A Novel Unified Dummy Fill Insertion Framework with SQP-Based Optimization Method

Yudong Tao
Fudan University
Shanghai, China
11300720039@fudan.edu.cn

Changhao Yan*
Fudan University
Shanghai, China
yanch@fudan.edu.cn

Yibo Lin
University of Texas at Austin
Austin, TX, US
yibolin@cerc.utexas.edu

Sheng-Guo Wang
University of North Carolina at Charlotte
Charlotte, NC, US
swang@uncc.edu

David Z. Pan
University of Texas at Austin
Austin, TX, US
dpan@cerc.utexas.edu

Xuan Zeng*
Fudan University
Shanghai, China
xzeng@fudan.edu.cn

ABSTRACT

Dummy fill insertion is widely applied to significantly improve the planarity of topographic patterns for chemical mechanical polishing process in VLSI manufacture. However, these dummies will lead to additional parasitic capacitance and deteriorate the circuit performance. The main challenge of dummy filling algorithms is how to balance multiple objectives, such as fill amount, density variation, parasitic capacitance, etc. which is the aim of ICCAD 2014 DFM contest. Traditional dummy fill insertion methods are no longer applicable because they generate large amount of fills or take unaffordable time. In this paper, we propose a unified dummy fill insertion optimization framework based on multi-starting points and sequential quadratic programming optimization solver, where all objectives are considered simultaneously without approximation. Selecting the initial points smartly with prior knowledge, the proposed method can be effectively accelerated. Even without any prior knowledge, it can also reach high fill quality by random initial points with high scalability. The proposed algorithm is verified by ICCAD 2014 DFM contest benchmark, which shows better quality of dummy filling over the state-of-the-art algorithms.

CCS Concepts

•Hardware → Physical design (EDA); Design for manufacturability;

Keywords

Dummy Fill Insertion; Sequential Quadratic Programming Method (SQP); Multi-Starting Point Technique (MSP)

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

ICCAD '16, November 07-10, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4466-1/16/11…$15.00
DOI: http://dx.doi.org/10.1145/2966986.2966994

1. INTRODUCTION

Chemical Mechanical Polishing (CMP) is widely applied to layout planarization in integrated circuit fabrication. The quality of post-CMP topography strongly depends on pattern features of layout, especially the pattern density uniformity [9]. Dummy fill insertion is generally processed to improve layout pattern density uniformity for CMP, where those logically non-functional patterns, dummy fills, are inserted in the sparse regions of layout. However, dummy fill insertion could induce additional parasitic capacitance and deteriorate the circuit performance [11]. Planarizing layout and avoiding performance degradation should be balanced and optimized by adjusting the amount, location and even shape of dummy fills.

Since a calibrated full chip CMP simulator is complicated and expensive, most research works of dummy fill insertion are based on prior knowledge of CMP process. And it is formulated as an optimization problem, where those experiences are formulated as optimal objectives or constraints. Traditionally, only pattern density variation and dummy fill amount are taken into consideration during problem formulation, so dummy fill insertion problem is generally formed as linear programming (LP) problem [8, 15]. Although an optimal solution can be sought under such formulation, these methods suffer from long runtime, especially when the circuit scale is getting larger and the necessary window size turns smaller [9].

Heuristic methods based on Monte Carlo (MC) are thus proposed to enhance the efficiency of algorithm [3, 4, 18]. The solutions are iteratively approached by filling predetermined amount of dummies in selected tile in each iteration. However, these methods either cannot ensure the satisfaction of performance constrains or still demand unaffordable runtime. The determination of fill quantity in each iteration has no clear guidance to follow as well. Meanwhile, a covering linear programming (CLP) formulation was proposed in [5, 6], which can effectively and efficiently balance planarity and fill amount, but the overlay is not considered carefully.

Moreover, Kahng *et al.* pointed out that overlay area is an important indicator of performance degradation other than fill amount [10], so it should be considered during problem formulation of dummy filling. Since the overlay is determined by all locations of fill, the new problem is formulated as integer linear programming (ILP), while both LP and MC methods are not directly applicable. Chen *et al.* [2] proposed the first ILP-based algorithm for dummy fill insertion and Xiang *et al.* [19] proposed an improved one to simultaneously optimize layout planarity and control overlay area.

To motivate the development of dummy fill insertion algorithm, ICCAD 2014 launched a DFM contest for dummy fill insertion algorithm based on a set of industrial benchmark suites [17]. This new benchmark uses multiple objectives to formulate the quality of dummy fills, including fill amount, overlay, density variation, line deviation and outliers, and the quality of method, including runtime and memory cost. Several methods have been proposed based on this benchmark. Liu *et al.* proposed a LP-based algorithm [13] and effectively produce solutions considering layout planarity and fill amount. However, overlay is ignored in its optimization process. Meanwhile, Lin *et al.* proposed an improved ILP-based algorithm [12] taking all the objectives into account, which shows great improvements over the top teams on ICCAD 2014 DFM contest. However, in [12] the non-linear factors in ICCAD 2014 DFM contest benchmark are simplified into the linear formulation, which makes it possibly suboptimal for the benchmark.

Additionally, rule-based approaches for dummy fill insertion have an intrinsic drawback that the rules can hardly describe the complex behaviours of the full chip CMP. Therefore, the results of the rule-based methods, even based on ICCAD 2014 DFM contest benchmark, remains dubious. Instead, model-based methods [14] utilize CMP simulator to completely calculate post-CMP topography and thus can avoid the incompleteness of rules for generating better solution to dummy fill insertion. Unfortunately, only few researches are performed on model-based methods, and the efficiency of model-based algorithms should be enhanced.

