NeurFill: Migrating Full-Chip CMP Simulators to Neural Networks for Model-Based Dummy Filling Synthesis

Junzhe Cai[1], Changhao Yan[1*], Yuzhe Ma[2], Bei Yu[2], Dian Zhou[3], Xuan Zeng[1*]
[1]State Key Lab of ASIC & System, Microelectronics Department, Fudan University, Shanghai, China
[2]The Chinese University of Hong Kong, Hong Kong [3]University of Texas at Dallas, USA

***Abstract*—Dummy filling is widely applied to significantly improve the planarity of topographic patterns for the chemical mechanical polishing (CMP) process in VLSI manufacturing. This paper proposes a novel model-based dummy filling synthesis framework NeurFill, integrated with multiple starting points-sequential quadratic programming (MSP-SQP) optimization solver. Inside this framework, a full-chip CMP simulator is first migrated to the neural network, achieving 8134× speedup on gradient calculation by backward propagation. Multi-modal starting points search is further applied in the framework to obtain satisfying filling quality optimums. The experimental results show that the proposed NeurFill outperforms existing rule- and model-based methods.**

I. Introduction

Chemical Mechanical Polishing (CMP) is widely applied to layout planarization in integrated circuit fabrications. To achieve uniform post-CMP topography, dummy fills are inserted in the sparse regions of layouts. However, dummy filling could induce additional parasitic capacitance and deteriorate the circuit performance [1]. The flow for dummy filling can be divided into two phases: filling synthesis and filling insertion [2]. The former determines the fill amount in each filling window, and the latter determines the shapes, locations of dummies in these windows. Though additional parasitic capacitance can be reduced by adjusting the shapes and locations of dummies in filling insertion [3]–[5], it is important to balance the planarization and performance degeneration in dummy filling synthesis [6].

Methods for dummy filling synthesis can be roughly classified into two categories: rule-based and model-based. Most research works are based on empirical knowledge of the CMP process, termed *rule*, for example, density variance, density gradient, etc. Rule-based dummy filling synthesis was first formulated as a linear programming problem to minimize density variance and fill amount [7]. To achieve better quality, multiple metrics such as overlay, line deviation, and outliers, are introduced to evaluate the filling result [8]–[11]. With the advancement of technology nodes, the intrinsic incompleteness of empirical rules limits the rule-based filling quality [12]. Therefore, model-based methods are applied. Tian *et al*. proposed a two-step model-based filling algorithm of global density assignment followed by local adjustment [13]. Sinha *et al*. proposed a model-based filling algorithm to minimize the post-CMP height range [14]. Both algorithms achieve excellent results, however, with simplified empirical models. Cai *et al*. proposed a full-chip CMP simulator-based filling framework optimizing multiple objectives and gained significant quality improvement [12]. However, even with parallel computing on 64 cores, [12] takes hours to complete computation and its filling quality depends on a single starting point.

Fig. 1 illustrates the basic flow of full-chip CMP model-based dummy filling synthesis. The target layout is divided into uniform windows, and the goal is to determine how many dummies to be inserted into each window. Usually, a layout can be divided into thousands of windows. Therefore, the optimization problem is extremely high dimensional. Inside the optimizer, the post-CMP height profile is generated by a full-chip CMP simulator and evaluated by the metrics. Numerical gradients of the model-based metrics are used to guide the optimization, which is time-consuming. Huge invocations of CMP simulators on numerical gradient calculation have become the bottleneck of model-based filling synthesis.

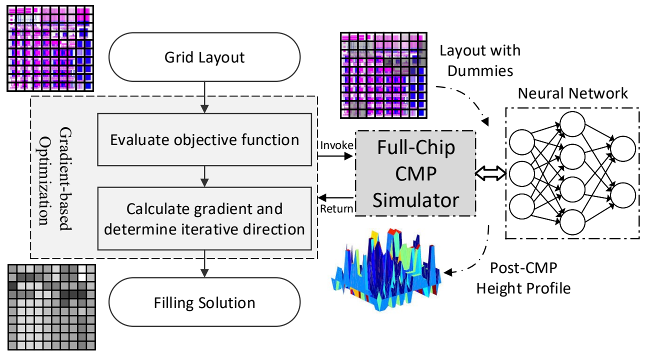

Fig. 1 The flow of model-based dummy filling synthesis and the motivation of leveraging neural network based CMP simulators.

In fact, massive simulator invocations can be easily observed in other model-based optimization problems, e.g., the optical proximity correction (OPC) problem. Recently, Jiang *et al*. proposed Neural-ILT for model-based OPC problem, leveraging neural network based simulation for acceleration [15]. Motivated by their work, we develop NeurFill to solve the efficiency bottleneck of model-based dummy filling synthesis. Because of the similarity to the image segmentation problem and the local effect of the CMP process, the full-chip CMP simulator is migrated to a pre-trained neural network as Fig. 1, where gradient calculation can be efficiently performed by backward propagation. Our main contributions can be summarized as follows.

1) We propose NeurFill, a model-based dummy filling synthesis framework based on a neural network. After migrating the conventional full-chip CMP simulator into a GPU-based neural network, we can achieve 188× speedup on objective evaluation and 8134× speedup on gradient calculation.
2) We leverage prior knowledge-based starting point for fast dummy filling synthesis. Due to the speedup in gradient calculation, sequential quadratic programming with multiple starting points (MSP-SQP) framework and multi-modal starting points search are proposed to obtain better filling quality without prior knowledge.
3) Experimental results show that the proposed NeurFill can achieve significant improvement in filling quality and overall score over the state-of-the-art.

