Monte-Carlo Methods For Chemical-Mechanical Planarization on Multiple-Layer and Dual-Material Models *

Yu Chen†, Andrew B. Kahng‡, Gabriel Robins§ and Alexander Zelikovsky¶

†Computer Science Department, UCLA, Los Angeles, CA 90095-1596
‡UCSD CSE and ECE Departments, La Jolla, CA 92093-0114
§Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442
¶Department of Computer Science, Georgia State University, Atlanta, GA 30303

ABSTRACT

Chemical-mechanical planarization (CMP) and other manufacturing steps in very deep submicron VLSI have varying effects on device and interconnect features, depending on the local layout density. To improve manufacturability and performance predictability, we seek to make a layout uniform with respect to prescribed density criteria, by inserting area fill geometries (dummy fill features) into the layout. We review previous research on single-layer fill for flat and hierarchical layout density control based on the Interlevel Dielectric CMP model. We also describe the recent combination of CMP physical modeling and linear programming for multiple-layer density control, as well as the Shallow Trench Isolation CMP model. Our work makes the following contributions for the Multiple-layer Interlevel Dielectric CMP model. First, we propose a new linear programming approach with a new objective for the multiple-layer fill problem. Second, we describe modified Monte-Carlo approaches for the multiple-layer fill problem. Comparisons with previous approaches show that the new linear programming method is more reasonable for manufacturability, and that the Monte-Carlo approach is efficient and yields more accurate results for large layouts. The CMP step in Shallow Trench Isolation (STI) is a dual-material polishing process, i.e., multiple materials are being polished simultaneously during the CMP process. Simple greedy methods were proposed for the non-linear problem with Min-Var and Min-Fill objectives, where the certain amount of dummy features are always added at a position with the smallest density. In this paper, we propose more efficient Monte-Carlo methods for the Min-Var objective, as well as improved Greedy and Monte-Carlo methods for the Min-Fill objective. Our experimental experience shows that they can get better solutions with respect to the objectives.

Keywords: Monte-Carlo, Greedy, CMP, Multiple-layer, STI

1. INTRODUCTION

Chemical-mechanical planarization (CMP) is an important step in the manufacturing of very deep submicron VLSI, and effects the overall production yield. In order to minimize the undesirable influence of local layout density variation on yield, we seek to improve the uniformity of the layout by inserting "dummy fill" features. That is, we introduce electrically inactive features into unused sparse areas in the original layout, as to increase the overall density.

Kahng *et al.*[7] first formulated the fill problem, developed layout density analysis algorithms, and gave a Linear-Programming based approach to the problem. In subsequent works, they also provided Monte-Carlo/Greedy methods[1] and iterated Monte-Carlo/Greedy methods[2] for the flat-layout fill problem, and proposed the hierarchical fill problem with corresponding methods.[3] Tian *et al.* pointed out the *Min-Fill* objective in the fill problem,[13] and proposed the methods for the dual-material fill problem.[14]

* This research was supported by a grant from Cadence Design Systems, Inc., by the MARCO/DARPA Gigascale Silicon Research Center, by a Packard Foundation Fellowship, by a National Science Foundation Young Investigator Award (MIP-9457412) by NSF grant CCR-9988331 and by the State of Georgia's Yamacraw Initiative.

Corresponding author: Yu Chen. Emails: Yu Chen (yuchen@cs.ucla.edu), Andrew Kahng (abk@ucsd.edu), Gabriel Robins (robins@cs.virginia.edu), Alexander Zelikovsky (alexz@cs.gsu.edu).

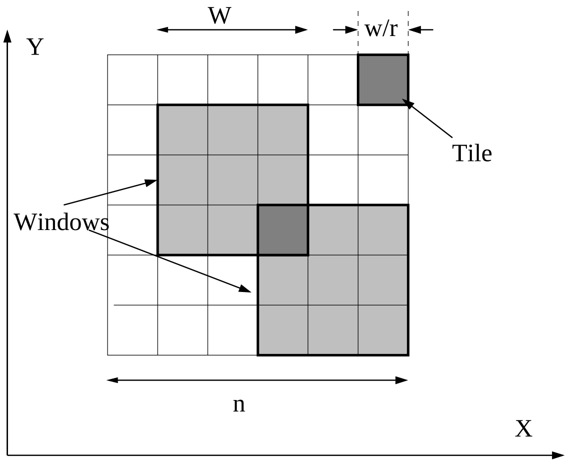

Figure 1. In the fixed r-dissection framework, the n-by-n layout is partitioned using r^2 (here, $r = 3$) distinct overlapping dissections, each with window size $w \times w$. Equivalently, the layout is partitioned into $\frac{nr}{w} \times \frac{nr}{w}$ *tiles*. Each dark-bordered $w \times w$ window consists of r^2 tiles.

All existing methods for fill synthesis are based on discretization: the layout is partitioned into *tiles*, and filling constraints or objectives (e.g., minimizing the maximum density variation) are enforced or pursued for square *windows* each consisting of $r \times r$ tiles. Thus, to practically control layout density in *arbitrary* windows, density bounds are enforced in only a *finite* set of windows. More precisely, both foundry rules and EDA physical verification and layout tools attempt to enforce density bounds within r^2 overlapping *fixed dissections*, where r determines the "phase shift" w/r by which the dissections are offset from each other. The resulting *fixed r-dissection* (see Fig. 1) partitions the $n \times n$ layout into tiles T_{ij}, then covers the layout by $w \times w$-windows W_{ij}, $i, j = 1, \ldots, \frac{nr}{w} - 1$, such that each window W_{ij} consists of r^2 tiles T_{kl}, $k = i, \ldots, i + r - 1$, $l = j, \ldots, j + r - 1$.

