Practical Iterated Fill Synthesis for CMP Uniformity *

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

UCLA Computer Science Dept., Los Angeles, CA 90095-1596
†Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442
‡Department of Computer Science, Georgia State University, Atlanta, GA 30303
{yuchen, abk}@cs.ucla.edu, robins@cs.virginia.edu, alexz@cs.gsu.edu

Abstract

We propose practical *iterated methods* for layout density control for CMP uniformity, based on linear programming, Monte-Carlo and greedy algorithms. We experimentally study the tradeoffs between two main filling objectives: minimizing density variation, and minimizing the total amount of inserted fill. Comparisons with previous filling methods show the advantages of our new iterated Monte-Carlo and iterated greedy methods. We achieve near-optimal filling with respect to each of the objectives and for both density models (*spatial* density [3] and *effective* density [8]). Our new methods are more efficient in practice than linear programming [3] and more accurate than non-iterated Monte-Carlo approaches [1].

1 Introduction

As the manufacturing process increasingly constrains physical layout design and verification [5], one particular requirement is to control manufacturing variation due to *chemical-mechanical polishing* (CMP) [4] [6] [9]. CMP variation is kept within acceptable limits by controlling local feature density, relative to a process-specific "window size" on the order of 1-3mm. Layout Density Control consists of two phases: *density analysis* and *fill synthesis*. Density analysis determines the area available for filling. Fill synthesis computes the amount of feature area which should be added into each part of the layout to achieve uniformity, then generates the actual fill geometries. In this paper, we address fill synthesis.

The Filling Problem: Given a design rule-correct layout in an $n \times n$ layout region, along with window size $w < n$, add fill geometries to create a *filled layout* such that either:

- *Min-Var Objective:* the *variation* in window density (maximum window density minus minimum window density) is minimized while no window density exceeds the given upper bound U; OR
- *Min-Fill Objective:* the number of inserted fill geometries is minimized while the density of any window remains between the given lower bound L and upper bound U.

* 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), and by a GSU Research Initiation Grant.

The Min-Var objective, first introduced in [3], captures the "manufacturing" objective of the most uniform density distribution. The Min-Fill objective, recently proposed in [8], captures the "design" objective of minimizing coupling capacitance and uncertainty caused by filling.[1] Our present work:

- gives a unified description of two existing models for CMP, which allows us to apply both new and previous methods to compute the filling area regardless of how the effective density [8] is defined;
- proposes new and *iterated* methods for solving the Filling Problem which: (1) improve the existing methods simultaneously in accuracy and running time; (2) efficiently address both *spatial* [3] and *effective* [8] density models; (3) are applicable to both the Min-Var and Min-Fill objectives; (4) are easily modifiable to take into consideration many practical issues such as the shape of filling geometries and various design rules; and (5) exhibits scalable behavior while giving results as good as or better than all previous methods under all known metrics.

2 Unified Description of Models for the CMP

Several models for oxide planarization via CMP have been proposed in [6]. Among them, the model of [7] is neither computationally expensive nor difficult to calibrate. In this model, the interlevel dielectric thickness z at location (x,y) is calculated as:

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

The crucial element of this model is the determination of the effective initial pattern density $\rho(x,y)$.

To make the filling problem more tractable, a standard practice is to consider only a finite set of layout windows. Bounding the effective density in a fixed set of $w \times w$ windows can incur substantial error, since other windows could still violate the density bounds. A common industry practice is to enforce density bounds in r^2 overlapping *fixed dissections*, where r determines the "phase shift" w/r by which the dissections are offset from each other. In other words, to help control layout density in arbitrary windows, density bounds are enforced only for windows of the *fixed r-dissection* (see Figure 1), which 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}, $k = i, \ldots, i+r-1$, $l = j, \ldots, j+r-1$.

[1] With ungrounded fill features, existing parasitic extraction tools do not handle the resulting floating capacitances well. To minimize coupling capacitance and performance uncertainty caused by filling, the objective becomes to minimize the total number of fill geometries (with the side benefit of reducing complexity of final GDSII output).

The simplest model for $\rho(x,y)$ is the local areal 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 model is due to [3], which solved the filling problem using linear programming.

A more accurate model considers the deformation of the polishing pad during the CMP process [2]: 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 [8]. The *discretized* effective local pattern density ρ for a window W_{ij} in the fixed-dissection regime (henceforth referred to as *effective window 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}.