The rest of the paper is organized as follows. Section II lists some preliminaries. Section III elaborates our motivation. Section IV gives

*Corresponding authors: {yanch, xzeng}@fudan.edu.cn

This research is supported partially by National Key R&D Program of China 2020YFA0711900, 2020YFA0711903, partially by National Natural Science Foundation of China (NSFC) research projects 61974032, 61774045, 61929102, partially supported by the Research Grants Council of Hong Kong SAR (No. CUHK14209420).

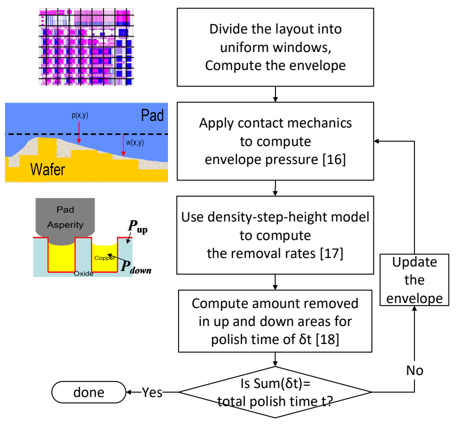
Pad
p(x,y)
w(x,y)
Wafer
Pad Asperity
P_{up}
Copper
P_{down}
Oxide
Divide the layout into uniform windows, Compute the envelope
Apply contact mechanics to compute envelope pressure [16]
Use density-step-height model to compute the removal rates [17]
Compute amount removed in up and down areas for polish time of δt [18]
Is Sum(δt)= total polish time t?
Yes
No
done
Update the envelope

Fig. 2 Framework of a full-chip CMP simulator.

the details of the proposed NeurFill framework. Section V presents experimental results, followed by a conclusion in Section VI.

II. Preliminaries

A. Full-Chip CMP Simulator

After 45nm technology node, full-chip CMP simulators have become the standard tools for overcoming DFM problems in foundries' reference flow. A typical full-chip CMP simulator takes a layout as input and outputs the predictive *dishing*, *erosion*, and average height profile of post-CMP chip surface. Generally, the simulation includes four steps as illustrated in Fig. 2: (1) Divide the whole chip into uniform windows and compute the envelope heights of the windows; (2) Solve the partial differential equations of contact mechanics and/or fluid mechanics to obtain the average window pressure [16]; (3) In each window, apply the density-step height (DSH) model [17] to compute the removal rates of up and down areas; (4) Compute the removed amounts within a unit polish time by the Preston equation [18]. All four steps iterate until a given total polishing time is reached.

B. Problem Formulation

Model-based evaluation metrics are modified from ICCAD 2014 dummy filling contest [8] to balance the planarization and performance degradation. In model-based dummy filling synthesis, an L-layer layout is divided into $L \times N \times M$ windows, and the window size is determined according to the CMP simulator.

Three objectives are calculated to evaluate the layout planarity, including height variance σ, line deviation σ^*, and outliers ol as

$$\sigma = \sum_{l=1}^{L} \frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} (H_{l,i,j} - \bar{H}_l)^2, \tag{1}$$

$$\sigma^* = \sum_{l=1}^{L} \sum_{i=1}^{N} \sum_{j=1}^{M} \left| H_{l,i,j} - \bar{H}_{l,j} \right|, \tag{2}$$

$$ol = \sum_{l=1}^{L} \sum_{i=1}^{N} \sum_{j=1}^{M} \max(0, H_{l,i,j} - 3 \cdot \sigma_l), \tag{3}$$

where $H_{l,i,j}$ is the height of window $W_{l,i,j}$, σ_l is the height variance of layer l, $\bar{H}_l$ and $\bar{H}_{l,j}$ are the average window height of layer l and of column j in layer l respectively. Two objectives overlay area ov and total fill amount fa are relevant to performance degradation. Total fill amount fa is the sum of each window's fill amount $x_{l,i,j}$ as

$$fa = \sum_{l=1}^{L} \sum_{i=1}^{N} \sum_{j=1}^{M} x_{l,i,j}, \tag{4}$$

Overlay area ov is estimated by the four-type region insertion in filling synthesis. Though the output file size fs is defined as a quality criterion, it is hard to formulate and not optimized. The objective of dummy filling synthesis is to maximize the filling quality score S_{qual} and can thus be formulated as

$$\max_{\mathbf{x}} [S_{qual} = S_{plan} + S_{PD}] \tag{5a}$$

$$S_{plan} = \alpha_\sigma f_\sigma(\mathbf{H}) + \alpha_{\sigma^*} f_{\sigma^*}(\mathbf{H}) + \alpha_{ol} f_{ol}(\mathbf{H}) \tag{5b}$$

$$S_{PD} = \alpha_{ov} f_{ov}(\mathbf{x}) + \alpha_{fa} f_{fa}(\mathbf{x}) \tag{5c}$$

$$\text{s.t. } x_{l,i,j} \in [0, s_{l,i,j}] \tag{5d}$$

where $\mathbf{x}$ and $\mathbf{H}$ refer to the vector of fill amount $x_{l,i,j}$ and height $H_{l,i,j}$ respectively; $s_{l,i,j}$ is the slack area in window $W_{l,i,j}$; f_σ, f_{σ^*}, f_{ol}, f_{ov} and f_{fa} refer to the score function of height variance, line deviation, outliers, overlay and fill amount respectively. The score function f can be generalized to

$$f(t) = \max(0, 1 - \frac{t}{\beta}), \tag{6}$$

where α_σ, α_{σ^*}, α_{ol}, α_{ov}, α_{fa} and β are benchmark-related coefficients.