Two main filling objectives are considered in the recent literature:

- *(Min-Var Objective)* the *variation* in window density (i.e., maximum window density minus minimum window density) is minimized while the window density does not exceed the given upper bound U;
- *(Min-Fill Objective)* the number of inserted fill geometries is minimized while the density of any window remains in the given range (L, U).[†]

In this work we first concentrate on the Multiple-layer Interlevel Dielectric CMP (see Ref. 13) where density variation on the higher layers should take in account density variation on the underlying layer. We propose a new linear programming formulation for the multiple-layer fill problem and compare it with the previous formulations showing that the new objective is more suitable for manufacturability. We also develop a modification of the Monte-Carlo approach for multiple-layer fill. Our experiments show that this method is more efficient and yields more accurate results for large layouts. We next concentrate on the CMP step in the Shallow Trench Isolation (STI) model (see Refs. 4, 5, 14) where multiple materials are being polished simultaneously during the CMP process. The model is quite involved and essentially relies on nonlinear mathematical programming. The first attempt to solve the problem efficiently was the greedy algorithm which is known to be inefficient and unreliable since it tends to get easily trapped in local minima. In this paper we develop Monte-Carlo algorithm for the STI model, and experimentally show that it is not only considerably faster and more scalable, but also achieves better filling results. The remainder of this paper is organized as follows. Section 2 is devoted to the multiple-layer oxide CMP fill problem, where a new multiple-layer objective and the corresponding Linear Programming

[†]The Min-Var objective was introduced by Kahng *et al.*,[6] and captures the "manufacturing side" of the Filling Problem by seeking the most uniform density distribution possible. The Min-Fill objective was introduced by Tian *et al.*,[13] and captures the "design side" by seeking to minimize the total coupling capacitance and uncertainty caused by dummy fill. Minimizing dummy fill has the side benefit of reducing the complexity of the output GDSII.

formulations are proposed, and the Monte-Carlo method is extended to the multiple-layers. Section 3 reviews the current methods for the STI fill problem and propose new methods for it. Finally, computational results for multiple-layer Monte-Carlo and new STI fill methods are reported in Section 4.

2. MULTIPLE-LAYER OXIDE CMP DUMMY FILL

2.1. Density Models for Oxide CMP Dummy Fill

Several models for oxide planarization via CMP are reviewed in Ref. 8. In particular, the accurate and well accepted model of Ref. 11 is neither computationally expensive nor difficult to calibrate. In this model, the interlevel dielectric thickness z at location (x, y) is calculated as:

$$z = \begin{cases} z_0 - (\frac{K_i t}{\rho(x,y)}) & t < (\rho_0 z_1)/K_i \\ z_0 - z_1 - K_i t + \rho_0(x,y) z_1 & t > (\rho_0 z_1)/K_i \end{cases} \quad (1)$$

where K_i is the blanket polish rate, z_0 is the height of oxide deposition, z_1 is the height of existing features, t is the polish time, and ρ_0 is the initial pattern density. The crucial element of this model is the determination of the effective initial pattern density, $\rho(x, y)$.

We seek to understand how the effective density depends on the spatial pattern density distribution in a window. The simplest model for $\rho(x, y)$ is the local area feature density, i.e., the window density is simply equal to the sum:

$$\rho(W_{ij}) = \sum_{k=i}^{i+r-1} \sum_{l=j}^{j+r-1} area(T_{kl}) \quad (2)$$

where $area(T_{kl})$ denotes the original layout area of the tile T_{kl}. This *spatial density* model is due to Ref. 6, which solved the resulting filling problem using linear programming.

A more accurate model considers the deformation of the polishing pad during the CMP process[9]: effective local density $\rho(x, y)$ is calculated as the sum of *weighted* spatial pattern densities within the window, relative to an elliptical weighting function:

$$f(x, y) = c_0 \exp[c_1(x^2 + y^2)^{c_2}] \quad (3)$$

with experimentally determined constants c_0, c_1, and c_2.[13] The *discretized* effective local pattern density ρ for a window W_{ij} in the fixed-dissection regime (henceforth referred to as *effective density*) is:

$$\rho(W_{ij}) = \sum_{k=i}^{i+r-1} \sum_{l=j}^{j+r-1} area(T_{kl}) \cdot f(k - (i + r/2), l - (j + r/2)) \quad (4)$$

where the arguments of the elliptical weighing function f are the x- and y-distances of the tile T_{kl} from the center of the window W_{ij}.

2.2. Multiple-Layer Fill Problem

In a layout containing multiple layers, no layer except the bottom-most can assume a perfectly flat starting surface. Thus filling each layer optimally yet independently may result in an unacceptable planarization on the top layers when the layers are stacked together during the manufacturing process. This motivates the following:

The Multiple-Layer Filling Problem: Solve the Filling Problem for a given multiple-layer layout so that either:

- (Min-Var Objective) the sum of variations in window density on each layer is minimized, or the maximum variation in window density on each layer is minimized; or
- (Min-Fill Objective) the number of inserted fill geometries is minimized while the density of any window remains within the given range (L_k, U_k) for each layer k.

