RC24076 (W0610-054) October 13, 2006
Computer Science

IBM Research Report

Is Your Layout Density Verification Exact? A Fast Exact Algorithm for Density Calculation

Hua Xiang[1], Kai-Yuan Chao[2], Ruchir Puri[1], Martin D.F. Wong[3]

[1]IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

[2]Intel Corporation
Hillsboro, OR

[3]ECE Department
University of Illinois at Urbana-Champaign
Urbana, IL

Is Your Layout Density Verification Exact ? — A Fast Exact Algorithm For Density Calculation

Hua Xiang
IBM T.J. Watson
Yorktown Heights NY
huaxiang@us.ibm.com

Kai-Yuan Chao
Intel Corporation
Hillsboro OR
kaiyuan.chao@intel.com

Ruchir Puri
IBM T.J. Watson
Yorktown Heights NY
ruchir@us.ibm.com

Martin D.F. Wong
ECE Dept, UIUC
Urbana IL
mdfwong@uiuc.edu

Abstract

As the device shapes keep shrinking, the designs are more sensitive to manufacturing processes. In order to improve performance predictability and yield, mask layout uniformity/evenness is highly desired, and it is usually measured by the feature density with defined feasible range in manufacture process design rules. To address the density control problem, one fundamental problem is how to calculate density accurately and efficiently. In this paper, we propose a fast exact algorithm to identify the maximum density for a given layout. Compared with the existing exact algorithms, our algorithm reduces the running time from days/hours to a few minutes/seconds. And it is even faster than the existing approximate algorithms in literature.

Category: B.7.2 [Integrated Circuits]: Design Aids - Layout; J.6 [Computer Applications]: Computer-Aided Engineering - Computer-aided design

Terms: Algorithms, Design

Keywords: density, fix-dissection, DFM

1. Introduction

In very deep-submicron VLSI, manufacturing variations have become an important consideration in chip designs [1, 15, 16, 19]. Mask layout uniformity is highly desired in order to improve performance predictability and yield. To evaluate layout planarity, one major criteria is the feature density, which is defined as the percentage of the total feature area on a given layer in a given check window. Several manufacturing processes, such as CMP (Chemical Mechanical Polishing), etch, CD (Critical Dimension) control, and even lithography (implicitly for pitch) [2] are all sensitive to pattern density such that foundries usually require an effective metal density to be satisfied, and the density rules are associated with many process layers including diffusion and thin-ox [3, 6]. Meanwhile, in order to satisfy density rules, dummy fills are inserted on the original layout to balance layout density distribution in order to resolve minimum density violations and achieve the evenness control. Therefore, to address the density control problem (both density rule checking and dummy fill insertion), one fundamental issue is how to calculate density accurately and efficiently.

Given a fixed layout and window size, ideally, we want to identify the windows with the maximum (minimum) density. In [3, 6], this problem is defined as "Extremal-Density Window Analysis".

Extremal Density Window Analysis (EDWA): Given a fixed window size W and a MxN layout with K non-overlap rectangles, find a WxW density window which has the maximum (minimum) density.

In this paper, we propose an exact and fast algorithm to find the maximum density window on a given layout. In some cases, identifying windows whose density is higher/lower than the given density is required. For example, a foundry requires that the density range should be $[15\%, 85\%]$. If the density of a window is outside the range, the window is reported to have the density violation, and it should be fixed with dummy fills/cheesing holes. With a little modification of our algorithm, it can be used to identify window density violations as well. In the rest of the paper, we will focus on the EDMA problem to find the maximum density windows. The minimum density window can be handled in the similar way.

In literature, [3, 6] proposed exact algorithms to address EDMA problems. However, the running time is very long. As shown later in the experimental result section, the running time is measured by hours or days. Such a long running time prohibits these algorithms to be used in real designs. [4, 6] also proposed approximate algorithms such that the difference between the reported maximum density and the actual maximum density is within the given error bound. However, the algorithms cannot report the exact maximum window density, and the running time depends on the given error bound. The smaller the error bound is, the longer the running time.

Due to the lack of efficient density calculation algorithms, most commercial tools use fix-dissection approach (it is also called sliding-window approach) to estimate density. In Section 2, we address the limitation of this approach, and show that only a very small percent of windows are selected for the density check. In Section 3, we present density theorems which are the basis of our density calculation algorithm. Then we propose an exact fast density calculation algorithm in Section 4. The experimental results are presented in Section 5, and Section 6 concludes the paper.