III. Motivation

In the CMP process, both mechanical and chemical methods are applied. Due to its complexity, the CMP simulators are regarded as nonlinear black boxes in existing model-based algorithms. Therefore, analytic gradient of the CMP model cannot be easily derived, and numerical gradient is used instead. However, the calculation of numerical gradient requires thousands of CMP simulator invocations. As the runtime of numerical gradient calculation dominates the runtime of model-based methods [12], the efficiency is strictly constrained.

A. Acceleration with Neural Network

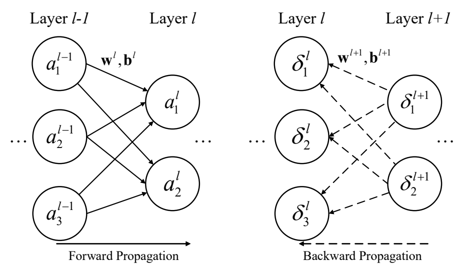

Fig. 3 The forward and backward propagation on neural networks.

Fig. 3 illustrates the basic idea of forward and backward propagation on neural networks. In forward propagation, the activation of l-th layer $\mathbf{a}^l$ is related to the activation of (l-1)-th layer $\mathbf{a}^{l-1}$ by

$$\mathbf{a}^l = f(\mathbf{w}^l \mathbf{a}^{l-1} + \mathbf{b}^l), \tag{7}$$

where $\mathbf{w}^l$ and $\mathbf{b}^l$ are the weight matrix and bias vector of l-th layer. f is the activation function. In back-propagation, the error, as well as gradient, of l-th layer δ^l is related to the error of (l+1)-th layer δ^{l+1} by

$$\delta^l = (\mathbf{w}^{l+1})^T \delta^{l+1} \odot f'(\mathbf{z}^l), \tag{8}$$

where $\odot$ is the Hadamard product, f' is the gradient of activation function and $\mathbf{z}^l$ is the weighted input of l-th layer as

$$\mathbf{z}^l = \mathbf{w}^l \mathbf{a}^{l-1} + \mathbf{b}^l. \tag{9}$$

The key idea of forward and backward propagation on neural networks is that the activation of the current layer is only related to the previous layer, and the gradient is only related to the latter layer. Therefore, backward propagation works far faster than other gradient approaches like numerical gradient calculation.

B. Similarity Between CMP Model and Image Segmentation

If the full-chip CMP model can be migrated to the neural networks, to some extent, the black box of the CMP model is opened, and fast gradient calculation can be performed by backward propagation.

Fortunately, we observe the underlying similarity between the CMP model and the image segmentation problem. In the image segmentation problem, the input is an image with pixels, which contains the information of RGB colors, and the output is a grayscale image indicating the areas of each segment. On the other hand, the input of the CMP model is an extracted grid layout, where each window contains parameters such as pressure, trench height and density, perimeter of coppers, etc. The output of the CMP model is a post-CMP height profile, providing a positive height of each window. Besides, due to the contact mechanics of rough polishing pads in the CMP process, the character length in the CMP model is ranged from 20$\sim$100 um [16], which limits the number of correlation windows. The local effect in the CMP model is similar to the convolutional kernels of neural networks. The intrinsic similarity between two problems, the local effect in the CMP model, and the potential acceleration motivate us to leverage pre-trained neural networks replacing the conventional CMP simulator.

IV. The NeurFill Framework

Due to the complexity of multi-objective optimization, traditional linear programming optimization can likely lead to a suboptimal solution. Therefore, sequential quadratic programming (SQP) [19] with multiple starting points (MSP) is applied in the NeurFill framework for model-based dummy filling synthesis.

The key part of MSP-SQP algorithm is to obtain the objective evaluation and gradient calculation in each iteration. As Equation (5a), filling quality score S_{qual} can be divided into the planarity score S_{plan} which involves the CMP model, and the performance degradation score S_{PD} which does not need simulation. In NeurFill, CMP neural network with an integrated pre-trained CMP model can fast obtain the score and gradient of planarity through forward and backward propagation. The performance degradation score and its analytic gradient are calculated by performance degradation estimation. A prior knowledge-based starting point is used for fast dummy filling synthesis, while a multi-modal starting points search is performed for better filling quality.

A. CMP Neural Network

Based on the similarity between the CMP model and the image segmentation problem, we selected UNet [20], a powerful convolutional neural network, to replace the full-chip CMP simulator. Besides, modern deep learning toolkit (e.g., *PyTorch*) provides efficient functions with built-in CUDA-based forward and backward propagation. Automatic backward propagation can be done by utilizing the toolkit.

As illustrated in Fig. 4, CMP neural network consists of an extraction layer, the pre-trained UNet, and objective layers. CMP neural network takes target GDS file and fill amount $\mathbf{x}$ as inputs, and outputs the planarity score S_{plan}. In the extraction layer, pattern-related parameters of each window such as density, average width, length, perimeter of coppers, and process-related parameters such as pressure, heights of trench side and bottom, are extracted into a layout parameter matrix $\mathbf{L}$. Pattern-related parameters in $\mathbf{L}$ are updated with regard to fill amount $\mathbf{x}$ according to the DSH model [17], and the gradient of extraction layer $\frac{\partial \mathbf{L}}{\partial \mathbf{x}}$ can be calculated automatically by deep learning toolkit. Given the layout parameter matrix $\mathbf{L}$, the pre-trained UNet generates the post-CMP height profile $\boldsymbol{H}_n$, which is analyzed by the following objective layers to calculate the planarity score S_{plan}.