2.3. Fill Synthesis for Multiple-Layer Layout

In the model proposed in Ref. 16, topographic variation of one layer attenuates through the subsequent CMP processes, each of which is modeled as a low-pass filter based on equations (3) and (4), according to the following equation:

$$\hat{\rho_{0(k)}} = \begin{cases} [\hat{d_k} + (\frac{z_{k-1}}{z_k})\hat{\rho_{0(k-1)}}] \times \hat{f} & k > 1 \\ \hat{d_1} \times \hat{f} & k = 1 \end{cases} \quad (5)$$

where " ˆ " is the Fast Fourier Transform (FFT) operator, $\hat{\rho_{0(k)}}$ is the effective local density, z_k is the step height (i.e., the height of layer k from the first layer), d_k is the local density, all for layer k, and f is the weighting function. In the discussion below, we will not explicitly address the multiple-layer model. However, our linear programming and Monte-Carlo algorithms have straightforward extensions for simultaneously handling multiple layers. By mathematical induction on the layer number k and the linearity of Fourier transforms, Equation 5 can be written as[13]:

$$\hat{\rho_{0(k)}} = \sum_{l=1}^{k} [(z_l/z_k)\hat{f}^{k-l+1} \times \hat{d_l} \quad (6)$$

Furthermore, for effective density at a location (i,j) on layer k, each term in the summation results in a multiple circular convolution in the physical domain:

$$\begin{aligned} [IFFT(\hat{f}^{(\alpha)} \times \hat{d_l})](i,j) &= [\overbrace{(f \otimes f \cdots f)}^{\alpha} \otimes d_l](i,j) \\ &= \sum_{i_1}\sum_{j_1}[f(i_1 - i, j_1 - j) \times \cdots \\ &\quad (\sum_{i_\alpha}\sum_{j_\alpha}[f(i_\alpha - i_{\alpha-1}, j_\alpha - j_{\alpha-1}) \times (x_{i_\alpha j_\alpha l} + x^0_{i_\alpha j_\alpha l})])] \end{aligned} \quad (7)$$

Since the multiple convolution written as a series of summations is linear in term of x_{ijk}, the LP formulations for single-layer fill problem can be easily extended for multiple layers.

2.3.1. Linear Programming Approaches for Multiple Layers

Tian *et al.*[13] extended the linear programming formulations (LP0) to address multiple layers with the objective of minimizing the sum of density variation over all layers:

Minimize: $\sum_{k=1}^{K}(H_k - L_k)$

Subject to:

$$0 \le L_k \le \rho_0(i,j,k) \le H_k \le \sum_{l=1}^{k}(\frac{z_l}{z_k}) \quad i,j = 0,\ldots,\frac{nr}{w}-1, k = 1,\ldots,K \quad (8)$$

$$0 \le x_{ijk} \le slack(T_{ijk}) \quad i,j = 0,\ldots,\frac{nr}{w}-1, k = 1,\ldots,K \quad (9)$$

Considering only the sum of variations on all layers in the objective function can not guarantee that the filling on each layer will satisfy the Min-Var objective. It is possible that the density variation of one of the layers will be too large to be manufacturable. Moreover, a bad polishing result on an intermediate layer due to nonplanarization can potentially cause problems on later (upper) layers. So it is more reasonable to ensure that the density variation on each layer is actually less than the bound required by the CMP process.

Based on these considerations, we propose a new linear program formulations (LP1) for multiple-layer fill with the objective of minimizing the maximum density variation across all layers:

Minimize: M

Subject to:

$$0 \le L_k \le \rho_0(i,j,k) \le H_k \le \sum_{l=1}^{k}(\frac{z_l}{z_k}) \quad i,j = 0,\ldots,\frac{nr}{w}-1, k = 1,\ldots,K \tag{10}$$

$$(H_k - L_k) \le M \quad k = 1,\ldots,K \tag{11}$$

$$0 \le x_{ijk} \le slack(T_{ijk}) \quad i,j = 0,\ldots,\frac{nr}{w}-1, k = 1,\ldots,K \tag{12}$$

2.3.2. Monte-Carlo Approaches for Multiple-Layer Fill

As pointed out in Ref. 1, LP-based methods have two main drawbacks: (1) solving a large problem is too time consuming, and (2) the rounding error will worsen the solution quality. In this section, we address the multiple-layer fill problem using new Monte-Carlo based approaches.

```
Multiple-Layer Monte-Carlo Fill Algorithm
-----------------------------------------
Input: layout with multiple layers, and dummy feature size
Output: layout with multiple filled layers with respect to the minimized
        sum of density variations of all layers, or other objectives
-----------------------------------------
For L= bottomLayer To topLayer Do
   For each tile[i][j] on layer L Do
      Compute its slack area slackAreaOfTile[L][i][j]
      and cumulative effective density mlEffDenOfTile[L][i][j]
      mlSlackAreaOfTileStack[i][j] += slackAreaOfTile[L][i][j]
      mlEffDenOfTileStack[i][j] += mlEffDenOfTile[L][i][j]
   End For
End For
Compute the priority of tile stacks according to mlEffDenOfTileStack[i][j]
While (sum of priorities > 0) Do
// Iteratively insert dummy features into tiles randomly according to a certain priority scheme
   Randomly select a tile stack TS(i,j) according to its priority
   For L = bottomLayer to topLayer Do
      If slackAreaOfTile[L][i][j] > 0 Then
         For every neighboring tile[L][m][n] located in (L + 1) × (L + 1) square Do
            If the insertion on tile[L][i][j] causes the neighboring tile meet the upperbound Then
               Exit loop
            End If
         End For
         Insert the fill feature into tile[L][i][j]
         Update the slackAreaOfTile[L][i][j] and mlSlackAreaOfTile[i][j]
         Update the priorities
         Exit loop
      Else
         Lock the tile on layer L
      End If
   End For
End While
```

Figure 2: Multiple-layer Monte-Carlo Fill Algorithm.

The Monte-Carlo method for the CMP fill problem was first introduced in Ref. 1. For the single-layer fill problem, the algorithm randomly chooses a tile according to its priority value, and increments its content (i.e., density) by a prescribed fill amount. For the multiple-layer fill problem where the objective is to minimize the sum of density variations on all layers, we define the *tile stack* as the column of tiles having the same position on all the layers. Similarly, we define the density of a tile stack as the sum of densities of all the tiles in that tile stack.