2. Fix-Dissection Approach Limitation

A standard practice in the density calculation is to consider only windows from a fix dissection [5, 7, 8, 9, 10, 11, 12, 13, 14, 17, 18, 20]. In this approach, a layout is partitioned into $\frac{M}{R}$x$\frac{N}{R}$ non-overlapping RxR windows. Usually W is multiple times of R, and R is called sliding step. Then only windows whose boundaries fall on the R-grid is checked for density as shown in Figure 1. For convenience, we call this kind of windows as sliding-window. In Figure 1, the two blue solid windows are on the grid, and they are checked for the window density. But the red dash window won't be considered. In this work, we are using square windows, while our algorithm is also applicable to general rectangular windows.

Fix-Dissection approach is fast, but it only checks a very limited number of windows. In this MxN layout, totally there are $(M-W+1)$x$(N-W+1)$ windows. With R-dissection, only $(\frac{M-W}{R}+1)$x$(\frac{N-W}{R}+1)$ windows are checked. For example, suppose the layout is $1mm$x$1mm$, and the window size is $20um$. Let the minimum feature unit is $10nm$. Then in this case, $M = N = 1mm/10nm = 10^5$,

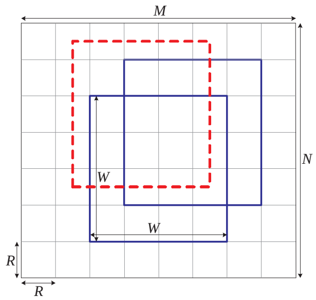


Figure 1: Fix-dissection approach. The layout is divided by a R-grid. Only windows (the blue solid windows) whose boundaries are on the grid are checked for density. Other windows such as the red dash window is not checked.

$W = 20um/10nm = 2000$. So totally there are around 9.6x10^9 windows. Most industrial applications use $R = W/2$, then the number of sliding-windows is less than 1x10^4. Therefore, only a very small percent of windows are checked. This may not be a problem in previous technology nodes, but the non-exact density verification will become a significant issue in deep submicron technology for DFM (Design For Manufacture). Especially, as device shapes keep shrinking, the minimum feature size will become even smaller, while the chip size will become larger, the demand on density accuracy will increase. For example, the recent TSMC DFM Data Kit (DDK) requires an accurate density input for etch/deposit/CMP depths calculation.

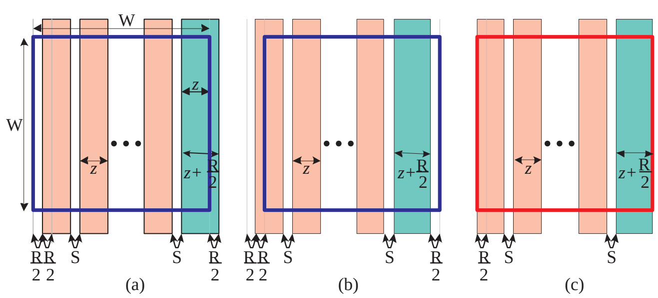


Figure 2: The two windows in (a) and (b) are on R-grid, and have the maximum density D from the fix-dissection approach. The window in (c) is not a sliding-window, but it has larger density.

Finally, we show that by shrinking R, the fix-dissection approach cannot produce the exact density calculation until R reaches the minimum feature size.

Lemma 1. *If R is larger than the minimum feature size, the fix-dissection approach cannot guarantee to solve the extremal-density window analysis problem exactly.*

Figure 2 shows a counter example. In Figure 2, suppose the maximum density of sliding-windows is D, and the two windows in Figure 2 (a) and (b) reach the maximum density D. For simplicity, let the wires long enough such that only x-direction need be considered in density calculation. Assume there are P wires, and all wires except the rightmost green one have the same wire width z. The rightmost wire has a width $z+R/2$. The wire spacing is $s = \frac{W-P\cdot z-R/2}{P-1}$. Then for the window in (c), its density is $\frac{P\cdot z+R/2}{W} > D$. This case shows that shrinking R in the fix-dissection approach cannot guarantee to find the maximum density window.

3. Density Bound

Although fix-dissection approach cannot guarantee to identify the maximum density window, it provides basic information on density distribution.

Theorem 1. *Any window Win must be fully covered by four sliding-windows. And the density of Win satisfies that* $D_{Win} - D_{Rwin} \leq \frac{R}{W} - (\frac{R}{2W})^2$, *where* D_{win} *is the density of Win, and* D_{Rwin} *is the maximum density of the four sliding-windows.*