In this paper, we develop a novel unified dummy fill insertion framework based on Sequential Quadratic Programming (SQP) with multi-starting point technique (MSP), where both rule and model-based methods can be applied. Our main contributions can be summarized as follows.

1. This paper proposes a novel unified dummy fill insertion framework, which formulates optimization objectives without simplification and approximation. Therefore, this framework is also extended to model-based method, or situations where the objectives are hard to simplify.

2. In order to avoid the trap of local optimal solution caused by complex objectives, SQP method [1] is applied togeter with MSP to obtain the optimized solution. The closed forms of objective function, gradient vector and Hessian Matrix of objective function are derived, which accelerates the SQP optimization tremendously.

3. This paper proposes an overlay estimation algorithm, which can accurately estimate the reachable minimal overlay before fill insertion. Meanwhile, a prior knowledge based starting point generation is proposed to improve both fill quality and efficiency.

4. The experimental results show significant improvement in the quality of fill solution over the results of existing methods on ICCAD 2014 DFM contest benchmarks [17].

The rest part of this paper is organized as follows: Section 2 presents the details of ICCAD 2014 DFM contest benchmark and general problem formulation; Section 3 shows details of our dummy fill insertion framework; Section 4 gives the experimental results and Section 5 concludes this work.

2. BENCHMARK

In this paper, ICCAD 2014 DFM contest benchmark is used for evaluating the quality of dummy filling algorithm. The benchmark requires four aspects of objectives, including layout planarity, performance degradation, file size and computer resource cost.

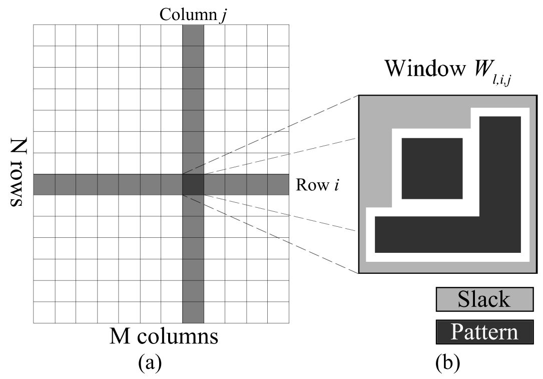

Figure 1: Example of Layout Grid and Window

In ICCAD 2014 DFM contest benchmark, a chip layout is divided into $N \times M$ square windows of given size for density calculation, as shown in Fig. 1(a), where $W_{l,i,j}$ denotes the window at row i and column j of layer l, the density $\rho_{l,i,j}$ of $W_{l,i,j}$ is the ratio of total pattern area to window area, and the slack area $s_{l,i,j}$ is the total area of blank regions for dummy filling in window $W_{l,i,j}$ as shown in Fig. 1(b), where the narrow blank space surrounding patterns is left for DRC rules.

Three objectives are calculated to evaluate the layout planarity, including density variation σ, line deviation σ^* and outliers ol. In detail, σ represents the standard deviation of window densities, σ^* represents the summation of density variation per window from column average and ol represents the summation of density variation per window beyond 3σ corner. These objectives are repeatedly computed for each layer and the results are added up. For each layer, they are calculated as

$$\sigma_l = \sqrt{\frac{1}{N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} (\rho_{l,i,j} - \bar{\rho}_l)^2} \tag{1a}$$

$$\sigma_l^* = \sum_{i=1}^{N} \sum_{j=1}^{M} |\rho_{l,i,j} - \bar{\rho}_{l,j}| \tag{1b}$$

$$ol_l = \sum_{i=1}^{N} \sum_{j=1}^{M} \max(0, \rho_{l,i,j} - 3 \cdot \sigma_l) \tag{1c}$$

where $\bar{\rho}_l$ and $\bar{\rho}_{l,j}$ refer to the average window density of layer l and of column j in layer l respectively, i.e.

$$\bar{\rho}_l = \frac{1}{M} \cdot \sum_{j=1}^{M} \bar{\rho}_{l,j}, \quad \bar{\rho}_{l,j} = \frac{1}{N} \cdot \sum_{i=1}^{N} \rho_{l,i,j} \tag{2}$$

Parasitic capacitance is another major concern during dummy fill insertion. [10] and [16] point out that overlay area and total fill amount during dummy fill insertion are two major factors relevant to performance degradation of post-CMP circuit. In the benchmark, the total overlay area ov is the summation of all overlay in adjacent layers, and total fill amounts fa are used to evaluate performance degradation. They are defined as

$$ov = \sum_{l=1}^{L-1} \sum_{i=1}^{N} \sum_{j=1}^{M} (ov_{l,i,j}^{d-d} + ov_{l,i,j}^{d-l} + ov_{l,i,j}^{l-d}) \tag{3a}$$

$$fa = \sum_{l=1}^{L} \sum_{i=1}^{N} \sum_{j=1}^{M} x_{l,i,j} \tag{3b}$$

where $ov_{l,i,j}^{d-d}$ refers to overlay between dummies in window $W_{l,i,j}$ of the l-th layer and in $W_{l+1,i,j}$ of the $(l+1)$-th layer, where the supperscript $d-d$ means *dummy-to-dummy*. In the same manner, $ov_{l,i,j}^{d-l}$ and $ov_{l,i,j}^{l-d}$ refers to *dummy-to-layout* and *layout-to-dummy* overlay between $W_{l,i,j}$ and $W_{l+1,i,j}$, respectively. $x_{l,i,j}$ refers to the fill amount in window $W_{l,i,j}$, and L is the total layer number of a chip layout.