The Monte-Carlo algorithm (see Fig. 2) takes the original maximum density to be the upper bound for each layer. It then randomly chooses a tile stack according to its priority value, selects a layer in the stack, and increments the tile's density and the tile stack's density by a prescribed fill amount, assuming that the insertion is permitted with respect to the objective.

The probability of choosing a particular tile stack TS_{ij} is referred as the *priority* of that tile stack. Note that the priority of a tile stack TS_{ij} is zero if and only if either TS_{ij} has already achieved the density upper bound U, or the slack of TS_{ij} is less than the prescribed fill area. Tiles with zero priority are said to be *locked.* Following Ref. 1, the priority of a tile stack TS_{ij} is chosen to be proportional to *U-EffDen(*TS_{ij}*)*, where *EffDen(*TS_{ij}*)* is the effective density of the tile stack TS_{ij}.

During the algorithm's execution, after choosing which tile stack to fill next, we also need to decide into which layer the dummy features should be inserted. We consider three different ways to choose the insertion layer. The first method is to choose the bottom layer first, and then try an upper layer if the current layer is not feasible. The second method is to select the top layer first, and then try a lower layer if the current layer is not feasible. For these two approaches, once no layer is suitable for fill, the tile stack will be "locked" and will not be subsequently selected for any more filling. The third approach is to randomly choose one layer for fill, and then to try the upper layer or the lower layer with equal probability. Here, a tile stack is "locked" when no slack area remains in it. Our experimental results indicate that the second method described above outperforms the other two approaches.

A variant of the Monte-Carlo approach is the deterministic *Greedy* algorithm. Unlike the Monte-Carlo approach, at each step the Greedy algorithm adds a prescribed amount of fill into a tile with the highest priority. The experiments show that the run times of this Greedy approach are slightly higher than Monte-Carlo's, due to the requirement of finding a tile with the highest priority, rather than a random tile.

As analyzed in Ref. 2, we can also implement the Iterated Monte-Carlo and Greedy methods for the multiple-layer fill problem, which alternating the Min-Var and Min-Fill objectives, resulting in a monotonic narrowing of the density variation (see Fig. 3). Such iterated methods are still very fast and retain all the advantages of the non-iterated Monte-Carlo and Greedy counterparts, but offer improved accuracy.

```
Iterated Monte-Carlo and Greedy Filling Algorithms
Input: n × n multiple-layer layout with l layers, fixed r-dissection, w × w window,
          density upper bound U_l on each layer
Output: Filled multiple-layer layout
Repeat forever
    Run Min-Var Monte-Carlo (Greedy) Algorithm with the upper window densities U_l
    If resulting sum of density variations equals the previous solution Then Exit repeat
    Else
      While there exist an unlocked tile stack Do
         Choose an unlocked tile stack TS_ij randomly, according to its priority
         Choose a layer
         If the deletion does not deteriorate the solution Then
            Delete a dummy feature from the layer
         Else
            Lock the tile stack
            Update the priorities of the tile stacks
         End If
      End While
    End If
End Repeat
```

Figure 3: The Iterated Monte-Carlo and Greedy multiple-layer fill approach.

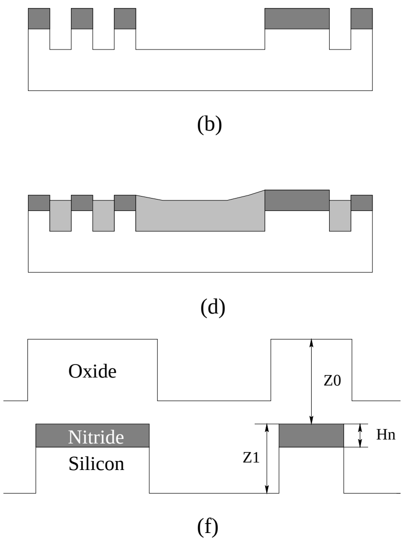

Figure 4. Shallow Trench Isolation process and parameters. (a) nitride deposition; (b) etch; (c) oxide deposition; (d) chemical-mechanical polishing; (e) nitride stripping; (f) STI parameters.

3. SHALLOW TRENCH ISOLATION DUMMY FILL

In integrated circuits, devices such as transistors are fabricated on a common piece of silicon. Without proper device isolation, these devices may interact with one another in undesirable ways. The need for a scalable CMOS isolation technology is critical in order to advance into the 0.25 m 256Mbit DRAM generation and beyond. The geometric characteristics of ideally scalable isolation technology are (i) an abrupt transition from active MOSFET regions to isolation regions, (ii) independence of isolation width and depth, and (iii) planarity.[4] Shallow Trench Isolation (STI) becomes the isolation technique of choice because it leads to higher device density. The process sequences for STI is the following. First, a thin pad oxide (200-600 Å) is thermally grown on the silicon wafer followed by the deposition of a relatively thick silicon nitride layer (1500-2000 Å). The nitride film is then patterned and etched in the isolation regions. A thin thermal oxide film is then grown to passivate the side-walls before filling the trenches with oxide. The deposited film is then polished until the nitride, which acts as a polish stop, is exposed. The nitride and the underlying pad oxide are then etched to expose the active device areas (see Fig. 4).