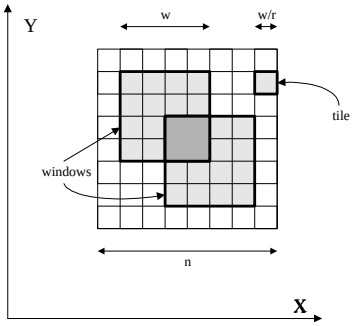

Figure 1: The layout is partitioned by r^2 ($r = 4$) distinct dissections (each with window size $w \times w$), into $\frac{nr}{w} \times \frac{nr}{w}$ *tiles*. Each dark-bordered $w \times w$ window consists of r^2 tiles.

3 Previous Approaches to Filling Synthesis

3.1 Linear Programming Approaches

The linear programming approach seeks the optimum fill area $p(T_{ij})$ to be inserted into each tile T_{ij}. Recall that the fill area $p(T_{ij})$ cannot exceed $slack(T_{ij})$, which is the area available for filling inside the tile T_{ij} computed during density analysis.

- The first LP for the Min-Var objective [3] is:

  Maximize: M

  subject to:

  $$0 \le p(T_{ij}) \le slack(T_{ij})$$

  $$M \le \rho(M_{ij}) \le U \quad i,j = 1,\ldots,\frac{nr}{w}-1$$

- A followup work[8] proposed the Min-Fill objective, along with a *Ranged Variation* LP:

  Minimize: $\sum_{i,j} p(T_{ij})$

  subject to:

  $$0 \le p(T_{ij}) \le slack(T_{ij})$$

  $$L \le \rho(M_{ij}) \le U \quad i,j = 1,\ldots,\frac{nr}{w}-1$$

- We also note a variant LP for the Min-Var objective, that is, given a target window density M (instead of an upper bound on window density), we minimize the variability budget ε:

  Minimize: ε

  subject to:

  $$0 \le p(T_{ij}) \le slack(T_{ij})$$

  $$M - \varepsilon/2 \le \rho(T_{ij}) \le M + \varepsilon/2 \quad i,j = 1,\ldots,\frac{nr}{w}-1$$

The number of variables and the number of constraints in these LPs are both bounded by $O((\frac{nr}{w})^2)$. Although LP solutions are optimal, there are several drawbacks: (1) solving a very large LP (with large r) is too time consuming (e.g., the run time is $O(r^6)$ since the number of variables in the LP is $O(r^2)$); (2) the optimal solution for an r-dissection is not necessarily the optimal solution for a $2r$-dissection and may also result in a high *floating* window density variation, (i.e., density variation over all windows, not only over the fixed-r dissection); and (3) rounding is another source of errors in LP formulations (when the tile size is sufficiently small, the problem becomes an instance of integer programming and rounding errors become crucial).

3.2 Monte-Carlo Approaches

The Monte-Carlo method for the Min-Var objective was introduced in [1]. The Min-Var Monte-Carlo algorithm randomly chooses a tile and increments its content (i.e., spatial/effective density) by a prescribed fill amount. The probability of choosing a particular tile T_{ij} is referred as the *priority* of that tile. Note that the priority of a tile T_{ij} is zero if and only if either T_{ij} belongs to a window which has already achieved the density upper bound U, or the slack of T_{ij} is equal to the already-inserted fill area. Tiles with zero priority are said to be *locked*. Following [1], the priority of a tile T_{ij} is chosen to be proportional to $U - MinWin(T_{ij})$, where $MinWin(T_{ij})$ is the minimum density over windows containing the tile T_{ij}. (Experimentally, this priority scheme outperforms a number of other simple schemes[1].) The Monte-Carlo method is very efficient: it can be implemented within $O(\frac{nr}{w}\log\frac{nr}{w})$ time, which is practically proportional to the number of tiles. Its main drawback is that it may insert an excessive amount of total fill.

A variant of the Monte-Carlo approach is the deterministic *Greedy algorithm*. At each step the Min-Var Greedy algorithm adds the maximum possible amount of fill into a tile with the highest priority. In other words, at each step a tile with the highest priority is locked. The performance of the Min-Var Greedy algorithm is illustrated in Table 2. Greedy run times are slightly higher than Monte-Carlo because of finding highest-priority rather than random tiles.