Furthermore, the inputs of benchmark are a layout with blackboxs for hiding signal wires, and densities of each window. The objective of the benchmark is to maximize Obj as

$$Obj = Quality + Comp \tag{4a}$$

$$Quality = f_{pd}(ov + fa) + f_{\sigma}(\sigma) + f_{\sigma^*}(\sigma^*) + f_{ol}(ol) + f_{fs}(fs) \tag{4b}$$

$$Comp = f_t(t) + f_{mem}(mem) \tag{4c}$$

where the fill quality $Quality$ includes scores of overlay and fill amount f_{pd}, layout planarity f_{σ}, f_{σ^*}, f_{ol}, and file size of output layout f_{fs}. The computing cost $Comp$ of algorithm includes the scores of run time f_t and memory f_{mem}. The weighted score function is defined in [17] with the form $f_X(x) = \alpha_X \cdot \max(0, 1 - \frac{x}{\beta_X})$, where α_X and β_X are given benchmark-based weighted coefficients, and X can be pd, σ, σ^*, ol, fs, t and mem. Additionally, since file size is hard to formulated, it is not optimized in this paper.

3. FRAMEWORK OF DUMMY FILL INSERTION

The framework of the proposed dummy fill insertion algorithm is shown in Fig. 2, which mainly includes three phases: density analysis, dummy fill synthesis and dummy fill insertion. In density analysis phase, window densities $\rho_{l,i,j}$ and slacks $s_{l,i,j}$ are calculated in each window as shown in Fig. 1(b). In dummy fill synthesis, the problem is firstly formulated into an optimization problem without approximation and simplification. Then in order to solve this optimization problem, multiple starting points are generated and they are optimized by sequential quadratic programming method [1] to obtain better $Quality$ defined in (4b). Since SQP is designed for non-linear optimization, this framework should be available even if the objective is modified or model-based dummy filling method is applied. Finally, the locations of all dummies are determined in dummy fill insertion phase, where the overlay and number of fills are balanced to decrease file size, and stored in a GDS file.

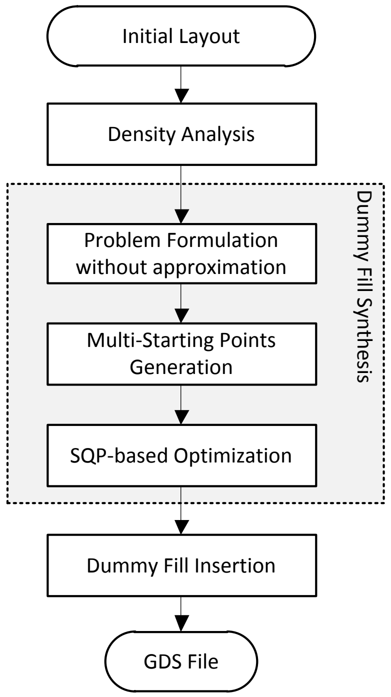

Figure 2: Framework of Dummy Filling Algorithm

3.1 Dummy Fill Synthesis

In this section, a SQP-based dummy fill synthesis method is proposed, where window density $\rho_{l,i,j}$ and slacks $s_{l,i,j}$ are used as inputs to determine the fill amount $x_{l,i,j}$ of each window with maximal $Quality$. Since the objective of optimization becomes more complicated, a linear approximation formulation can likely lead to a suboptimal solution. SQP method [1], known as one of the most popular algorithm for non-linear optimization, is applied directly in our framework for fill amount optimization. With SQP and error-free formulation of the original problem, perhaps, a better solution for dummy fill insertion can possibly be obtained.

We use the same objective proposed in ICCAD 2014 DFM contest benchmark as defined in (4b) and the problem can thus be formulated as

$$\min_{x_{l,i,j}} [g(\mathbf{x}) = g_{pd}(ov + fa) + g_{\sigma}(\sigma) + g_{\sigma^*}(\sigma^*) + g_{ol}(ol)] \tag{5a}$$

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

$$g_X(x) = \frac{\alpha_X}{\beta_X} \cdot x \tag{5c}$$

where $\mathbf{x}$ refers to the vector of fill amount $x_{l,i,j}$, and $g_X(x)$ is reformed from of $f_X(x)$ defined in (4b). Therefore, $Quality$ can reach its maximum as long as (5a) has its minimal value. The key point of this formulation is without any approximation and simplification compared with (4b).

Since there exist inequality constrains in (5), we use La-

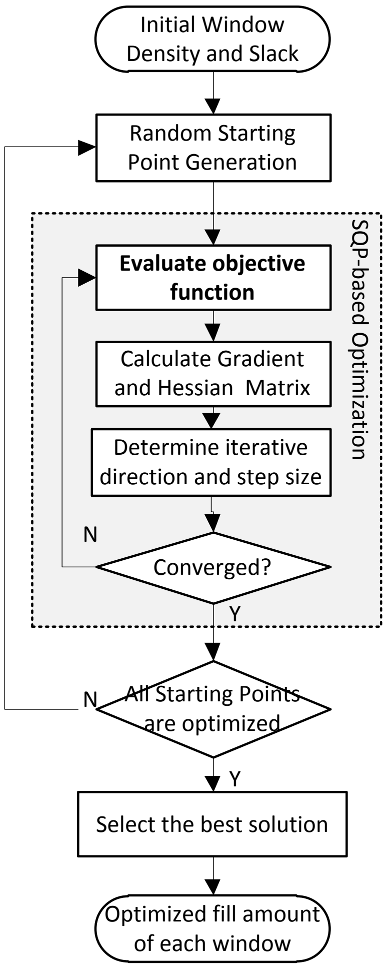
Initial Window Density and Slack
Random Starting Point Generation
SQP-based Optimization
Evaluate objective function
Calculate Gradient and Hessian Matrix
Determine iterative direction and step size
N
Converged?
Y
N
All Starting Points are optimized
Y
Select the best solution
Optimized fill amount of each window