By utilizing toolkit functions, height variance σ, line deviation σ^*, and outliers ol in Equations (1) to (3) can be expressed as:

$$\sigma = VAR(\boldsymbol{H}_n), \tag{10a}$$

$$\sigma^* = SUM(ABS((\boldsymbol{H}_n - MEAN(\boldsymbol{H}_n, 1) \cdot ONES(1, M))), \tag{10b}$$

$$ol = SUM(SIGMOID(\eta(\boldsymbol{H}_n - 3 \cdot VAR(\boldsymbol{H}_n)))), \tag{10c}$$

where VAR, SUM, ABS, $MEAN$, $ONES$, $SIGMOID$ are functions from torch toolkit, η is a hyper-parameter. Since the definition of outliers ol in Equation (3) is non-differentiable, sigmoid function with hyper-parameter η is used to replace the original piecewise function. The merging layer merges the objective scores to form the planarity score S_{plan} as Equation (5b). Automatic gradient calculation of toolkit can be performed on Equation (5b) for $\frac{\partial S_{plan}}{\partial \sigma}$, $\frac{\partial S_{plan}}{\partial \sigma^*}$, $\frac{\partial S_{plan}}{\partial ol}$ in merging layer, and on Equation (10) for $\frac{\partial \sigma}{\partial \boldsymbol{H}_n}$, $\frac{\partial \sigma^*}{\partial \boldsymbol{H}_n}$, $\frac{\partial ol}{\partial \boldsymbol{H}_n}$ in height variance, line deviation, outliers layer respectively.

Therefore, with CMP neural network, the planarity score S_{plan} can be calculated by forward propagation, and the gradient ∇S_{plan} can be calculated by backward propagation through the chain rule:

$$\begin{aligned}\nabla S_{plan} =& \frac{\partial S_{plan}}{\partial \sigma}\frac{\partial \sigma}{\partial \boldsymbol{H}_n}\frac{\partial \boldsymbol{H}_n}{\partial \mathbf{L}}\frac{\partial \mathbf{L}}{\partial \mathbf{x}} + \frac{\partial S_{plan}}{\partial \sigma^*}\frac{\partial \sigma^*}{\partial \boldsymbol{H}_n}\frac{\partial \boldsymbol{H}_n}{\partial \mathbf{L}}\frac{\partial \mathbf{L}}{\partial \mathbf{x}} \\ &+ \frac{\partial S_{plan}}{\partial ol}\frac{\partial ol}{\partial \boldsymbol{H}_n}\frac{\partial \boldsymbol{H}_n}{\partial \mathbf{L}}\frac{\partial \mathbf{L}}{\partial \mathbf{x}},\end{aligned} \tag{11}$$

where $\frac{\partial S_{plan}}{\partial \sigma}$, $\frac{\partial \sigma}{\partial \boldsymbol{H}_n}$, $\frac{\partial \boldsymbol{H}_n}{\partial \mathbf{L}}$ and $\frac{\partial \mathbf{L}}{\partial \mathbf{x}}$ can be obtained by backward propagation in the merging layer, the height variance layer, the pre-trained UNet and the extraction layer respectively as Fig. 4. The same backward propagation procedure can be done for other parts in Equation (11).

B. Performance Degradation Estimation

The performance degradation score S_{PD} evaluates the parasitic capacitance induced by dummy filling as Equation (5c). Since there is no participation of the CMP model, analytic gradients can be derived. The analytic gradient of total fill amount ∇fa can be derived from Equation (4) as

$$\nabla fa = J_{L,N,M}, \tag{12}$$

where $J_{L,N,M}$ is the all-ones matrix.

The overlay area remains unknown until the locations of dummies are determined. In this paper, four-type region insertion is used to estimate overlay area. As illustrated in Fig. 5, because overlay only exists in the vertical direction, fillable slack regions of a window can be divided into four types according to upper and lower layer conditions. The dummies of each window are inserted to slacks by the priority from type 1 to 4 as $x^1_{l,i,j}$, ..., $x^4_{l,i,j}$. Total overlay can be divided into dummy-to-wire overlay ov^{d-w} and dummy-to-dummy overlay ov^{d-d}. Dummy-to-wire overlay can be calculated as

$$ov^{d-w} = \sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{M} x^2_{l,i,j} + x^3_{l,i,j} + 2x^4_{l,i,j}. \tag{13}$$

In window $W_{l,i,j}$, dummy-to-dummy overlay $ov^{d-d}_{l,i,j}$ counts the overlay with upper layer as

$$ov^{d-d}_{l,i,j} = \max(0, x^1_{l,i,j} + x^1_{l+1,i,j} - s^*_{l,i,j}), \tag{14}$$

where $s^*_{l,i,j}$ refers to the area of non-overlap slacks between layer l and l+1. Therefore, overlay area can be calculated as

$$ov = ov^{d-w} + \sum_{l=1}^{L}\sum_{i=1}^{N}\sum_{j=1}^{M} ov^{d-d}_{l,i,j}, \tag{15}$$

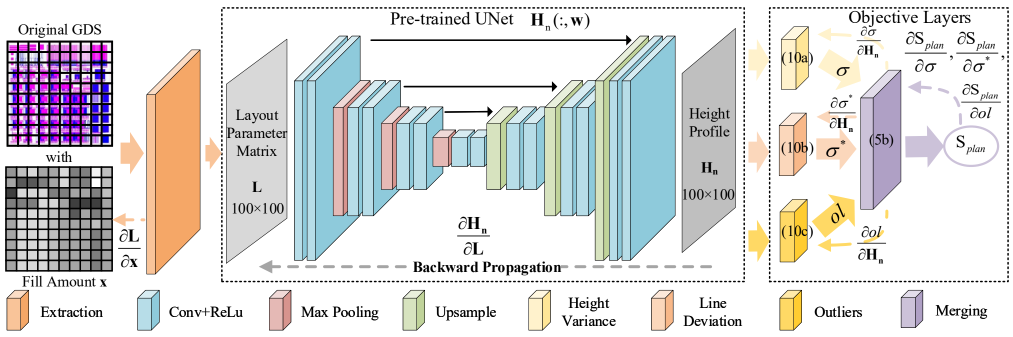
Original GDS
with
Fill Amount x
$\frac{\partial L}{\partial x}$
Pre-trained UNet $H_n(:, w)$
Layout Parameter Matrix
L
100×100
Height Profile
H_n
100×100
$\frac{\partial H_n}{\partial L}$
Backward Propagation
Objective Layers
$\frac{\partial \sigma}{\partial H_n}$
$\frac{\partial S_{plan}}{\partial \sigma}, \frac{\partial S_{plan}}{\partial \sigma^*}$,
$\frac{\partial S_{plan}}{\partial ol}$
(10a)
σ
(10b)
σ^*
$\frac{\partial \sigma^*}{\partial H_n}$
(5b)
S_{plan}
(10c)
ol
$\frac{\partial ol}{\partial H_n}$
Extraction
Conv+ReLu
Max Pooling
Upsample
Height Variance
Line Deviation
Outliers
Merging

