Haoyu Yang received the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 2020.

He is currently a Senior Research Scientist with the Design Automation Research Group, NVIDIA, Austin, TX, USA. His research interests include AI for electronic design automation, AI for computational lithography, GPU acceleration, and machine learning security.

Dr. Yang received the Best Paper Award from IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING in 2022 and the Best Paper Award Nomination from ASPDAC 2019. He is also the recipient of the 2019 Nick Cobb Scholarship from SPIE and Mentor Graphics.

Bei Yu (Senior Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received nine best paper awards from IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING in 2023, DATE 2022, ICCAD 2021 and 2013, ASPDAC 2021 and 2012, ICTAI 2019, *Integration, the VLSI Journal* in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, and six ICCAD/ISPD contest awards. He has served as the TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He is an Editor of the IEEE TCCPS Newsletter.

Yuzhe Ma (Member, IEEE) received the B.E. degree from the Department of Microelectronics, Sun Yat-sen University, Guangzhou, China, in 2016, and the Ph.D. degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong, in 2020.

He is currently an Assistant Professor with the Microelectronics Thrust, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou. His research interests include agile VLSI design methodologies, machine-learning-assisted VLSI design, and hardware-friendly machine learning.

Dr. Ma received the Best Paper Awards from ICCAD 2021, ASPDAC 2021, and ICTAI 2019, and the Best Paper Award Nomination from ASPDAC 2019.