Figure 3: Framework of MSP-SQP Algorithm

grangian function to merge these constrains into objective function, which is a general process of SQP for optimization with constrains[1]. The objective with merged constrains becomes

$$\min_{x_{l,i,j}} \mathcal{L}(\mathbf{x}, \mathbf{v_1}, \mathbf{v_2}) = g(\mathbf{x}) + \mathbf{v_1^T} \cdot (\mathbf{x} - \mathbf{s}) + \mathbf{v_2^T} \cdot (-\mathbf{x}) \quad (6)$$

where $\mathbf{s}$ refers to the vector of slack $s_{l,i,j}$; $\mathbf{v_1}$ and $\mathbf{v_2}$ refer to vectors of Lagrangian multipliers for constrain merging and they are iteratively calculated during SQP optimization.

After all inequality constrains are added to the original objective function, the optimized solution can possibly be found in the points where gradient of objective function is zero.

Moreover, since SQP algorithm can be trapped during the optimization by local optimal point, MSP technique is applied. Although SQP suffers from its extremely high runtime due to Hessian-matrix-related calculation, we can easily perform the parallel algorithm. The parallel performance of the algorithm is shown in Section 4.

Fig. 3 shows the basic flow of the MSP-SQP algorithm. The simplest strategy of generating starting points is by random methods, i.e. all initial $x_{l,i,j}$ are randomly and uniformly sampled in range $[0, s_{l,i,j}]$. From these starting points, the SQP optimization algorithm is followed, and finally select the best results as the output.

The standard SQP algorithm usually includes steps of iteratively evaluating objective function, calculating gradient and Hessian matrix, determining iterative direction and step size, and improving the results.

However, there is a big obstacle in this flow, in that all objective function can be easily evaluated, except the overlay area, i.e. $g_{pd}(ov)$ in (5a). The exact overlay area can be obtained only after the locations of all dummies are determined, which is performed in the final dummy insertion phase. If we have to perform dummy fill insertion in each iteration of SQP to determine locations of dummies and evaluate the overlay area, it will extremely obstruct algorithm efficiency. Fortunately, an accurate and efficient overlay area estimation method is proposed, which does not need the locations of dummies, and will be described in detail in Section 3.2.

Furthermore, the CMP simulator can be introduced in this framework during optimization process. Generally speaking, model-based approach can provide better solution for dummy fill insertion than rule-based one, and a model-based method based on our framework can possibly improve it further.

3.2 Overlay Area Estimation

Overlay area remains unknown until the locations of all dummy fills are determined. However, before dummy insertion, there are two methods to minimize the overlay area. One is slightly decreasing the fill amount, and the other is planning the places of dummies into different locations. The key idea of the latter is trying to divide the slack regions carefully into different types as shown in Fig. 4.

According to the signal wire locations in neighbouring layer, the slack regions of layer l can be divided into four types. Slack Type1 region in layer l refers to no signal wires in the same region in layer $l+1$ and $l-1$, slack Type2 refers to signal wires appearing only in layer $l+1$, Type3 refers to signal wires appearing only in layer $l-1$, and Type4 refers to signal wires appearing in both layers.

Then in each window $W_{l,i,j}$, the unknown fill amount $x_{l,i,j}$ also need to be divided into four variables $x^1_{l,i,j}$, $x^2_{l,i,j}$, $x^3_{l,i,j}$, and $x^4_{l,i,j}$ according to their different slack types. Although the extra three variables introduced in each window increases the complexity of optimization, such fillable region separation make overlay area estimation possible before dummy insertion.

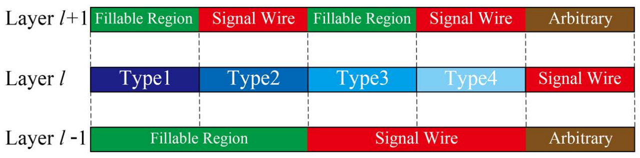

Figure 4: Four Types of Fillable Regions

Considering the fact that overlay only exists in the vertical direction, the overlay of a window *sandwich* at planar location (i, j), i.e., the overlay among $W_{l,i,j}, l = 1, ..., L$, can be estimated as follows.

Dummy-to-layout overlay area of a window sandwich can be observed from Fig. 5, where dummies filled in slacks Type2 and Type4 in $W_{l,i,j}$ are overlapped with signal wires

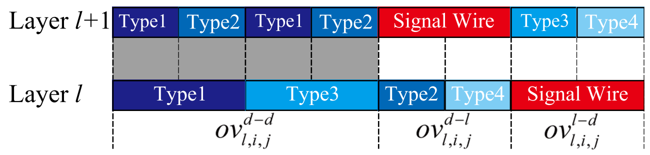

Figure 5: Overlay between Fills in $W_{l,i,j}$ and $W_{l+1,i,j}$

in $W_{l+1,i,j}$ and produce overlay area as

$$ov^{d-l}_{i,j} = \sum_{l=1}^{L-1} ov^{d-l}_{l,i,j} = \sum_{l=1}^{L-1} x^2_{l,i,j} + x^4_{l,i,j} \tag{7}$$

And *layout-to-dummy* overlay can be estimated in similar way,

$$ov^{l-d}_{i,j} = \sum_{l=1}^{L-1} ov^{l-d}_{l,i,j} = \sum_{l=1}^{L-1} x^3_{l+1,i,j} + x^4_{l+1,i,j} \tag{8}$$

The lack of $x^1_{l,i,j}$ in (7) and (8) means that no overlay is produced by dummies in slack Type1.