3.1. STI CMP Model

The basic idea of Stine's model[11] for oxide CMP is that when polishing the initial raised areas of the layout, the polishing rate of the up area is inversely proportional to the effective feature density. The down areas are not polished at all while there exist local step height differentials. Once the up areas are completely removed, the down areas begin to be polished. However, local pad compression has been observed in recent experiments, where at a certain step height, the pad touches both the top areas and the down areas, at which point the polishing rate changes. The model proposed by Grillaert[5] shows that the polishing rates for both the top and bottom areas of the step are:

$$\frac{dH_u}{dt} = \begin{cases} -K/\rho & t < t_c \\ -K - (1-\rho)\frac{h_c}{\tau}e^{-(t-tc)/\tau} & t > t_c \end{cases} \qquad \frac{dH_d}{dt} = \begin{cases} 0 & t < t_c \\ -K + \rho\frac{h_c}{\tau}e^{-(t-tc)/\tau} & t > t_c \end{cases} \tag{13}$$

where the contact height h_c is the local step height at which the pad starts to touch the down area of the steps, t_c is the contact time, K is the blanket polishing rate, ρ is the local effective density, and $\tau = \frac{\rho h_c}{K}$. Smith[10] pointed out that h_c is a function of density and space and can be calculated as $h_c = a_1 + a_2 e^{-\rho/a_3}$.

The model for a dual-material polishing was first studied by Grillaert[5] for STI, and was extended to copper CMP by Tugbawa *et al.*[15]:

$$dH_u/dt = -K_u[1 + \frac{H_s}{h_c}\frac{1-\rho}{\rho}] \qquad dH_d/dt = -K_d[1 - \frac{H_s}{hc}] \tag{14}$$

where $H_s = H_u - H_d$, K_u is the blanket polishing rate for the material at the top areas, and K_d is the blanket polishing rate for the bottom areas. Equation (14) assumes that the pad is touching both surfaces, i.e., $H_s \leq h_c$.

3.2. STI Dummy Fill Problem

Based on the above model, Tian *et al.*[14] derived the following equations for the STI CMP process under the assumption that the oxide profile is deposited conformally onto the underlying trench profile, and the side walls of both the deposited oxide and the underlying trenches are straight. Some STI parameters: oxide deposition thickness (z_0), total initial step height (z_1), and the thickness of nitride film (H_n), are shown in Fig. 4 (f). The height of oxide surface after $t > t_c$ is:

$$H_{ox}(\rho, t) = z_0 - K_{ox}t - (1-\rho)h_c(z_1/hc - e^{-(t-tc)/\tau}) \tag{15}$$

where K_{ox} is the blanket polishing rate for oxide, and the contact time is calculated as $t_c = \rho(z_1 - h_c)/K_{ox}$.

Let t_1 be the time at which all the oxide above the nitride layer has been polished, that is, $H_{ox}(\rho, t_1) = 0$. At $t = t_1$, the local step height is $h_i = h_c e^{-(t_1 - t_c)/\tau}$.

After that, both nitride atop silicon and oxide in the trenches are polished together. Integration of Eq. (14) with the initial condition $H_u(t=0) = H_n$ gives the nitride surface height as:

$$H_{ni}(\rho, t) = H_n - (1 + \frac{(1-\rho)K_\delta\tau_2}{\rho h_c}K_{ni}t - \frac{1-\rho)K_\delta\tau_2}{\rho h_c}(h_i - K_\delta\tau_2)(1 - e^{-t/\tau_2}) \tag{16}$$

where K_{ni} is the blanket polishing rate for nitride, $\tau_2 = (\frac{K_{ox}}{h_c} + \frac{K_{ni}(1-\rho)}{h_c\rho})^{-1}$, and $K_\delta = K_{ox} - K_{ni}$.

Let t_2 be the time at which the nitride layer is cleared. Because different locations may reach the end point in nitride at different times, Tian *et al.* proposed the nonlinear equation for the height difference between a location (i, j) with the minimum density and a location (i', j') with the maximum density as

$$\Delta H(i,j,i',j') = H_n - (1 + \frac{(1-\rho(i',j'))K_\delta\tau_2}{\rho(i',j')h_c})K_{ni}\Delta t(i,j,i',j') - \frac{1-\rho(i',j')K_{ni}\tau_2}{\rho(i',j')h_c}(h_i - K_\delta\tau_2)(1 - e^{-\Delta t(i,j,i',j')/\tau 2}) \tag{17}$$

where $\Delta t(i,j,i',j')$ is the nitride polishing time of location (i',j') as $\Delta t(i,j,i',j') = t_1(i,j) + t_2(i,j) - t_1(i',j')$.

3.3. Monte-Carlo and Greedy Approaches for STI Fill

Equation (17) shows that the STI post-CMP variation can be controlled by changing the feature density distribution using dummy features insertion.[14] The STI Fill problem then seeks to minimize the maximum height variation ΔH for the *Min-Var* objective, or else to minimize the total amount of inserted fill, while respecting the given lower bound for the *Min-Fill* objective. Tian *et al.*[14] formulated the problem as the non-linear programming problem and proposed Greedy methods for the *Min-Var* and *Min-Fill* objectives. In their Min-Var Greedy methods, a dummy feature is always added at a location having the smallest effective density. The iteration terminates when there is no feasible fill position left in the layout. For the Min-Fill objective, this method concludes once the given bound for ΔH is satisfied. Obviously, this *Min-Var* Greedy method is not guaranteed to find a global minimum. Actually, our computational experience shows that it is difficult for the simple Greedy method to avoid a local minima, since it is deterministic. Also, for the *Min-Fill* Greedy method, simple termination when the bound is first encountered is not sufficient to yield optimal or even near-optimal solutions.

The Monte-Carlo algorithm for the Min-Var objective, first introduced in Ref. 1, iterates by randomly choosing a tile and incrementing its effective density by a prescribed fill amount. The probability of choosing a particular tile T_{ij} is referred to as the *priority* of that tile. Both the Monte-Carlo and Greedy algorithms are suboptimal for the Min-Var objective, and although they are both fast in practice, the resulting density variation may be significantly larger than the optimum. *Iterated* methods proposed in Ref. 2 result in a monotonic narrowing of the gap between the upper density bound U and the minimum density L. Such iterated methods are still very fast and retain all the advantages of the non-iterated Monte-Carlo and Greedy counterparts, yet offer improved accuracy (see Table 5).

