New Multilevel and Hierarchical Algorithms for Layout Density Control*

Andrew B. Kahng, Gabriel Robins†, Anish Singh† and Alexander Zelikovsky

UCLA Department of Computer Science, Los Angeles, CA 90095-1596
†Department of Computer Science, University of Virginia, Charlottesville, VA 22903-2442
{abk,alexz}@cs.ucla.edu, {robins,as6f}@cs.virginia.edu

Abstract

Certain manufacturing steps in very deep submicron VLSI involve chemical-mechanical polishing (CMP) which has varying effects on device and interconnect features, depending on local layout characteristics. To reduce manufacturing variation due to CMP and to improve yield and performance predictability, the layout needs to be made uniform with respect to certain density *criteria, by inserting "fill" geometries into the layout. This paper presents an efficient multilevel approach to density analysis that affords user-tunable accuracy. We also develop exact fill synthesis solutions based on combining multilevel analysis with a linear programming approach. Our methods apply to both flat and hierarchical designs.*

1 Introduction

VLSI technology is entering ever deeper into submicron regimes, as predicted by SIA's National Technology Roadmap for Semiconductors [8]. The manufacturing process is thus having an increasingly constraining effect on physical layout design and verification, and manufacturing costs tend to increasingly drive design [6]. To maximize yield, devices and interconnect must be manufactured in a predictable and uniform manner with respect to a total *variability budget* distributed among the design attributes. As the interactions between design and manufacturing become tighter, many traditional manufacturing problems can be (partially) solved at the design level, when more effective solutions may be applied.

Attaining large process windows and uniform manufacturing characteristics is difficult since these properties depend on "global" relationships among layout features, not only the traditional local ones. Hence, the traditional way of representing manufacturing constraints as design rules is often not appropriate. In particular, this is true when we need to address the problem of controlling manufacturing variation due to *chemical-mechanical polishing* (CMP) [5] [7] [9], the procedure by which wafers are polished using a rotating pad and slurry to achieve the planarized surfaces on which succeeding processing steps can build.

CMP variation can be controlled if the *local feature density* is controlled, where the definition of "local" is determined by the *length scale* at which feature density impacts oxide thickness, and corresponds to the "window size" within which feature density must be controlled. For oxide CMP, this length scale has been estimated to be on the order of 1-3mm, depending on CMP pad material, slurry composition, etc. [2].[1]

To minimize the impact of manufacturing process physics on device yield, foundries typically impose *density* rules[2] for features on active and metal layers, so that the layout becomes more uniform. Many process layers, including diffusion and even thin-ox, can have associated density rules that are satisfied with layout post-processing that adds *fill* geometries. Traditionally, only foundries or specialized TCAD tools companies have performed the post-processing of layout needed to achieve this uniformity. Today, however, ECAD tools for physical design and verification cannot remain oblivious to such post-processing. Without an accurate estimate of the downstream filling at the foundry, all the RC extraction, delay calculation, timing, noise and reliability analyses will be highly inaccurate [3], leading to a broken design flow.

Layout Density Control consists of two phases: *density analysis* and *fill synthesis*. The goal of density analysis is to determine the area available for filling within each window. The fill synthesis phase then actually generates the fill geometries that go into these available areas. We refer to the general case (i.e., when we examine and fill *all* possible windows) as the *floating window* regime. On the other hand, when we are concerned with only windows from some fixed dissection over the layout, we refer to these cases as the *fixed-dissection* regime.

Previous papers on this topic gave the first formulations of the *filling problem* that arises in layout post-processing and optimization for manufacturability and yield [3] [4]. These works also developed a number of algorithms for density analysis and proposed filling synthesis algorithms in the fixed-dissection regime for flat designs [3] [4]. This paper proposes new fast multilevel algorithms for maximum density analysis and for filling synthesis in the floating window regime for flat designs as well as for *hierarchical* designs.

2 Multilevel Density Analysis

This section develops new *multilevel* maximum density analysis and filling methods for flat designs. These new methods are based on overhead density estimation, hierarchical zooming, and the combination of floating and fixed-dissection - based techniques.