Though the initial patterns of given benchmark are black-boxes, we can still use the window densities as weighted coefficients to carefully estimate the *dummy-to-layout* overlay area as

$$ov^{d-l}_{l,i,j} = \rho^0_{l+1,i,j} \cdot (x^2_{l,i,j} + x^4_{l,i,j}) \tag{9a}$$

$$ov^{l-d}_{l,i,j} = \rho^0_{l,i,j} \cdot (x^3_{l+1,i,j} + x^4_{l+1,i,j}) \tag{9b}$$

where $\rho^0_{l,i,j}$ and $\rho^0_{l+1,i,j}$ are the given initial layout pattern densities of $W_{l,i,j}$ and $W_{l+1,i,j}$ respectively.

For *dummy-to-dummy* overlay area, however, it depends on fill amounts and locations of dummies on neighbouring layers, but it satisfies the following constraints

$$ov^{d-d}_{i,j} = \sum_{l=1}^{L-1} \max(0, ov^{d-d}_{l,i,j}) \tag{10a}$$

$$ov^{d-d}_{l,i,j} = (x^1_{l,i,j} + x^3_{l,i,j}) + (x^1_{l+1,i,j} + x^2_{l+1,i,j}) - s^*_{l,i,j} \tag{10b}$$

where $s^*_{l,i,j}$ refers to the shadow region area as shown in Fig. 5, i.e. the slack area of fillable regions interaction between $W_{l,i,j}$ and $W_{l+1,i,j}$. If arranging these dummies properly, it will be possible for $ov^{d-d}_{l,i,j}$ to be zero when total fill amount of both layers is not greater than $s^*_{l,i,j}$. However, if these dummies excess the slack regions $s^*_{l,i,j}$, there exists a minimal overlay at least. Those excessive dummies have to overlap with dummies located in the same region in adjacent layers.

Finally, the total overlay area can be estimated as (3a), and since slack region is separated into four types, the optimization problem can be reformed as

$$\min_{x^k_{l,i,j}} g_{pd}(ov + fa) + g_\sigma(\sigma) + g_{\sigma^*}(\sigma^*) + g_{ol}(ol) \tag{11a}$$

$$\text{s.t. } x^k_{l,i,j} \in [0, s^k_{l,i,j}] \tag{11b}$$

where $s^k_{l,i,j}$ refers to slack area of Type$k$ region in $W_{l,i,j}$. Therefore, random starting points should be generated by randomly determining all $x^k_{l,i,j}$ in corresponding range $[0, s^k_{l,i,j}]$. Furthermore, the meaning of $\mathbf{x}$ in (6) should be changed to the vector of fill amount in any type of region $x^k_{l,i,j}$ and $\mathbf{s}$ refers to the vector of slack $s^k_{l,i,j}$ now. The flow of optimization process keeps unchanged.

3.3 Prior Knowledge Based Starting Point Generation

Since layout is getting more and more complicated, the problem size of dummy fill insertion optimization becomes larger. As the result, optimization algorithm suffers a lot from its high runtime, especially when SQP method requires computing Hessian matrix in each iteration, and MSP strategy is applied. However, there are prior knowledges that we can apply for improving the efficiency of algorithm. In this subsection, a Prior Knowledge (PK) based starting point generation method is proposed which includes two separated phases, i.e., target density planning and overlay minimization with determined target fill amount. The two phases can be combined as a pre-optimized solution, and used as a good starting point for the following SQP optimization algorithm. This strategy can improve the runtime and effectiveness of the SQP sovler tremendously, though this prior knowledge only considers the objectives of planarity and overlay.

3.3.1 Target Density Planning

Target density planning is proposed by Lin[12] to maximize layout planarity regardless of any other objectives. Layout planarity is irrelevant between layers so each layer can be considered separately and processed among layers independently. A target layer density td_l is first determined in advance for each layer, which represents the expecting density of each window. After td_l is determined, a trivial solution to fill dummies for maximum density uniformity can be obtained,

$$x_{l,i,j} = \begin{cases} 0, & \text{if } td_l < \rho^0_{l,i,j} \\ s_{l,i,j}, & \text{if } td_l > A \cdot \rho^0_{l,i,j} + s_{l,i,j} \\ A \cdot (td_l - \rho^0_{l,i,j}), & \text{otherwise} \end{cases} \tag{12}$$

where A refers to the window area. Eq.(12) shows a simple strategy to minimize density variation, which is fill dummies to let window density as near as target layer density. Then the goal of this phase becomes finding the best target layer density for each layer.

For each window, the density will be in the interval $[\rho^0_{l,i,j}, \rho^0_{l,i,j} + s_{l,i,j}]$, if dummies are inserted. The best situation of target layer density td_l is when there exists a available value in all windows intervals in layer l. In this case, td_l can be determined as

$$td_l = \max(\rho^0_{l,i,j}),\ i \in [1..M],\ j \in [1..N] \tag{13}$$

Otherwise, a linear search of target layer density is performed on this layer and the solution with the best layout planarity is chosen. The search range is from $\max(\rho^0_{l,i,j})$ to $\min(\rho^0_{l,i,j} + s_{l,i,j})$ and the search step should be relatively small. Although the result of this phase is possibly not the optimal solution for maximizing layout planarity, it should be enough to generate a starting point.