PROOF. Suppose the coordinate of the left bottom corner of Win is (x,y). Then it must be covered by four sliding-windows whose left bottom corners are $(\lfloor\frac{x}{R}\rfloor\cdot R, \lfloor\frac{y}{R}\rfloor\cdot R)$, $(\lfloor\frac{x}{R}\rfloor\cdot R+R, \lfloor\frac{y}{R}\rfloor\cdot R)$, $(\lfloor\frac{x}{R}\rfloor\cdot R, \lfloor\frac{y}{R}\rfloor\cdot R+R)$, and $(\lfloor\frac{x}{R}\rfloor\cdot R+R, \lfloor\frac{y}{R}\rfloor\cdot R+R)$. As shown in Figure 3, the red window is fully covered by the four blue windows in Figure 3 (a), (b), (c) and (d). For convenience, the four windows are named as W_{LB}, W_{RB}, W_{LU} and W_{RU}, respectively. Also let their density be D_{LB}, D_{RB}, D_{LU} and D_{RU}. Assume that $u = x - \lfloor\frac{x}{R}\rfloor\cdot R$ and $v = \lfloor\frac{x}{R}\rfloor\cdot R + R - x$. Similarly, $s = y - \lfloor\frac{y}{R}\rfloor\cdot R$ and $t = \lfloor\frac{y}{R}\rfloor\cdot R + R - y$.

The density difference between Win and one of the four sliding-windows is decided by the metal area in the shadow regions (the blue region and the red region in Figure 3). The maximum difference can be reached when one region has no features while the other region is totally occupied. Therefore, we have

$$\begin{aligned} D_{Win} - D_{LB} &\leq \frac{W\cdot u + W\cdot s - u\cdot s}{W^2} \\ D_{Win} - D_{RB} &\leq \frac{W\cdot v + W\cdot s - v\cdot s}{W^2} \\ D_{Win} - D_{LU} &\leq \frac{W\cdot u + W\cdot t - u\cdot t}{W^2} \\ D_{Win} - D_{RU} &\leq \frac{W\cdot v + W\cdot t - v\cdot t}{W^2} \end{aligned}$$

Since $D_{Rwin} = \max\{D_{LB}, D_{RB}, D_{LU}, D_{RU}\}$, we have

$$D_{Win} - D_{Rwin} \leq \min\{\frac{W\cdot(u+s)-u\cdot s}{W^2}, \frac{W\cdot(v+s)-v\cdot s}{W^2}, \frac{W\cdot(u+t)-u\cdot t}{W^2}, \frac{W\cdot(v+t)-v\cdot t}{W^2}\}$$

Without loss of generality, we assume $u \leq v$ and $s \leq t$. Then

$$\min\{\frac{W\cdot(u+s)-u\cdot s}{W^2}, \frac{W\cdot(v+s)-v\cdot s}{W^2}, \frac{W\cdot(u+t)-u\cdot t}{W^2}, \frac{W\cdot(v+t)-v\cdot t}{W^2}\} = \frac{W\cdot(u+s)-u\cdot s}{W^2}$$

In short, we get $D_{Win} - D_{Rwin} \leq \frac{W\cdot(u+s)-u\cdot s}{W^2}$.

Next, we prove that $\frac{W\cdot(u+s)-u\cdot s}{W^2} \leq \frac{R}{W} - (\frac{R}{2W})^2$.

Since $u+v = R$ and $u \leq v$, we get $u \leq \frac{R}{2}$. Similarly, $s \leq \frac{R}{2}$. Let $u = \frac{R}{2} - a$ and $s = \frac{R}{2} - b$ $(a, b \geq 0)$. Then

$$\begin{aligned} \frac{W\cdot(u+s)-u\cdot s}{W^2} &= \frac{R-a-b}{W} - \frac{1}{W^2}\cdot(\frac{R}{2}-a)\cdot(\frac{R}{2}-b) \\ &= \frac{1}{W^2}\cdot(a+b)\cdot(\frac{R}{2}-W) - \frac{a\cdot b}{W^2} + \frac{R}{W} - (\frac{R}{2W})^2 \end{aligned}$$

We know that $W > \frac{R}{2}$. Therefore, only when $a = b = 0$, the maximum value can be reached. Thus we have $\frac{W\cdot(u+s)-u\cdot s}{W^2} \leq \frac{R}{W} - (\frac{R}{2W})^2$. □

Theorem 1 gives the density bound of a window. The next theorem states the properties of a maximum density window within