The algorithms described in earlier papers have two major drawbacks: the fast analysis in the fixed-dissection regime can significantly underestimate the maximum floating-window density in the worst case, while the floating window analysis is too slow when the number of rectangles is large [3]. In order to sidestep these drawbacks, we use four ideas that are based on the following observation.

* Research at UCLA was supported by a grant from Cadence Design Systems, Inc. Professor Robins was supported by a Packard Foundation Fellowship and by NSF Young Investigator Award MIP-9457412.

[1] The length scale over which polishing depth varies is significantly higher than the scale at which microloading and iso-dense optical lithography effects occur. Hence, the latter sources of manufacturing variation tend to be addressed by proximity corrections to the mask geometries, not by feature density control.

[2] For example, local interconnect metal layers may have rules of form: "the total area density of metal features on the layer must be between 0.30 and 0.70". In 0.35μm and below, one major semiconductor house requires diffusion area density between 0.25 and 0.40, and metal area density between 0.40 and 0.70. Another major semiconductor house requires metal area density to be at least 0.35. Density rules may differ between ASIC and high-end microprocessor houses due to tradeoffs between device performance and predictability [1] [10].

Observation 1 *Given a fixed r-dissection*[3]*, any arbitrary floating $w \times w$ window will contain some* shrunk $w(1-1/r) \times w(1-1/r)$ *window of the fixed r-dissection, and will be contained in some* bloated $w(1+1/r) \times w(1+1/r)$ *window of the fixed r-dissection (Figure 1).*

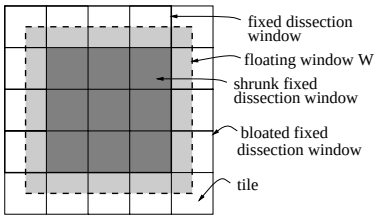

Figure 1: Any floating $w \times w$-window W always contains a *shrunk* $(r-1) \times (r-1)$-window of a fixed r-dissection, and is always covered by a *bloated* $(r+1) \times (r+1)$-window of the fixed r-dissection.

1. We can estimate the possible error inherent in a fixed dissection approximation more accurately than suggested by Theorems 7 and 8 in [3]. The basic idea for our improved approximation is the implication of Observation 1 that the maximum floating window area can be upper-bounded by the maximum area of the bloated windows, and lower-bounded by the maximum area of the shrunk windows.

2. We can use the notion of "zooming" to speed up fixed-dissection density analysis: run the fixed-dissection analysis with $r = r_0/2^k$, find all bloated windows which may contain overfilled floating windows, and delete all tiles which do not belong to any such window. Next, subdivide the remaining tiles in this dissection into 2×2 subtiles, and repeat the fixed-dissection analysis for $r = r_0/2^{k-1}$ and so on, until r becomes equal to r_0.

3. The recursive subdivision above may be continued until the number of rectangles in the remaining tiles is sufficiently small to apply the floating density analysis algorithm from [3] without incurring significant runtime penalty.

4. Alternatively, we may terminate the recursion when we are assured that actual optimal floating window density is within some user-defined accuracy threshold, say $\epsilon = 1\%$ (by comparing it to the computational upper and lower bounds from part 1).

The Multilevel Density Analysis algorithm in Figure 2 incorporates the above ideas. We assume that $\epsilon > 0$ is a user-predefined accuracy threshold for this multilevel analysis, and note that the overall runtime of this multilevel density analysis depends on ϵ.

Since (by Observation 1) any floating $w \times w$ window W is contained in some bloated window, the filled area in W ranges between Max (maximum $w \times w$ standard window filled area found so far) and $BloatMax$ (maximum bloated window filled area found so far). The algorithm terminates when the relative gap between Max and $BloatMax$ is at most 2ϵ and then outputs the middle of the range

[3] A *fixed r-dissection* of the layout is defined as the set of $w \times w$ windows having bottom-left corners at points $(i \cdot \frac{w}{r}, j \cdot \frac{w}{r})$, for $i, j = 0, 1, \ldots, r(\frac{n}{w} - 1)$, where r is an integer divisor of w. Thus, a fixed r-dissection divides the layout into $\frac{n}{w}r \times \frac{n}{w}r$ *tiles*, each of size $\frac{w}{r} \times \frac{w}{r}$, and each $w \times w$ window in a fixed r-dissection consists of r^2 non-overlapping tiles.