4 New Approaches to Filling with the Min-Fill Objective

In the presence of two objectives, a natural strategy is first to find a solution that optimizes one of the objectives (Min-Var), and then modify that solution with respect to the other objective (Min-Fill), hopefully without degrading the solution quality relative to the first

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). This strategy can be implemented with the LP-based method as follows:

1. Solve the Min-Var LP formulation with the given upper bound U on window density;
2. Decrease the obtained minimum window density M by a given amount $L = M(1-\varepsilon)$;
3. Solve the Min-Fill LP formulation within the interval (L,U).

To implement the same strategy with either the Monte-Carlo or greedy approach, we assume that the density of each window is already within the interval (L,U) and then solve the following:

Fill-Deletion Problem (with the Min-Fill objective): delete as much previously inserted fill as possible, while maintaining a minimum window density of no less than L.

To solve the Fill-Deletion problem with the Monte-Carlo approach, we iteratively delete a filling geometry from a tile randomly chosen according to a certain priority. It is natural to choose priority symmetrical to the priority in the Min-Var Monte-Carlo algorithm, i.e., proportional to $MinWin(T_{ij}) - L$. Again symmetrically, no filling geometry can be deleted from the 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.

Thus, the Min-Fill Monte-Carlo algorithm deletes fill geometries from unlocked tiles which are randomly chosen according to the above priority scheme (see Figure 2). Similarly, the Min-Fill Greedy algorithm iteratively deletes a filling geometry from an unlocked tile with the currently highest priority.

Min-Fill Monte-Carlo Algorithm
Input: $n \times n$ filled layout, fixed r-dissection, $w \times w$ window, lower bound on window density L
Output: Filled layout with minimized amount of inserted fill area
While there exist an unlocked tile **do** Choose an unlocked tile T_{ij} randomly, according to its priority Delete a filling geometry from T_{ij} Update priorities of tiles Output resulting layout

Figure 2: The Monte-Carlo Algorithm for the Fill-Deletion Problem deletes fill geometries from randomly chosen unlocked tiles (i.e., tiles which still have filling geometries but which belong to windows having density greater than L).

5 Iterated Monte-Carlo and Greedy Methods

Min-Var Objective: As mentioned above, both the Monte-Carlo and Greedy Algorithms are suboptimal for the Min-Var Objective, and although they are both fast in practice, the resulting minimum window density may be significantly lower than the optimum. We now propose *iterated* methods based on alternating the Min-Var and Min-Fill objectives (see Figure 3), resulting in a monotonic narrowing of the gap between the upper window density bound U and the minimum window density L. 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 (see Table 2).

Min-Fill Objective: To solve the Filling Problem with the Min-Fill Objective, the Iterated Monte-Carlo and Greedy Filling algorithms (see Figure 3) may be modified as follows:

Iterated Monte-Carlo and Greedy Filling Algorithms
Input: $n \times n$ layout, fixed r-dissection, $w \times w$ window, upper bound on window density U
Output: Filled layout
Repeat forever Run Min-Var Monte-Carlo (Greedy) Algorithm with the upper window density U If resulting minimum window density equals the previous M then exit repeat Update the densities of tiles and windows and the minimum window density M Run Min-Fill Monte-Carlo (Greedy) Algorithm with the lower window density M

Figure 3: In the Iterated Monte-Carlo and Greedy Filling approach, each iteration consists of two applications (with the Min-Var and Min-Fill objectives) of the Monte-Carlo and Greedy algorithms.

1. Interrupt the filling process as soon as the lower bound L on window density is reached, i.e., when $M = L$, instead of improving the minimum window density (while possible) for the Min-Var objective.
2. Continue iterating, but without changing the lower density bound $M = L$. Although this does not guarantee that the total filling area will not increase, an improved solution can typically be obtained if we will keep track of the best solution seen over all iterations.

6 Experimental Results

Test Cases				
Test case	L1	L2	L1x4	L2x4
layout size n 40 units = 1 μ	125,000	112,000	250,000	224,000
# rectangles k	49,506	76,423	198,024	305,692

Table 1: Parameters of four industry test cases.