Fig. 4 Overview of CMP neural network.

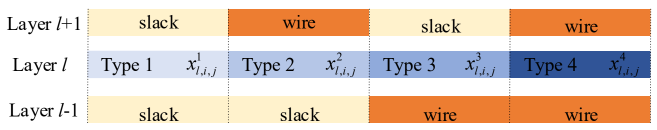

Fig. 5 Four types of fillable slack region in a window.

and analytic gradient of overlay area can be derived as

$$\nabla ov_{l,i,j} = \begin{cases} 0, & \text{if } x^1_{l,i,j} + x^1_{l+1,i,j} < s^*_{l,i,j}; \\ 2, & \text{if } x^4_{l,i,j} > 0; \\ 1, & \text{otherwise.} \end{cases} \tag{16}$$

Therefore, analytic gradient of performance degradation score ∇S_{PD} can be derived from Equation (5c) as:

$$\nabla S_{PD} = -\frac{\alpha_{fa}}{\beta_{fa}} \nabla fa - \frac{\alpha_{ov}}{\beta_{ov}} \nabla ov, \tag{17}$$

where α_{fa}, α_{ov}, β_{fa} and β_{ov} are benchmark-related constants.

C. Prior Knowledge Based Starting Point Generation

Modified from the rule-based target density planning [10], prior knowledge-based (PKB) starting point for the model-based algorithm is proposed in [12]. In this method, target layer density td_l is first determined in advance for each layer, representing each window's expected density. After td_l is determined, a trivial solution to fill dummies for maximum density uniformity can be obtained,

$$x_{l,i,j} = \begin{cases} 0, & \text{if } td_l < \rho_{l,i,j}; \\ s_{l,i,j}, & \text{if } td_l > \rho_{l,i,j} + s_{l,i,j}; \\ td_l - \rho_{l,i,j}, & \text{otherwise,} \end{cases} \tag{18}$$

where $\rho_{l,i,j}$ and $s_{l,i,j}$ is the wire density and the slack area of window $W_{l,i,j}$ respectively. A linear search of target layer density is performed, and the solution with the best quality is chosen as the starting point.

D. Multi-modal Starting Points Search

In dummy filling synthesis problem with thousands of dimensions, a random starting point may be trapped in a suboptimal local maximum. Even with prior knowledge, there is no guarantee that PKB starting point's filling quality is not suboptimal. To maximize the filling quality, all local optimums of the quality score function should be evaluated.

The problem of locating all the local optimums is termed multi-modal optimization [21], formulated as

$$\max |XS = \{\mathbf{x}^{lo}_i, i = 1, 2, \cdots, n\}| \tag{19}$$

where $|\cdot|$ is the size of set, and $\mathbf{x}^{lo}_i$ is the i-th local optimum.

Fig. 6 illustrates the topography of the quality score function of a layout with two fillable windows. Four peak regions of local optimums are depicted in red circles. An efficient multi-modal optimization solver can locate all global and local optimums accurately. In general, multi-

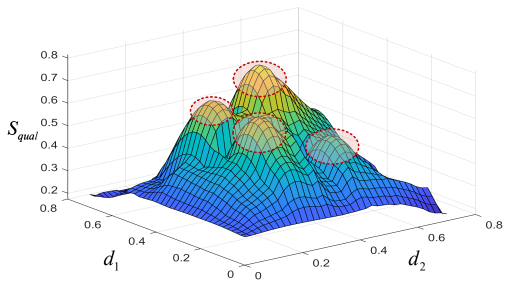

Fig. 6 The quality score function of a layout with two fillable windows.

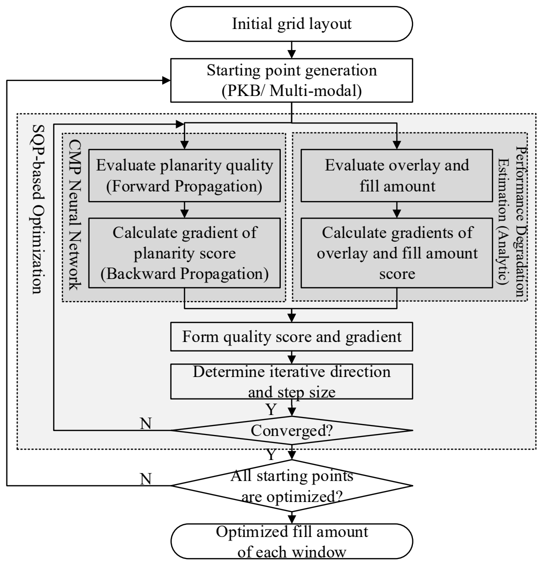
Initial grid layout
Starting point generation (PKB/ Multi-modal)
SQP-based Optimization
CMP Neural Network
Evaluate planarity quality (Forward Propagation)
Calculate gradient of planarity score (Backward Propagation)
Performance Degradation Estimation (Analytic)
Evaluate overlay and fill amount
Calculate gradients of overlay and fill amount score
Form quality score and gradient
Determine iterative direction and step size
Converged?
N
Y
All starting points are optimized?
N
Y
Optimized fill amount of each window