For the *Min-Fill* objective, our method first finds a solution that optimizes the *Min-Var* objective or satisfies the given lower bound, and then modifies the solution with respect to the *Min-Fill* objective. Thus, the first objective (density variation) can hopefully be traded off towards a significant improvement in the second objective (the amount of inserted fill). To implement this strategy with either the Monte-Carlo or Greedy methods, we use the *Fill-Removal* phase to iteratively delete an inserted dummy feature from a tile, randomly chosen according to its priority. It is natural to choose a priority symmetrical to the priority in the Min-Var Monte-Carlo algorithm. Thus, the Min-Fill Monte-Carlo/Greedy algorithm deletes fill geometries from unlocked tiles which are randomly chosen according to the priority scheme, i.e., proportional to $\Delta H - \Delta H(i, j, i_{max}, j_{max})$. No filling geometry can be deleted from a tile T_{ij} (i.e., T_{ij} is *locked*) if and only if it either has zero priority or else all fill previously inserted into T_{ij} has been deleted (see Fig. 5).

STI Min-Fill Monte-Carlo/Greedy Algorithm
Input: $n \times n$ filled layout, fixed r-dissection, $w \times w$ window, height difference bound ΔH
Output: Filled layout with minimized amount of inserted fill area satisfying the bound ΔH

Run the Min-Var Monte-Carlo/Greedy algorithm
Compute the removal priority of each tile
While there exist an unlocked tile **Do**
    Choose an unlocked tile T_{ij} randomly, according to its priority
    Delete a fill geometry from T_{ij}
    Update the tile priorities
End While
Output resulting layout

Figure 5: The STI Min-Fill Monte-Carlo/Greedy Algorithm.

4. COMPUTATIONAL EXPERIENCE

In this section, we compare the performance of the Monte-Carlo methods and Greedy methods on the multiple-layer and STI fill problems. All experiments in this work are performed using part of a metal layer extracted from an industry standard-cell layout[‡] (Table 1). Benchmark L1 is the M2 layer from an 8,131-cell design; benchmark L2 is the M3 layer from a 20,577-cell design; benchmark L3 is the M2 layer from the same 20,577-cell design as L2; benchmark L4 consists of two metal layers from an 8,131-cell design; and benchmark L5 and L6 consists of two metal layers from the same 20,577-cell design as L2.

Table 1: Parameters of four industry test cases. In this coordinate system, 40 units is equivalent to 1 micron.

testcase	Single-layer Test Cases			Multiple-layer Test Cases		
test case	L1	L2	L3	L4	L5	L6
layout size n	125,000	112,000	112,000	125,000	112,000	112,000
# rectangles k	49,506	76,423	133,201	99,012	209,624	152,846

[‡]Our experimental testbed integrates GDSII Stream input, conversion to CIF format, and internally-developed geometric processing engines, coded in C++ under Solaris. We use CPLEX version 7.0 as the linear programming solver. All runtimes are CPU seconds on a 300 MHz Sun Ultra-10 with 1GB of RAM.

4.1. Multiple-layer Fill

Table 2 shows the performances of the two LP formulations proposed by Tian *et al.* and by us respectively for the different multiple-layer fill objectives. The experiments indicate that the LP formulations with objective to minimize the sum of density variations on all layers can not also minimize the maximum density variation at the same time.

Table 2. The performance of the LP formulations under the objectives of minimizing (i) the sum of density variations, and (ii) the maximum density variation, on all layers. **Notation:** *L/W/r*: layout / window size / r-dissection; *LP0*: the linear programming formulations with objective to minimize the sum of density variations on all layers; *LP1*: the linear programming formulations with objective to minimize the maximum density variation across all layers; *SumVar*: the sum of density variations on all layers; *maxDenVar*: the maximum density variation across all layers; *CPU*: the run time; *Area*: the number of inserted dummy features.

test case	**LP0**				**LP1**			
L/W/r	SumVar	maxDenVar	CPU	Area	SumVar	maxDenVar	CPU	Area
L4/16/4	0.2690	0.1696	42.0	20921	0.2875	0.1666	37.6	19609
L4/16/8	0.6626	0.4696	44.2	14769	0.6626	0.4696	43.2	14330
L4/8/4	0.6626	0.4696	42.7	14769	0.6626	0.4696	43.1	14330
L4/8/8	0.9031	0.6346	157.3	9959	0.9455	0.6255	137.7	10133
L5/16/4	0.3436	0.2420	101.0	38152	0.3843	0.1932	69.8	38241
L5/16/8	1.0585	0.5531	283.3	34942	1.0621	0.5393	671.9	33376
L5/8/4	1.0585	0.5531	279.7	34942	1.0621	0.5393	655.7	33376
L6/16/4	0.5986	0.4080	91.3	65578	0.6333	0.3737	71.0	62113
L6/16/8	1.6116	1.1155	13322.0	67178	1.6584	1.0903	6622.0	65576
L6/8/4	1.6116	1.1155	12617.0	67178	1.6584	1.0903	6649.0	65576

Table 3 compares the sum of density variations on all layers and the associated run times for the linear programming method (LP0), Greedy method, Monte-Carlo (MC) method, Iterated Greedy (IGreedy) method and Iterated Monte-Carlo (IMC) method. Our results show that the accuracy of the Monte-Carlo/Greedy methods is very high. When the window size is small and/or the number of fixed dissections is large, the LP method becomes impractical for multiple-layer fill problem[§], while the Monte-Carlo/Greedy methods are still fast. On the other hand, the rounding error inherent in the LP method make its performance worse than the Monte-Carlo/Greedy methods on the large test cases.