Multilevel Density Analysis Algorithm

Input: $n \times n$ layout and accuracy $\epsilon > 0$
Output: maximum area density of $w \times w$ window with accuracy ϵ

Make a list $ActiveTiles$ of all $w/r \times w/r$-tiles
$Accuracy = \infty$, $r = 1$
While $Accuracy > 1 + 2\epsilon$ **do**
  Find all rectangles in $w/r \times w/r$-tiles from $ActiveTiles$
  Find the area of each standard window consisting of tiles from $ActiveTiles$
  Insert all such windows to $WINDOWS$ list
  Max = maximum area of standard window with tiles in $ActiveTiles$
  $BloatMax$ = maximum area of bloated window with tiles in $ActiveTiles$
  For each tile T in $ActiveTiles$ that do not belong to any bloated window of area more than Max **do**
    If $Accuracy > 1 + \epsilon$, then put T in $TILES$
    Remove T from $ActiveTiles$
  Replace in $ActiveTiles$ each tile with four of its subtiles
  $Accuracy = BloatMax/Max$, $r = 2r$
Move all tiles from $ActiveTiles$ to $TILES$
Output max window density = $(Max + BloatMax)/(2 \cdot w^2)$

Figure 2: Multilevel density analysis algorithm.

$(Max, BloatMax)$, which means that our maximum floating window density estimate is at that point within ϵ of the exact value.

Observation 2 *The Multilevel Algorithm (Figure 2) finds a maximum $w \times w$ floating window density that is within a user-predefined accuracy ϵ of the exact value.*

The runtime of multilevel density analysis depends on ϵ. At each iteration of the main loop the difference in area between the bloated and standard window is reduced by half. The main loop terminates when the original area difference $3w^2$ decreases to 2ϵ or below, i.e., when

$$\frac{3w^2}{2^t} \leq 2\epsilon$$

Thus, the maximum number of iterations is estimated by

$$\log_2(\frac{3}{2} \cdot w^2 \cdot \epsilon^{-1}) = O(\log(w/\epsilon))$$

This implies a worst-case runtime of $O((\frac{n}{w} \log \frac{w}{\epsilon})^2)$. In practice, the layout is typically unevenly filled and the majority of tiles are eliminated during the early iterations of the main loop. This explains the excellent performance of multilevel density analysis for actual VLSI layouts (Section 6).

The multilevel analysis can also be applied in finding the *minimum* window density. By Observation 1, the minimum layout area in *shrunk* windows (i.e., fixed r-dissection windows consisting of $(r-1) \times (r-1)$ tiles) is a lower bound for the layout area in an arbitrary $w \times w$ window. Therefore, the multilevel algorithm can be easily modified to find the minimum window density with respect to a user-defined accuracy.

3 Multilevel Filling Area Computation

We now show how to exploit information obtained during the multilevel density analysis phase in order to compute the required fill amounts. Recall that during the multilevel density analysis, we kept track of active tiles (i.e. tiles which may possibly belong to some maximum density window) and

we checked the area of some windows in order to update the maximum window density if necessary. The main goal of the multilevel filling area computation is to decrease the number of variables and constraints in the resulting linear program (LP) described below.

Let $r_{max} = 2^{l_{max}}$ be the highest r reached in the multilevel density analysis algorithm; this corresponds to the user-defined accuracy tolerance parameter ϵ. Instead of considering all $\frac{w}{r_{max}} \times \frac{w}{r_{max}}$-tiles and all $w \times w$-windows consisting of such tiles, we propose to consider only $\frac{w}{2^l} \times \frac{w}{2^l}$-tiles, where $l \leq l_{max}$, and windows consisting of tiles that were tried during the multilevel density analysis phase.