Fig. 7 Framework of model-based MSP-SQP NeurFill Algorithm.

modal solvers are usually combined with evolutionary algorithms, generating multiple subpopulations. Each subpopulation represents one peak region of the target function. In this paper, the niching migratory multi-swarm optimizer (NMMSO) [22] is applied to explore the problem space. When the NMMSO method converges, it outputs all the possible subpopulations in the problem space, representing potential peak regions of the quality score function.

E. MSP-SQP framework of NeurFill

Fig. 7 summarizes the MSP-SQP framework of NeurFill. The starting points are generated by the PKB method or multi-modal search, then SQP optimization is applied to these starting points. The scores and gradients of planarity and performance degradation are calculated by CMP neural network and performance degradation estimation respec-

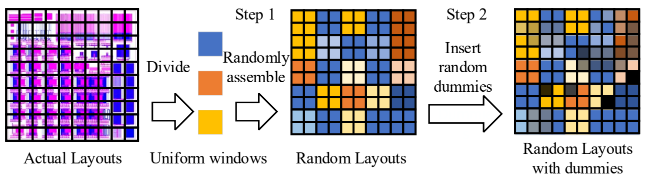

Fig. 8 Two-step random procedure of training data generation.

tively, and merged to form the filling quality score and gradient, then the direction and step size of the current iteration are determined.

F. Pre-training of UNet Model

As depicted in Fig. 4, the UNet architecture consists of a down-sampling path to capture features and an up-sampling path to generate the post-CMP height profile. Given a set of extracted layout parameter matrix $\mathcal{L} = \{\boldsymbol{L}_1, \boldsymbol{L}_2, \cdots, \boldsymbol{L}_n\}$ and the corresponding post-CMP height profile set $\mathcal{H}_s = \{\boldsymbol{H}_{s1}, \boldsymbol{H}_{s2}, \cdots, \boldsymbol{H}_{sn}\}$ generated by the full-chip CMP simulator, the training procedure of UNet is to minimize the following objective:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \lambda ||\boldsymbol{H}_n(\mathcal{L}, \boldsymbol{w}) - \mathcal{H}_s||_2^2, \tag{20}$$

where $\boldsymbol{H}_n(:, \boldsymbol{w})$ is the network output with respect to weights $\boldsymbol{w}$, and λ is a configurable hyper-parameter.

The input and output dimensions are fixed for neural networks. In this paper, the layout size for UNet is 100×100 windows. Layouts that are smaller than the fixed size will be duplicated several times to cover the whole input layout. Since it is hard to find enough actual layouts for UNet training, a two-step random procedure is applied in this paper to generate training data, as depicted in Fig. 8. First, the three available layouts are divided into uniform windows. These windows are randomly assembled to generate 200 layouts of 100×100 windows. Second, random dummies are inserted into the assembled layouts with no design rule violation. Finally, 20,000 layouts are generated by the two-step random procedure and simulated by the full-chip CMP simulator as the training set. The two-step random procedure aims to produce training instances that are close to the layouts neural networks may process in the filling optimization. Besides, the extension ability of the pre-trained UNet model is verified by the training set generated by two layouts and a testing set generated by the third.

V. Experimental Results

The NeurFill framework is developed in Python with *PyTorch* and runs on a Linux server with 2.6GHz Intel Xeon CPU and Nvidia Tesla K80 GPU. Compared rule-based algorithms Lin [10], Tao [11] and model-based algorithm Cai [12] are developed in C/C++ and run on a Linux server with 64-core 2.6GHz Intel Xeon CPU. The full-chip CMP simulator is calibrated under a 45nm process of a foundry, and the accuracy is matched with the CMP Predictor [23], a commercial full-chip CMP simulator by Cadence. The window sizes of the full-chip CMP simulator and dummy filling synthesis are both 100um×100um. Three layout designs are used in neural network training and filling performance comparison. Design A is a CMP test design with chip size 5cm×5cm and file size 16.4MB. Design B is a field programmable gate array (FPGA) design with chip size 6.7cm×6.3cm and file size 948.7MB. Design C is a RISC-V CPU design with chip size 10cm×10cm and file size 80.6MB.

A. The Accuracy of Pre-trained Model

20,000 layout instances in the training set are generated by the two-step random procedure based on three available layouts. UNet is pre-trained for 20 epochs on the GPU for 32 hours, then the accuracy of the pre-trained UNet model is evaluated by a testing set of 1000 layout instances.

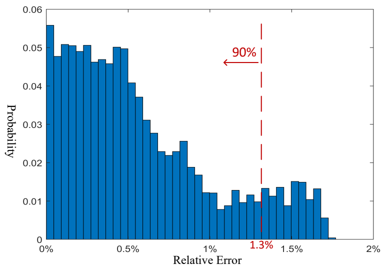

Fig. 9 Average relative error distribution of height in windows.