Our experiments were performed using part of a metal layer extracted from an industry standard-cell layout[2] (Table 1). Benchmark L1 is the M2 layer from an 8,131-cell design, and Benchmark L1x4 is the same layout replicated four times in a 2x2 array to create a larger test case. Benchmark L2 is the M3 layer from a 20,577-cell layout. L2x4 is this layout replicated four times in a 2x2 array. Our implementation of layout density control is enhanced with the following important practical features:

- *Grid slack computation:* In previous academic and industry approaches, the area slack in each tile (i.e., the area available for filling), was assumed to be proportional to the real empty space in the tile. Our alternative grid slack computation entails using an underlying grid to compute the maximum number of legal positions for fill geometries in each tile. This method of slack calculation is more realistic, and guarantees that the calculated amount of fill can actually be legally inserted into the corresponding tile.
- *Doughnut area computation:* In shallow-trench isolation processes, so-called reverse active-area mask steps lead to a density criterion whereby only the width-d "outer ring" of a large feature contributes to the effective density. Our tool optionally applies such "doughnut" area computations.
- *Wraparound window density analysis and synthesis:* Because of the way dies are arrayed on a wafer, we make windows "wrap around" the layout during density analysis as well as

[2]Our experimental testbed integrates GDSII Stream input, conversion to CIF format, and internally-developed geometric processing engines, coded in C++ under Solaris. All run times are reported in CPU seconds on a 140MHz Sun Ultra-1 with 256MB of RAM. All experiments assume that U is equal to the maximum window density of the original layout.

Spatial Density Model												
	Orig. Density		**LP**		**Greed**		**MC**		**IGreed**		**IMC**	
Test case	Max	Min	Min	CPU	Min	CPU	Min	CPU	Min	CPU	Min	CPU
L1/32/8	0.21447	0.10414	0.19864	41.5	0.18779	18.2	0.19221	17.3	0.19871	26.9	0.19871	24.8
L1/32/16	0.21783	0.10088	0.19768	1077.5	0.19044	21.9	0.19410	19.6	0.19779	98.1	0.19740	93.5
L1/16/8	0.26452	0.07803	0.17519	161.1	0.17556	21.8	0.17556	18.9	0.17556	36.7	0.17556	30.2
L1/16/16	0.26452	0.08551	*0.17169*⋆	N/A	0.18868	44.2	0.18868	23.4	0.18868	202.3	0.18868	168.9
L2/32/8	0.22648	0.07039	0.14467	43.0	0.14257	25.5	0.13565	24.4	0.14469	41.3	0.14463	68.6
L2/32/16	0.22648	0.07650	0.15093	2716.0	0.14621	33.8	0.14459	29.4	0.14971	538.5	0.14940	317.2
L2/16/8	0.33022	0.04552	0.17926	1912.4	0.16709	42.1	0.17748	30.5	0.17980	170.1	0.17980	169.4
L1x4/32/8	0.21693	0.09657	0.18643	255.7	0.18183	82.6	0.18282	72.3	0.18648	131.9	0.18648	111.9
L1x4/32/16	0.21793	0.10263	*0.18448*⋆	N/A	0.19574	124.3	0.19547	80.2	0.19933	632.8	0.19933	565.1
L2x4/32/8	0.22226	0.05776	0.14647	532.6	0.14480	150.7	0.13824	117.7	0.14649	289.5	0.14655	469.7
Effective Density Model												
L1/32/8	0.41625	0.16255	0.31970	32.4	0.31859	22.8	0.31994	22.3	0.31994	26.5	0.31994	23.9
L1/32/16	0.46662	0.10626	0.28249	105.5	0.28353	27.4	0.28353	24.0	0.28353	33.2	0.28353	27.8
L1/16/8	0.46662	0.10626	0.28249	105.2	0.28353	27.1	0.28353	23.1	0.28353	32.8	0.28353	26.0
L1/16/16	0.48313	0.05693	*0.13285*⋆	N/A	0.24748	49.7	0.24748	27.1	0.24748	74.2	0.24748	33.4
L2/32/8	0.53585	0.07249	0.34777	66.8	0.34538	39.7	0.31153	38.3	0.34629	49.5	0.33858	68.9
L2/32/16	0.84446	0.03514	0.35956	520.5	0.36007	57.4	0.34049	41.4	0.36007	67.9	0.35276	107.4
L2/16/8	0.84446	0.03514	0.35956	526.7	0.36007	57.3	0.34206	40.1	0.36007	68.7	0.35120	90.1
L1x4/32/8	0.43270	0.14665	0.28487	171.5	0.28505	107.2	0.28505	90.7	0.28505	126.5	0.28505	100.9
L1x4/32/16	0.46740	0.10494	0.28732	1238.8	0.28835	177.0	0.28835	106.3	0.28835	262.4	0.28835	125.5
L1x4/16/8	0.46740	0.10494	0.28732	1387.8	0.28835	188.8	0.28835	106.3	0.28835	266.6	0.28835	121.5
L1x4/16/16	0.48313	0.05160	*0.19859*⋆	N/A	0.27197	586.0	0.27197	119.5	0.27197	975.0	0.27197	150.1
L2x4/32/8	0.52179	0.04467	0.34176	637.4	0.32008	241.5	0.30799	165.6	0.33620	342.0	0.33524	435.9