Table 3. The performance of LP0, Greedy, MC, IGreedy and IMC for the sum of density variations across all layers. **Notation:** *L/W/r*: layout / window size / r-dissection; *SumVar:* the sum of density variations across all layers; *CPU:* the run time. The data in bold denotes the best results.

test case	**LP0**		**Greedy**		**MC**		**IGreedy**		**IMC**	
L/W/r	SumVar	CPU	SumVar	CPU	SumVar	CPU	SumVar	CPU	SumVar	CPU
L4/16/8	0.6626	**33.1**	0.6420	36.3	**0.6285**	36.6	**0.6285**	37.7	**0.6285**	33.9
L4/16/5	**0.5435**	30.7	0.5541	32.2	0.5535	33.0	0.5535	31.1	0.5535	**30.5**
L4/8/8	0.9031	140.1	0.7794	48.1	0.7766	36.2	**0.7762**	74.7	**0.7762**	**34.5**
L4/8/5	0.8351	33.4	0.7882	35.4	**0.7804**	32.7	**0.7804**	39.1	**0.7804**	**30.7**
L5/8/8	2.2118	8093.0	**2.0526**	102.8	2.0913	**65.4**	**2.0526**	111.7	2.0716	67.6
L5/8/5	1.3494	8879.0	1.3450	65.0	1.3943	**54.2**	**1.3252**	79.6	1.3476	59.3

Table 4 shows the performance of the Linear Programming method (LP1), Greedy method, Monte-Carlo method, Iterated Greedy method and Iterated Monte-Carlo method with respect to the maximum density

[§]For example, the LP0 method does not report results for the test case L6/8/5 after running for more than 12 hours.

variation objective across all layers. The Monte-Carlo and Greedy methods yield better solutions than LP on these test cases within shorter run times.

Table 4. The performance of LP1, Greedy, MC, IGreedy and IMC on the maximum density variation across all layers. **Notation:** *L/W/r*: layout / window size / r-dissection; *MaxDen:* the maximum density variation on all layers; *CPU:* the run time. The data in bold denotes the best results.

test case	**LP1**		**Greedy**		**MC**		**IGreedy**		**IMC**	
L/W/r	MaxDen	CPU	MaxDen	CPU	MaxDen	CPU	MaxDen	CPU	MaxDen	CPU
L4/16/8	0.4696	34.8	0.4459	36.3	**0.4454**	36.6	**0.4454**	37.7	**0.4454**	**33.9**
L4/16/5	0.3638	36.5	0.3638	**30.2**	**0.3635**	33.0	**0.3635**	31.1	**0.3635**	32.5
L4/8/8	0.6255	120.8	0.5437	48.1	0.5410	36.2	**0.5406**	74.7	**0.5406**	**34.5**
L4/8/5	0.5897	33.2	0.5576	35.4	**0.5497**	32.7	**0.5497**	39.1	**0.5497**	**30.7**
L5/8/8	1.2174	761.3	**1.1081**	102.8	1.1089	**65.4**	**1.1081**	111.7	**1.1081**	67.6
L5/8/5	0.6886	524.0	0.6857	65.0	0.7050	**54.2**	**0.6698**	79.6	0.6746	59.3

4.2. STI Fill

Our experiments for the STI fill problem ran on metal layers from industrial designs, with parameters typical of industry STI processes. We compare the performances of different Monte-Carlo and Greedy methods for both the Min-Var and Min-Fill objectives. Table 5 shows the post-fill layout height difference from the Greedy, Monte-Carlo, Iterated Greedy, and Iterated Monte-Carlo methods. The data indicates that for the Min-Var objective, the performance of the Monte-Carlo (MC), Iterated Greedy (IGreedy) and Iterated Monte-Carlo (IMC) methods are all better than the simple Greedy approach. For example, The Iterated Monte-Carlo method can improve over the simple Greedy approach by 30% for L1/32/8.

Table 5. Methods for Shallow Trench Isolation (STI) Fill under the Min-Var objective. **Notation:** *orig*ΔH: the original height difference of the layout; ΔH: the post-fill height difference of the layout; *L/W/r*: layout / window size / r-dissection; *CPU*: the run time. The data in bold denote the best results.

test case		**Greedy**		**MC**		**IGreedy**		**IMC**	
L/W/r	orig ΔH	ΔH	CPU	ΔH	CPU	ΔH	CPU	ΔH	CPU
L1/32/4	695.2	305.3	3.2	335.0	3.2	304.5	3.5	**290.2**	3.4
L1/32/8	999.6	426.1	3.8	325.8	3.4	407.4	7.1	**307.2**	4.2
L2/28/4	801.8	487.9	5.1	374.0	5.2	487.9	5.7	**348.0**	5.6
L2/28/8	1124.6	569.8	5.7	536.1	5.2	546.3	10.1	**482.7**	6.6
L3/28/4	1095.2	577.8	8.5	563.2	8.3	577.8	8.9	**563.2**	9.1

The performances of the simple Greedy Min-Fill method, Monte-Carlo Min-Fill method, Greedy Min-Fill method with a fill removal phase, and Monte-Carlo Min-Fill method with a fill removal phase are compared in Table 6. Our results indicate that the simple Greedy Min-Fill method does not yield near-optimal results. For example, the solutions of the simple Greedy Min-Fill method for test case L2/28/8 are at least 50% away from optimal, as compared with the results from the Monte-Carlo method with a fill removal phase.

5. CONCLUSION AND FUTURE RESEARCH

In this paper, we reviewed recent works on the multiple-layer and STI fill problems. For the multiple-layer fill problem, we presented a new linear programming approach with a new objective, as well as modified Monte-Carlo approaches. For the STI fill problem, we addressed the shortcomings of the current methods and proposed Monte-Carlo/Greedy methods for both the Min-Var and the Min-Fill objectives. Our experiments indicate that the Monte-Carlo methods are efficient for the multiple-layer fill problem, and that the improved STI methods can yield better solutions with respect to the objectives we studied.