TABLE I Runtime Comparisons for Objective Evaluation and Gradient Calculation

Operation	Full-chip CMP Simulator (1c)	Full-chip CMP Simulator (64c)	CMP Neural Network (GPU)	Speedup (GPU v.s. 64c)
Objective Evaluation	4.7s	4.7s	0.025s	188×
Gradient Calculation	34100s	545s	0.067s	8134×

The average relative error of the post-CMP height profile generated by the pre-trained UNet and the full-chip CMP simulator in the testing set is 0.6%. For each window in the layout, average relative error of height is measured in the testing set, and the distribution is illustrated in Fig. 9. The maximum average relative error of height in a specific window is 1.77%, and the average relative error of height is below 1.3% in 90% of the windows. Besides, the average relative error of the post-CMP height profile in the extension ability testing set is 2.7%. Therefore, the accuracy of the pre-trained UNet model is acceptable compared to the full-chip CMP simulator.

B. The Acceleration of CMP Neural Network

TABLE I shows the runtime comparisons for objective evaluation and gradient calculation between the full-chip CMP simulator and the proposed CMP neural network. For single-precision performance, GPU's computation capability is up to 8.74 TFLOPS, while 64-core CPU is up to 8.12 TFLOPS. The computation capabilities of two platforms are considered equivalent.

On a layout design of 100×100 windows, CMP neural network performs objective evaluation in 0.025s by forward propagation, which achieves 188× speedup. For gradient calculation, the backward propagation of CMP neural network is 8134× faster than the numerical gradient calculation of the 64-core full-chip CMP simulator. The efficiency of CMP neural network enables MSP-SQP framework and multi-modal starting points search for optimization.

C. Filling Quality Comparison

TABLE II shows the score function coefficients for objectives. TABLE III shows the comparison results of the proposed NeurFill algorithm and existing rule- and model-based methods on three designs. In the column *Method*, NeurFill (PKB) is the proposed method of the PKB starting point followed by SQP optimization in the NeurFill framework. NeurFill (MM) is the proposed method of MSP-SQP optimization with multi-modal starting points search in the NeurFill framework.

Considering the runtime and memory usage, NeurFill (PKB) gets the best overall score over all three existing methods. The overall score of NeurFill (PKB) is 8.6% better than the best existing method Cai [12] in Design A, 15.9% better than Tao [11] in Design B, and 2.2% better than Lin [10] in Design C.

When focusing on filling quality, model-based algorithms outperform rule-based algorithms. Compared to state-of-the-art model-based method Cai [12], NeurFill (PKB) obtains 2% better quality with 58×

TABLE II Score Function Coefficients of Three Layout Designs

	#L	File Size	α_{ov}	β_{ov}	α_{fa}	β_{fa}	α_{σ}	β_{σ}	α_{σ^*}	β_{σ^*}	α_{ol}	β_{ol}	α_{fs}	β_{fs}	α_t	β_t	α_m	β_m
A	3	16.4M	0.15	2400724	0.05	2400724	0.2	209	0.2	78132	0.15	7.1	0.05	32.8M	0.15	20min	0.05	8G
B	3	948.7M	0.15	6596491	0.05	6596491	0.2	133	0.2	23616	0.15	25	0.05	1897.4M	0.15	20min	0.05	8G
C	3	80.6M	0.15	3232445	0.05	3232445	0.2	105	0.2	17281	0.15	17	0.05	161.2M	0.15	20min	0.05	8G

TABLE III Performance Comparison On Three Layout Designs

Design	Method	ΔH	Performance	Variation	Line Deviation	Outliers	File Size	Runtime	Memory	Quality	Overall
A	Lin [10]	174Å	0.000	0.145	0.445	1.000	0.967	1.000 (1s)	0.756	0.395	0.504
	Tao [11]	174Å	1.000	0.142	0.425	1.000	0.970	0.968 (39s)	0.756	0.640	0.695
	Cai [12]	139Å	1.000	0.595	0.750	1.000	0.989	0 (1.5h,64c)	0.756	0.835	0.706
	NeurFill (PKB)	135Å	0.978	0.613	0.765	1.000	0.976	0.375 (12.5min)	0.832	0.837	**0.767**
	NeurFill (MM)	128Å	0.973	0.625	0.769	1.000	0.971	0 (3.0h)	0.725	**0.839**	0.708
B	Lin [10]	283Å	0.000	0.425	0.525	0.333	0.683	1.000 (1s)	0.707	0.343	0.459
	Tao [11]	283Å	1.000	0.445	0.530	0.380	0.714	0.915 (1.7min)	0.707	0.610	0.660
	Cai [12]	219Å	1.000	0.457	0.550	0.973	0.874	0 (3.0h,64c)	0.707	0.739	0.627
	NeurFill (PKB)	204Å	0.960	0.571	0.608	1.000	0.823	0.695 (6.1min)	0.832	0.774	**0.765**
	NeurFill (MM)	197Å	0.957	0.577	0.614	1.000	0.817	0 (4.9h)	0.613	**0.776**	0.651
C	Lin [10]	241Å	1.000	0.424	0.260	1.000	0.822	0.993 (9s)	0.712	0.660	0.712
	Tao [11]	240Å	1.000	0.430	0.271	0.999	0.820	0.910 (1.8min)	0.712	0.664	0.703
	Cai [12]	198Å	0.277	0.809	0.816	1.000	0.580	0 (17.2h,64c)	0.712	0.699	0.595
	NeurFill (PKB)	198Å	0.274	0.843	0.824	1.000	0.558	0.800 (4.0min)	0.832	0.707	**0.728**
	NeurFill (MM)	199Å	0.298	0.844	0.847	1.000	0.607	0 (18.3h)	0.590	**0.723**	0.608

speedup. With 21% longer runtime compared to Cai [12], NeurFill (MM) can obtain the highest quality scores, which are 0.5%, 5.0% and 3.4% better on three designs.