3.3.2 Overlay Minimization in Window Sandwich

After target density is determined, i.e. the fill amount $x_{l,i,j}$ of window is obtained, the local optimization of overlay minimization is processed immediately. In this phase, the fill amount $x_{l,i,j}$ of each window should be assigned within four type of slack regions $s^k_{l,i,j}, k = 1, ..., 4$, for a minimal overlay area.

For each window sandwich (i, j), we try to arrange the dummies $x_{l,i,j}, l = 1..L$ properly to find the optimal fill amount $x_{l,i,j}^k, k = 1, ..., 4$ in different slack types for minimal overlay among this window sandwich. The formulation is

$$\min_{x_{l,i,j}^k} \quad ov_{i,j}^{d-l} + ov_{i,j}^{l-d} + ov_{i,j}^{d-d} \tag{14a}$$

$$\text{s.t. } x_{l,i,j}^k \in [0, s_{l,i,j}^k], \quad k \in 1, 2, 3, 4 \tag{14b}$$

$$x_{l,i,j} = \sum_{k=1}^{4} x_{l,i,j}^k \tag{14c}$$

The formulation is linear after removing the absolute operations in (14a), so local optimals of each segment are easily available. And the solution to (14) is the best one of these local optimals. Because the number of variables in each window sandwich is very small, this optimization problem can be solved quickly. Meanwhile, different window sandwich is totally independent, all window sandwich can be parallel solved.

Combined with the above two phases, a better starting point for dummy fill synthesis is obtained, which will be helpful for the following SQP optimization.

3.4 Dummy Fill Insertion

After optimizing the fill amounts in each slack type in each window, the dummy fill insertion phase will determine all positions of fills without violating DRC rules. The dummy fill insertion flow is briefly shown in Algorithm 1.

Algorithm 1 Dummy Fill Insertion

```
Input: Fill amount $x_{l,i,j}^k$ of each type in each window
Output: Locations of all dummies
1: Classify slack into four types
2: Cut the slack patterns into rectangles by the method
   proposed in [7].
3: Shrink slacks to satisfy all DRC rules
4: Insert dummies into each type of slacks of each $W_{l,i,j}$ in
   order of their areas
5: Insert a proper small dummy to minimize fill amount
   gap
```

4. EXPERIMENTAL RESULTS

We implement the proposed algorithm in C/C++ language. All the experiments of our algorithm are performed on a 2.67GHz Linux server with 64 CPU cores. In Table 2, the results proposed by [12] are tested on an 8-Core 3.4GHz Linux server, and the results of the top three of ICCAD 2014 contest are tested on a 2.6GHz computer. The coefficients to calculate objective function listed in Table 1 are exactly the same as the ones of ICCAD 2014 benchmark.

4.1 Results of Scores

The experimental results of our algorithm, the algorithm proposed in [12] and ICCAD 2014 contest top three are listed in Table 2.

In Table 2, Design lists layout names. Performance represents the score of overlay. Variation, Line Deviation and Outliers represent the score of density planarity. File Size is the score of file size. Quality is a weighted score represents the filling score including overlay, density planarity and file size, which is the same as definitions in (4b). Overall is the total weighted score including Quality, Runtime and Memory.

In these methods, the top three of ICCAD 2014 are list in first 3 rows. Lin is the algorithm proposed in [12]. MSP+SQP is the proposed method of SQP optimization algorithm with 20 random starting points. PKB is the results only after the prior knowledge based initial filling as proposed in section 3.3. PKB+SQP(8c) is the result of using PKB's result as a starting point and then optimizing it by SQP algorithm with 8 CPU Cores, and the 64 Cores results are also listed for scalability.

Table 2 shows that the fill quality of all our three proposed methods MSP+SQP, PKB and PKB+SQP are better than that of Lin's state-of-the-art method in all designs. Our best *Quality* scores is PKB+SQP, which is 8.16% higher than Lin's on average.

The PKB method gets the highest Overall scores, which is attributed to its extremely high run speed. The key point of the high speed of the PKB method is dividing the fill amount x into four type x^i based on the four different slack type, which only needs to solve many small linear optimization problems independently.

However in our opinion, the PKB method can still obtain the highest overall score for industry layouts is suspicious. A more robust strategy is taking the filling results of PKB method as a good starting point of SQP optimization algorithm. The optimization of SQP will improve the initial result to some extent, at least to a local optimization. As shown in Table 2, the results of PKB+SQP are always with higher quality than that of the PKB's.

Certainly, the trade-off is the more CPU time. Though the overall scores of PKB+SQP are less than that of PKB. However, for the big layout m with 2.2GB file size, the extra 18 min time for optimization is possibly affordable for a robust and better fill quality, and with the help of parallel computing the extra CPU time can be decreased to 6.5 min with 64 CPU cores.

MSP+SQP method costs the longest CPU time, because 20 random starting points are generated and then globally optimized by SQP method. In the biggest design *m*, MSP+SQP method needs about 12 hours. MSP+SQP's filling qualities, however, can still beat all results of the existing methods without any prior knowledge. This is extremely helpful for the unified dummy filling framework, in that if new optimization objectives are included and no prior knowledge methods are developed, it will be the only left method for such situations. Meanwhile, the unified dummy filling framework with random MSP strategy will become more feasible with the prevalence of high performance computation clusters, which makes the significance of well-designed objective-oriented methods weak to some extent.