The multilevel filling area computation is implemented as follows. During multilevel density analysis (Figure 2), we save tiles into the set $TILES$ whenever they become "deactivated" (i.e., determined to not belong to any maximum density window), or else when their size becomes $\frac{w}{r_0} \times \frac{w}{r_0}$. We also record the area and slack of each such deactivated tile. On the other hand, each time when we find the area of a $w \times w$-window W, we insert W into the list $WINDOWS$.

In the LP formulation for multilevel filling area computation, each window W in $WINDOWS$ induces two constraints: (i) the first constraint upper-bounds the filled area of W (i.e., the area remaining empty after fill geometries are added to the original layout), and (ii) the second constraint forces an auxiliary variable M to be no greater than the filled area in W, which will enforce the min-variation objective. Each filled window area is expressed as a sum of areas of tiles. In addition, tile fill amount constraints ensure that each tile fill amount is nonnegative as well as not greater than the corresponding tile slack×pattern.

4 Floating Deviation LP Formulation

We now propose another LP formulation that may better reflect the quality of the fill amount computation, since the linear program for the fixed-dissection regime is susceptible to density deviations in floating windows. Consider two different LP solutions in a fixed-dissection regime with different numbers of fixed dissections: the first has r^2 dissections and the second has $(2r)^2$ dissections. Clearly, the more dissections we consider, the more accurate the result will be. On the other hand, more dissections induce more LP constraints and therefore worse (i.e., bigger) deviation is achieved (i.e., smaller value of the auxilary variable M).

A fair comparison of results with different number of fixed dissections entails finding the *floating deviation*, i.e., the difference between the minimum and maximum floating window density. However, since the number of floating windows is too large, we propose comparing *worst-case* estimates of the floating deviation. This can be derived from Observation 1, which implies that the floating deviation cannot be greater than the difference between the maximum area in any $w(1+1/r) \times w(1+1/r)$-window and the minimum area in any $w(1-1/r) \times w(1-1/r)$-window. Therefore, this difference is a reasonable estimate for the floating deviation.

Moreover, instead of comparing LP solutions according to the above estimate of floating deviation, we use this estimate as an *objective* in a new LP formulation. Specifically, we constrain the area of each bloated $w(1+1/r) \times w(1+1/r)$-window by the user-defined density upper bound U, and then seek to maximize the auxiliary variable M which is the lower bound for the area of *any* $w(1-1/r) \times w(1-1/r)$-window. We refer to this formulation as the *min-variation floating window* LP formulation, as it optimally decreases the estimate of the density range between the maximum- and minimum-density floating windows.

5 Hierarchical Layout Density Control

Most modern designs are hierarchical because this makes them substantially smaller, easier and faster to process than flat designs. However, adapting known (flat) layout methods to hierarchical designs is usually a difficult task. In this section we discuss how the multilevel approach to layout density control in the fixed-dissection regime can be applied to hierarchical designs. The main obstacle in hierarchical layout density control is that we cannot flatten the layers because doing so will be prohibitive in terms of memory usage. The problem then is how to analyze and fill hierarchical designs without actually flattening them.

5.1 Hierarchical Filling

We assume that we can fill the slack area of each cell independently and uniformly, as is the case when the size of fill geometries is sufficiently small. For flattened designs, our LP formulation includes area, slack and filling computations for each tile, as follows.