Besides, NeurFill (MM) has two advantages. First, the filling quality of NeurFill (PKB) is unpredictable, while filling quality optimized by NeurFill (MM) is the best among all available local optimums located by the multi-modal algorithm. In our opinion, it is worthwhile to spend extra time for the certainty of better filling quality. Second, the prior knowledge used in NeurFill (PKB) is based on the empirical rules, which have shown intrinsic drawbacks in deep-submicron technology. Even if NeurFill (PKB) is efficient under the $45nm$ technology node, it may result in a bad local optimum under other advanced technology nodes. On the other hand, NeurFill (MM) can be adapted to any technology node and generate a satisfying filling solution.

VI. Conclusion

In this paper, we propose NeurFill, a model-based dummy filling synthesis framework based on a neural network. In NeurFill, CMP neural network achieves 8134× speedup on gradient calculation by backward propagation. With multi-modal starting points search and MSP-SQP framework, the satisfying filling solution can be obtained. Furthermore, we believe that the proposed framework can be extended to other high-dimensional model-based optimizations with proper adjustment of the pre-trained neural network model.

References

[1] W.-S. Lee, K.-H. Lee, J.-K. Park, T.-K. Kim, Y.-K. Park, and J.-T. Kong, "Investigation of the capacitance deviation due to metal-fills and the effective interconnect geometry modeling," in *Proc. ISQED*, 2003, pp. 373–376.

[2] C. Feng, H. Zhou, C. Yan, J. Tao, and X. Zeng, "Efficient approximation algorithms for chemical mechanical polishing dummy fill," *IEEE TCAD*, vol. 30, no. 3, pp. 402–415, 2011.

[3] B. Jiang, X. Zhang, R. Chen, G. Chen, P. Tu, W. Li, E. F. Young, and B. Yu, "FIT: Fill insertion considering timing," in *Proc. DAC*, 2019, pp. 1–6.

[4] A. B. Kahng and R. O. Topaloglu, "Performance-aware CMP fill pattern optimization," *Proc. VMIC*, pp. 3–9, 2007.

[5] S.-J. Yu, C.-C. Kao, C.-H. Huang, and I. H.-R. Jiang, "Equivalent capacitance guided dummy fill insertion for timing and manufacturability," in *Proc. ASPDAC*, 2020, pp. 133–138.

[6] T. Lan, X. Li, J. Chen, J. Yu, L. He, S. Dong, W. Zhu, and Y.-W. Chang, "Timing-aware fill insertions with design-rule and density constraints," in *Proc. ICCAD*, 2019, pp. 1–8.

[7] A. B. Kahng, G. Robins, A. Singh, and A. Zelikovsky, "Filling algorithms and analyses for layout density control," *IEEE TCAD*, vol. 18, no. 4, pp. 445–462, 1999.

[8] R. O. Topaloglu, "ICCAD-2014 CAD contest in design for manufacturability flow for advanced semiconductor nodes and benchmark suite," in *Proc. ICCAD*, 2014, pp. 367–368.

[9] C. Liu, P. Tu, P. Wu, H. Tang, Y. Jiang, J. Kuang, and E. F. Young, "An effective chemical mechanical polishing filling approach," in *Proc. ISVLSI*, 2015, pp. 44–49.

[10] Y. Lin, B. Yu, and D. Z. Pan, "High performance dummy fill insertion with coupling and uniformity constraints," *IEEE TCAD*, vol. 36, no. 9, pp. 1532–1544, 2017.

[11] Y. Tao, C. Yan, Y. Lin, S.-G. Wang, D. Z. Pan, and X. Zeng, "A novel unified dummy fill insertion framework with SQP-based optimization method," in *Proc. ICCAD*, 2016, pp. 1–8.

[12] J. Cai, C. Yan, Y. Tao, Y. Lin, S.-G. Wang, D. Z. Pan, and X. Zeng, "A novel and unified full-chip CMP model aware dummy fill insertion framework with SQP-based optimization method," *IEEE TCAD*, vol. 40, no. 3, pp. 603–607, 2021.

[13] R. Tian, D. Wong, and R. Boone, "Model-based dummy feature placement for oxide chemical-mechanical polishing manufacturability," *IEEE TCAD*, vol. 20, no. 7, pp. 902–910, 2001.

[14] S. Sinha, J. Luo, and C. Chiang, "Model based layout pattern dependent metal filling algorithm for improved chip surface uniformity in the copper process," in *Proc. ASPDAC*, 2007, pp. 1–6.

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

[16] C. Feng, C. Yan, J. Tao, X. Zeng, and W. Cai, "A contact-mechanics-based model for general rough pads in chemical mechanical polishing processes," *J. Electrochem. Soc.*, vol. 156, no. 7, p. H601, 2009.

[17] H. Cai *et al.*, "Modeling of pattern dependencies in the fabrication of multilevel copper metallization," Ph.D. dissertation, Massachusetts Inst. Technol., 2007.

[18] L. M. Cook, "Chemical processes in glass polishing," *J. Noncrystalline Solids*, vol. 120, no. 1-3, pp. 152–171, 1990.

[19] P. T. Boggs and J. W. Tolle, "Sequential quadratic programming," *Acta Numerica*, vol. 4, no. 1, pp. 1–51, 1995.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. MICCAI*, 2015, pp. 234–241.

[21] X. Wang, T. Gu, C. Yan, X. Wu, F. Yang, S.-G. Wang, D. Zhou, and X. Zeng, "An efficient and robust yield optimization method for high-dimensional SRAM circuits," in *Proc. DAC*, 2020, pp. 1–6.

[22] J. E. Fieldsend, "Running up those hills: Multi-modal search with the niching migratory multi-swarm optimiser," in *Proc. IEEE Congr. Evol. Comput.*, 2014, pp. 2593–2600.

[23] Cadence corp., "CMP predictor," https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/silicon-signoff/cmp-predictor.html.