4.2 Scalability of SQP

Furthermore, the SQP optimization is the most time-consuming process of SQP-based algorithms (MSP+SQP and PKB+SQP). However, it can be effectively accelerated by parallel computation. Fig. 6 shows the scalability of SQP on design *b*, where we can see that the speedup of SQP is roughly linear to CPU cores within maximal 64 cores. For the MSP part, the starting points are totally independent, so good scalability can easily be obtained for high performance computing platform.

Table 1: Coefficients of ICCAD 2014 Contest Benchmark

Design	#L	File Size	α_{PD}	β_{PD}	α_{σ}	β_{σ}	$\alpha_{\sigma*}$	$\beta_{\sigma*}$	α_{O}	β_{O}	α_{FS}	β_{FS}	α_{t}	β_{t}	α_{mem}	β_{mem}
s	3	48M	0.2	79154	0.2	0.077	0.2	11.758	0.15	0.014	0.05	32	0.15	60	0.05	1024
b	3	1.1G	0.2	6111303	0.2	0.517	0.2	3578	0.15	22.801	0.05	2048	0.15	600	0.05	32768
m	3	2.2G	0.2	10276835	0.2	0.53	0.2	6052	0.15	27.56	0.05	1536	0.15	1200	0.05	32768

Table 2: Experimental Results on ICCAD 2014 Contest Benchmark

Design	Team	Performance	Variation	Line Deviation	Outliers	File Size	Runtime	Memory	**Quality**	Overall
s	3rd	0.613	0.985	0.990	1.000	0.158	0.842	0.429	**0.676**	0.823
	2nd	0.743	0.909	0.967	0.975	0.103	0.846	0.831	**0.675**	0.844
	1st	0.743	0.636	0.733	1.000	0.976	0.877	0.885	**0.621**	0.797
	Lin	0.723	0.948	0.979	0.994	0.887	0.872	0.818	**0.724**	0.895
	MSP+SQP	0.725	1.000	1.000	1.000	0.792	0(6min)	0.805	**0.735**	0.775
	PKB	0.725	1.000	1.000	1.000	0.802	0.900	0.805	**0.735**	0.910
	PKB+SQP(8c)	0.725	1.000	1.000	1.000	0.802	0.687	0.805	**0.735**	0.878
	PKB+SQP(64c)	0.725	1.000	1.000	1.000	0.802	0.780	0.805	**0.735**	0.892
b	3rd	0.576	0.485	0.601	0.000	0.568	0.554	0.339	**0.361**	0.461
	2nd	0.841	0.381	0.534	0.000	0.053	0.513	0.828	**0.354**	0.472
	1st	0.748	0.368	0.364	0.871	0.924	0.515	0.891	**0.473**	0.594
	Lin	0.685	0.499	0.470	0.953	0.765	0.351	0.852	**0.512**	0.607
	MSP+SQP	0.410	0.851	0.872	0.867	0.756	0(14h)	0.847	**0.594**	0.637
	PKB	0.460	0.858	0.917	0.788	0.730	0.110	0.847	**0.601**	0.662
	PKB+SQP(8c)	0.480	0.848	0.911	0.783	0.761	0(22min)	0.847	**0.604**	0.646
	PKB+SQP(64c)	0.480	0.848	0.911	0.783	0.761	0.280(7.2min)	0.847	**0.604**	0.688
m	3rd	0.510	0.509	0.689	0.000	0.807	0.748	0.772	**0.382**	0.533
	2nd	0.668	0.460	0.618	0.000	0.000	0.780	0.761	**0.349**	0.504
	1st	0.598	0.462	0.486	0.204	0.941	0.556	0.845	**0.387**	0.513
	Lin	0.493	0.643	0.766	0.088	0.905	0.750	0.786	**0.439**	0.591
	MSP+SQP	0.677	0.543	0.790	0.000	0.901	0(12h)	0.771	**0.447**	0.489
	PKB	0.723	0.550	0.814	0.000	0.803	0.234	0.771	**0.458**	0.533
	PKB+SQP(8c)	0.716	0.553	0.825	0.000	0.837	0(18min)	0.771	**0.461**	0.499
	PKB+SQP(64c)	0.716	0.553	0.825	0.000	0.837	0.680(6.5min)	0.771	**0.461**	0.601