Maximize M subject to:

$$p_{ij} \geq 0, \quad i,j = 1,\ldots,\frac{nr}{w} - 1 \tag{1}$$

$$p_{ij} \leq slack(T_{ij}), \quad i,j = 1,\ldots,\frac{nr}{w} - 1 \tag{2}$$

$$\sum_{s=i}^{i+r-1} \sum_{t=j}^{j+r-1} p_{st} \leq \alpha_{ij} \left(U \cdot w^2 - area_{ij}\right), \quad i,j = 1,\ldots,\frac{nr}{w} - r + 1 \tag{3}$$

$$M \leq area_{ij} + \sum_{s=i}^{i+r-1} \sum_{t=j}^{j+r-1} p_{st}, \quad i,j = 1,\ldots,\frac{nr}{w} - r + 1 \tag{4}$$

where

$$area_{ij} = \sum_{s=i}^{i+r-1} \sum_{t=j}^{j+r-1} area(T_{st})$$

is the area of the (i,j)-th window, and $\alpha_{ij} = 0$ if $area_{ij} > U \cdot w^2$ and 1, otherwise.

The constraints (1) imply that we can only add features, and cannot delete features from any tile. The slack constraints (2) are computed for each tile. If a tile T_{ij} is originally overfilled, then we set $slack(T_{ij}) = 0$. The values of p_{ij} from the LP solution indicate the fill amount to be inserted in each tile T_{ij}. The constraints (3) say that the no window can have density more than U after filling unless it was initially overfilled. Inequalities (4) imply that the auxiliary variable M is a lower bound on all window densities. The linear program seeks to maximize M, thus achieving the min-variation objective. The number of variables and the number of constraints in the linear program are both $O((\frac{nr}{w})^2)$. In practice, even for a large die and a user requirement of high accuracy, we might have $n = 15,000$, $w = 3,000$, and $r = 10$, which still yields a linear program of tractable size.

We now show how to compute the fill areas for cells (rather than tiles), with the main goal being to minimize the window density variation while preserving the hierarchical structure of the design. For each tile T_{ij} we consider its partition into cells, while viewing the fill area as a sum of normalized fill areas in each intersecting cell (Figure 3).

$$area(T_{ij}) = \sum_{kl} \beta_{ij,kl} \cdot area(C_{kl}) \tag{5}$$

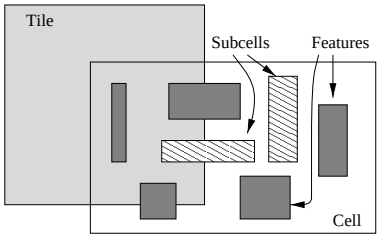

Figure 3: Intersection of a tile and a cell. The dark features and patterned subcells may completely or partially overlap with the tile.

where for a tile T_{ij} intersecting the l-th copy C_{kl} of a the k-th cell C_k, we define $\beta_{ij,kl} = area(T_{ij} \cap C_{kl})/area(C_k)$ to be the normalized area of features overlapping with each tile. Let the variables d_k indicate the fill amount to be inserted in each cell C_k.

$$p_{ij} = \sum_{k=1}^{c} d_k \cdot \sum_{l} \gamma_{ij,kl} \tag{6}$$

where $\gamma_{ij,kl} = slack(T_{ij} \cap C_{kl})/slack(C_k)$ is the normalized slack of each cell-tile intersection. In the new LP formulation we add constraints (5 - 6) and replace constraints (1-2) with the following two constraints:

$$d_k \geq 0, \quad k = 1, \ldots, c \tag{7}$$

$$d_k \leq slack(C_k), \quad k = 1, \ldots, c \tag{8}$$

Observation 3 *The solution for the LP above optimally solves the fill synthesis problem for hierarchical designs.*

6 Experimental Results

Our current experimental testbed integrates GDSII Stream input, conversion to CIF format, and internally-developed geometric processing engines, coded in C++ under Solaris. We ran experiments using three metal layers extracted from industry standard-cell layouts (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; and Benchmark L3 is the M2 layer from the same 20,577-cell design. The layout dimension N, number of rectangles k, and window size w (w always chosen to equal 1.5mm) for each test case are shown in Table 1.

Table 2 compares the runtimes[4] of the fixed-dissection analysis for $r = 4, 8$ and the multilevel analysis with accuracy tolerance $\epsilon = 2\%, 3\%$. To enable comparison of results, for multilevel analysis we report the maximum density of a standard window, rather than the midpoint between the maximum-density standard and bloated windows. The data indicates that the multilevel approach is more accurate as well as more efficient than the fixed-dissection analysis.