Table 6. Methods for Shallow Trench Isolation (STI) Fill with Min-Fill objective. **Notation:** *orig*ΔH: the original height difference of the layout; *final*ΔH: the post-fill height difference of the layout; *GreedyI*: the Greedy method, which terminates after satisfying the given height difference bound; *MCI*: the Monte-Carlo method, which terminates after satisfying the given height difference bound; *GreedyII*: the Greedy method with a fill removal phase; *MCII*: the Monte-Carlo method with a fill removal phase; *L/W/r*: layout / window size / r-dissection; *Area*: the amount of inserted dummy features; *CPU*: the run time. The data in bold denote the best results.

test case			GreedyI		MCI		GreedyII		MCII	
L/W/r	orig ΔH	final ΔH	Area	CPU	Area	CPU	Area	CPU	Area	CPU
L1/32/4	695.2	395.1	10336	3.1	12003	3.1	**8962**	3.2	9141	3.2
L1/32/8	999.6	462.7	22091	3.9	20679	3.4	15615	3.6	**14754**	3.4
L2/28/4	801.8	526.2	7491	4.8	15164	4.9	7593	5.1	**6543**	5.1
L2/28/8	1124.6	639.8	16808	5.7	26114	5.5	8367	5.9	**7142**	5.5
L3/28/4	1095.2	563.2	24274	8.2	27114	8.0	16628	8.6	**16142**	8.5

Ongoing research includes further study of the multiple-layer fill objectives, and more powerful Monte-Carlo methods for the multiple-layer fill problem. We are also pursuing a CMP simulation tool to check the feasibility of our STI fill methods.

ACKNOWLEDGMENTS

The authors thank Prof. D. F. Wong and Mr. Ruiqi Tian for their help in obtaining the STI CMP parameters.

REFERENCES

1. Y. Chen, A. B. Kahng, G. Robins and A. Zelikovsky, "New Monte-Carlo Algorithms for Layout Density Control", *Proc. ASP-DAC*, 2000, pp. 523-528.
2. Y. Chen, A. B. Kahng, G. Robins and A. Zelikovsky, "Practical Iterated Fill Synthesis for CMP Uniformity", *Proc. Design Automation Conf.*, Los Angeles, June 2000, pp. 671-674.
3. Y. Chen, A. B. Kahng, G. Robins and A. Zelikovsky, "Hierarchical Dummy Fill for Process Uniformity", *Proc. ASP-DAC*, Jan. 2001, pp.139-144.
4. Terence Gan, "Modeling of Chemical Mechanical Polishing for Shallow Trench Isolation", *M.S. Thesis*, Massachusetts Institute of Technology, May 2000.
5. J. Grillaert, "A Study of the Planarization Process during Chemical-Mechanical Polishing for Oxides and Shallow Trench Isolation", *Ph. D. Dissertation*, Catholic University, Leuven, Belgium, May 1999.
6. A. B. Kahng, G. Robins, A. Singh, H. Wang and A. Zelikovsky, "Filling Algorithms and Analyses for Layout Density Control", *IEEE Trans. Computer-Aided Design* 18(4), 1999, pp. 445-462.
7. A. B. Kahng, G. Robins, A. Singh, H. Wang and A. Zelikovsky, "Filling and Slotting: Analysis and Algorithms", *Proc. ACM/IEEE Intl. Symp. on Physical Design*, Apr. 1998, pp. 95-102.
8. G. Nanz and L. E. Camilletti, "Modeling of Chemical-Mechanical Polishing: A Review", *IEEE Trans. on Semiconductor Manufacturing* 8(4), 1995, pp. 382-389.
9. D. Ouma, D. Boning, J. Chung, G. Shinn, L. Olsen, and J. Clark, "An Integrated Characterization and Modeling Methodology for CMP Dielectric Planarization", *International Interconnect Technology Conference*, San Francisco, CA, June 1998.
10. T. Smith, "Device Independent Process Control of Dielectric Chemical Mechanical Polishing", *Ph. D. Dissertation*, MIT, Cambridge, MA, 1999.
11. B. Stine, "A Closed-Form Analytical Model for ILD Thickness Variation in CMP Processes", *Proc. CMP-MIC*, 1997.
12. B. Stine, D. Ouma, R. Divecha, D. Boning and J. Chung, "Rapid Characterization and Modeling of Pattern Dependent Variation in Chemical Mechanical Polishing", *IEEE Trans. Semi. Manuf.*, Feb. 1998.
13. R. Tian, D. Wong, and R. Boone, "Model-Based Dummy Feature Placement for Oxide Chemical-Mechanical Polishing Manufacturability", *Proc. Design Automation Conf.*, June 2000, pp. 667-670.
14. R. Tian, X. Tang and D. F. Wong, "Dummy feature placement for chemical-mechanical polishing uniformity in a shallow trench isolation process ", *International Symposium on Physical Design*, Apr. 2001, pp. 118-123.
15. T. Tugbawa, T. Park, D. Boning, T. Pan, P. Li, S. Hymes, T. Brown, and L. Camilletti, "A Mathematical Model of Pattern Dependence in Cu CMP Process", *CMP Symposium in Electrochemical Society Meeting*, Honolulu, HI. 1999, pp. 605-615.
16. T. Yu, S. Cheda, J. Ko, M. Roberton, A. Dengi, and E. Travis, "A Two-Dimensional Low Pass Filter Model for Die-Level Topography Variation Resulting from Chemical Mechanical Polishing of ILD Films", 1999 *IEDM*, Washington, DC, Dec. 1999.