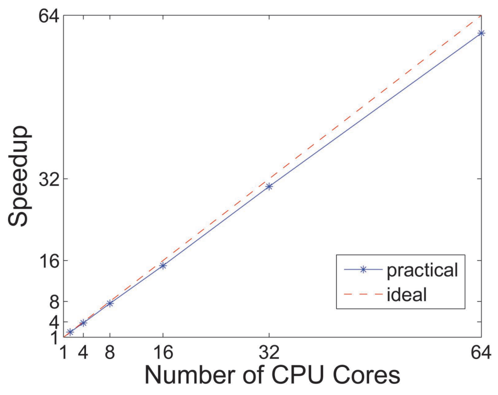

Figure 6: Scalability of SQP Method

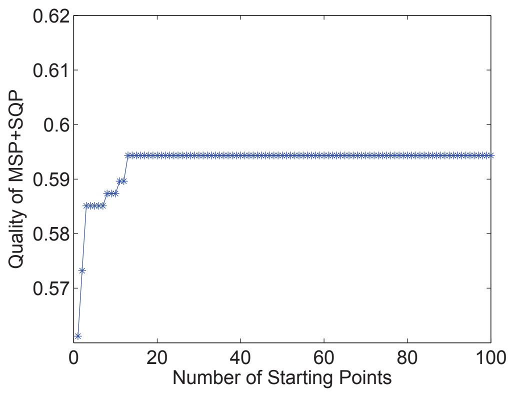

Figure 7: Number of Starting Points vs. Quality of MSP+SQP

4.3 Quality of MSP-SQP with Different Number of Starting Points

Although the MSP strategy and SQP optimization method are very suitable to the unified dummy filling framework, they suffers the long runtime. It is important to study how many starting points are needed. Fig. 7 shows the quality of MSP-SQP on design b with number of starting points, where x-axis is number of starting points and y-axis is the quality of MSP-SQP. From Fig. 7, we can see that MSP strategy can efficiently reach the high quality within only 15 random starting points.

5. CONCLUSIONS

In this paper, we propose a novel unified framework for dummy fill insertion, which is suitable for complicated optimization objectives. ICCAD 2014 contest benchmark is applied to verify the performance of our algorithm. Experimental results show that our algorithm is effective for multiple objectives for dummy fill insertion. To obtain solution with better quality, our algorithm requires more runtime than other proposed algorithm while the cost is affordable. A prior-knowledge-based starting point generation method is proposed to improve both performance and efficiency. Meanwhile, parallel computation can be easily applied to enhance the efficiency. In the future, we plan to improve the algorithm for run-time and consider more factors, such as lithography impacts and CMP model.

6. ACKNOWLEDGMENT

This research was supported partly by Key Technologies R&D Program 2016YFB0201304, partly by National Natural Science Foundation of China (NSFC) research project 61274032, 61376040, 61574046, and partly by NSF Grant 1115564 and 1218906.

7. REFERENCES

[1] P. T. Boggs and J. W. Tolle. Sequential quadratic programming. *Acta Numerica*, 4(1):1–51, January 1995.

[2] Y. Chen, P. Gupta, and A. B. Kahng. Performance-impact limited area fill synthesis. In *Proc. ACM/IEEE Des. Autom. Conf.*, pages 22–27, 2003.

[3] Y. Chen, A. B. Kahng, G. Robins, and A. Zelikovsky. Monte-carlo algorithms for layout density control. In *Proc. IEEE Asia South Pacific Des. Autom. Conf.*, pages 523–528, 2000.

[4] Y. Chen, A. B. Kahng, G. Robins, and A. Zelikovsky. Practical iterated fill synthesis for cmp uniformity. In *Proc. ACM/IEEE Des. Autom. Conf.*, pages 671–674, 2000.

[5] C. Feng, H. Zhou, C. Yan, J. Tao, and X. Zeng. Provably good and practically efficient algorithms for cmp dummy fill. In *Proc. ACM/IEEE Des. Autom. Conf.*, pages 539–544, 2009.

[6] C. Feng, H. Zhou, C. Yan, J. Tao, and X. Zeng. Efficient approximation algorithms for chemical mechanical polishing dummy fill. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 30(3):402–415, March 2011.

[7] K. D. Gourley and D. M. Green. Polygon-to-rectangle conversion algorithm. *IEEE Comp. Graphics & Applic.*, 3(1):31–36, January 1983.

[8] A. B. Kahng, G. Robins, A. Singn, and A. Zelikovsky. Filling algorithms and analyses for layout density control. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 18(4):1132–1147, October 2002.

[9] A. B. Kahng and K. Samadi. Cmp fill synthesis: A survey of recent studies. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 27(1):3–19, January 2008.

[10] A. B. kahng and R. O. Topaloglu. A doe set for normalization-based extraction of fill impact on capacitances. In *Proc. Int. Symp. Quality Electron. Des.*, pages 467–474, 2007.

[11] 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. Int. Symp. Quality Electron. Des.*, pages 373–376, 2003.

[12] Y. Lin, B. Yu, and D. Z. Pan. High performance dummy fill insertion with coupling and uniformity constraints. In *Proc. ACM/IEEE Des. Autom. Conf.*, pages 1–6, 2015.

[13] C. Liu, peishan Tu, P. Wu, H. Tang, Y. Jiang, J. Kuang, and E. F. Y. Young. An effective chemical mechanical polishing filling approach. In *Proc. IEEE Comput. Soc. Annual Sym. VLSI*, pages 44–49, 2015.

[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. IEEE Asia South Pacific Des. Autom. Conf.*, pages 1–6, 2007.

[15] R. Tian, D. F. Wong, and R. Boone. Model-based dummy feature placement for oxide chemical-mechanical polishing manufacturability. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 20(7):902–910, July 2001.

[16] R. O. Topaloglu. Energy-minimization model for fill synthesis. In *Proc. Int. Symp. Quality Electron. Des.*, pages 444–451, 2007.

[17] R. O. Topaloglu. Iccad-2014 cad contest in design for manufacturability flow for advanced semiconductor nodes and benchmark suite. In *Proc. IEEE Int. Conf. Comput.-Aided Des.*, pages 367–368, 2014.

[18] X. Wang, C. C. Chiang, J. Kawa, and Q. Su. A min-variance iterative method for fast smart dummy feature density assignment in chemical-mechanical polishing. In *Proc. Int. Symp. Quality Electron. Des.*, pages 258–263, 2005.

[19] H. Xiang, L. Deng, R. Puri, K.-Y. Chao, and M. D. F. Wong. Fast dummy-fill density analysis with coupling constraints. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 27(4):633–642, April 2008.