Table 2: The Iterated Greedy (IGreed) and Iterated Monte-Carlo (IMC) algorithms are more accurate than the non-iterated versions (Greed and MC), and are faster than a linear program-based approach (LP).

in fill synthesis, so that windows overlapping with the upper (right) edge of the chip layout also contain the corresponding tiles from the bottom (left) edge of the layout.

- *Different pattern types:* In order to reduce worst-case coupling capacitance to fill, we impose a uniformity constraint on the fill pattern so that same amount of fill pattern area is intersected by any vertical or horizontal line. To this end, we use a basket-weaving pattern suggested in [3]. Our implementation can also support more exotic fill pattern types.

Table 2 compares the minimum window density and the associated run times for the minimum variation linear program, Greedy algorithm, Monte-Carlo algorithm, Iterated Greedy and Iterated Monte-Carlo algorithms. The table consists of two parts, corresponding to the spatial and effective density measures, respectively. The left column of Table 2 gives for each test case the window size (in thousands of units), as well as the number r of fixed dissections.

The smaller r-value corresponds to the maximal value for which the LP approach can still give the optimal minimum window density within a reasonable run time, and the larger r-value is selected sufficiently high to demonstrate the accuracy of the suggested heuristics. The next two table columns report the maximum and minimum window densities of the original layout before filling.

Table 2 indicates that the iterated methods are more accurate than previous non-iterated approaches, that they are more efficient than LP-based methods, and that they offer more even filling or larger number of tiles (corresponding to larger r). Finally, note that the iterated Monte-Carlo and Greedy algorithms can output better solutions than LP-based approaches, since the LP's rounding error becomes more significant for larger r.

7 Conclusion

We have presented a new unified approach to capturing different models of Layout Density Control for CMP. This enables us to apply Greedy and Monte-Carlo methods that simultaneously address different filling objectives for spatial and effective density definitions. Our new iterated Greedy and Monte-Carlo methods are more accurate and practical than previous linear-program based methods. Our implementation of the proposed methods incorporates several practical features, and improves on the present generation of available tools for layout density control.

Our ongoing research addresses the following two applications of the suggested methods:

- *Hierarchical Filling*: The iterated Monte-Carlo method can substantially improve the efficiency of filling hierarchical designs, and significantly decrease the size of the filled layout output file, resulting in a better integration into available hierarchical physical verification engines.

- *Multi-Layer Density Control*: A more accurate model (recently suggested by the authors of [8]) takes into account the influence of density variation in lower layers on density variation in upper layers. Thus, the methods developed in our paper seem more attractive than traditional LP approaches, due to the drastic increase in the number of variables.

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] R. R. Divecha, B. E. Stine, D. O. Ouma, J. U. Yoon, D. S. Boning, et al., "Effect of Fine-line Density and Pitch on Interconnect ILD Thickness Variation in Oxide CMP Process", *Proc. CMP-MIC*, 1998.

[3] 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.

[4] H. landis, P. Burke, W. Cote, W. Hill, C. Hoffman, et al., "Integration of Chemical-Mechanical Polishing into CMOS Integrated Circuit Manufacturing", *Thin Solid Films* 220(20) (1992), pp. 1-7.

[5] W. Maly, "Moore's Law and Physical Design of ICs", (special address), *Proc. ISPD*, 1998.

[6] G. Nanz and L. E. Camilletti, "Modeling of Chemical-Mechanical Polishing: A Review", *IEEE Trans. on Semiconductor Manufacturing* 8(4) (1995), pp. 382-389.

[7] B. Stine, "A Closed-Form Analytical Model for ILD Thickness Variation in CMP Processes", *Proc. CMP-MIC*, 1997.

[8] R. Tian, D. Wong, R. Boone and A. Reich, "Dummy Feature Placement for Oxide Chemical-Mechanical Polishing Manufacturability", *Tech Rep. 9-19, University of Texas at Austin CS Dept.*, 1999.

[9] M. Tomozawa, "Oxide CMP Mechanisms", *Solid State Technology* 40(7) (1997), pp. 169-175.