Table 3 compares the fixed-dissection [4] and floating deviation LP approaches. The minimum density achieved by the new approach is slightly lower than for the standard approach because the new approach tries to make the filling area distribution more uniform. The more uniform distribution is achieved with similar total runtime (LP generation, LP solution and fill pattern generation).

[4]CPU time corresponds to seconds on a 140MHz Sun Ultra-1 with 256MB RAM.

Acknowledgments. We thank Larry Camilletti and Duane Boning for enlightening discussions, and gratefully acknowledge software donations from Avant! Corp. and Artwork Conversions, Inc.

References

[1] R. Bek, C. C. Lin and J. H. Liu, personal communication, December, 1997.

[2] R. R. Divecha, B. E. Stine, D. O. Ouma, J. U. Yoon, D. S. Boning, J. E. Chung, O. S. Nakagawa, and S. Y. Oh, *Effect of Fineline Density and Pitch on Interconnect ILD Thickness Variation in Oxide CMP Process*, in Proc. CMP-MIC, Santa Clara, February 1998.

[3] A. B. Kahng, G. Robins, A. Singh, H. Wang, and A. Zelikovsky, *Filling and Slotting: Analysis and Algorithms*, in Proc. International Symposium on Physical Design, Monterey, CA, April 1998, pp. 95–102.

[4] A. B. Kahng, G. Robins, A. Singh, and A. Zelikovsky, *New and Exact Filling Algorithms for Layout Density Control*, in Proceedings of the 12th International Conference on VLSI Design, 1999.

[5] H. Landis, P. Burke, W. Cote, W. Hill, C. Hoffman, C. Kaanta, C. Koburger, W. Lange, M. Leach, and S. Luce, *Integration of Chemical-Mechanical Polishing into CMOS Integrated Circuit Manufacturing*, Thin Solid Films, 220 (1992), pp. 1–7.

[6] W. Maly, *Moore's Law and Physical Design of ICs*, in Proc. International Symposium on Physical Design, Monterey, California, April 1998. special address.

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

[8] SIA, *The National Technology Roadmap for Semiconductors*, Semiconductor Industry Association, December 1997.

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

[10] K. Wampler and T. Laidig, personal communication, September, 1997.

Industry Test Cases			
Benchmark	layout size	# rectangles	window size
L1	125,000	49,506	31,250
L2	112,000	76,423	28,000
L3	112,000	133,201	28,000

Table 1: Parameters of three industry test cases.

Fixed-Dissection & Multilevel Density Analysis							
	Fixed-Dissection			Multilevel			
Bench-mark	r	Max Density	CPU (sec.)	ϵ	Max Std Density	CPU (sec.)
L1	4	.2125	2.9	3%	.2184	2.8
L1	8	.2170	9.2	2%	.2184	2.8
L2	4	.1791	4.5	3%	.1829	3.8
L2	8	.1791	14.5	2%	.1830	6.9
L3	4	.2895	8.0	3%	.2911	6.6
L3	8	.2910	25.1	2%	.2925	7.1

Table 2: Fixed-dissection ($r = 4, 8$) and multilevel (accuracy tolerance $\epsilon = 2\%$, 3%) density analysis results.

		Fixed-dissection LP		Floating Deviation LP	
Bench-mark	r	Min Density	CPU (sec.)	Min Density	CPU (sec.)
L1	2	.2192	7.6	.2184	7.5
L1	4	.2192	7.6	.2165	7.7
L1	8	.2189	31.9	.2109	19.3
L2	2	.1816	8.0	.1748	8.0
L2	4	.1704	11.9	.1470	10.8
L2	8	.1631	62.5	.1354	64.9
L3	2	.2640	13.5	.2619	14.3
L3	4	.2606	18.2	.2578	20.2
L3	8	.2553	59.7	.2487	54.1

Table 3: Total CPU times for solving the Filling Problem while minimizing fixed-dissection and floating deviation.