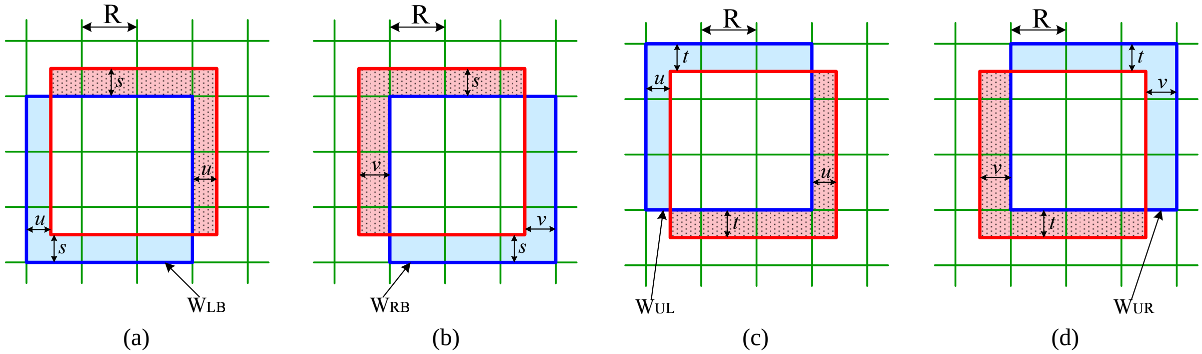


Figure 3: Any window (the red window) can be covered by four sliding-windows.

a given region. For convenience, we define edge directions of a window/rectangle as the clockwise direction.

Theorem 2. *Given a region with k rectangles, there exists a maximum density window that has two adjacent window edges overlap two rectangle edges. Furthermore, the overlapped window edges and rectangle edges are in the same directions.*

PROOF. First, we prove that if neither the left nor the right edge of a maximum density window touches the edges of any rectangles, then the window density is not changed when the window moves left/right.

In Figure 4, the blue solid window is a maximum density window, and no rectangle edges are on the window edge. When the maximum density window moves left before touching any rectangle edge, the feature area in the two shadow regions is the same. Otherwise, suppose the purple dotted window has less density. Since the shadow region has the same width, only the total height of the rectangles that cross the left/right edge of the blue window matters. If the dotted window has less density, we have $h_1+h_2 < h_3$ as shown in Figure 4 (a). Then the green dash window in Figure 4 (b) has a density higher than the maximum density since $v\cdot(h_1+h_2) < v\cdot h_3$.

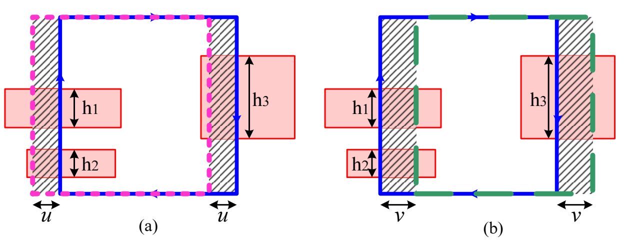


Figure 4: The blue solid window has the maximum density. (a) When the blue window moves left without touching any rectangle edges, the total feature areas in the two shadow regions are the same. (b) When the blue window moves right without touching any rectangle edges, the feature areas in the two shadow regions are the same.

Similarly, if we move a maximum density window up/down without touching a rectangle edge, the window density won't be changed. Therefore, we can always find a maximum density window whose two adjacent edges overlap with two edges of rectangles.

Next, we show that the overlapped window edges and rectangle edges are in the same directions. Suppose no same direction window/rectangle edge pair overlaps with each other.

In Figure 5, the blue solid window *Win* has the maximum density, and either the left window edge touches a right rectangle edge (such as C), or the right window edge touches a left rectangle edge (such as E and F). But no rectangles like A or B exist. We define:

H_l: total height of rectangles crossing the left window edge;

H_r: total height of rectangles crossing the right window edge;

T_l: total height of rectangles whose right edge touches the left window edge;

T_r: total height of rectangles whose left edge touches the right window edge.

For the example in Figure 5, $H_l = h_d$, $T_l = h_c$, $H_r = h_g$ and $T_r = h_e + h_f$. When the window moves left or right with a step u without touching any new rectangle edges, we get two new windows as shown in Figure 5. One is the purple dotted window Win_l and the other is the green dash window Win_r.

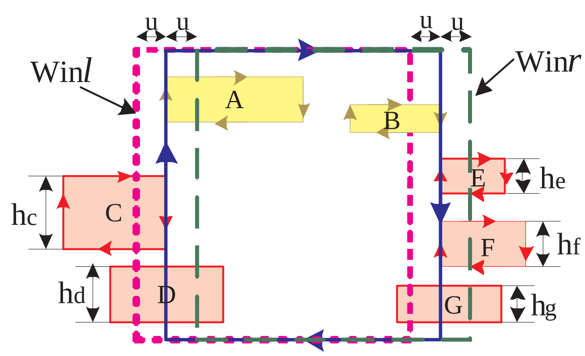


Figure 5: Assume that rectangles A and B do not exist. Rectangle C, E and F have an edge overlapping with a window edge. But the directions are different.

Also let D, D_l and D_r be the density of Win, Win_l and Win_r, respectively. Then we get $D - D_l = \frac{(H_r-H_l-T_l)\cdot u}{W^2}$, and $D - D_r = \frac{(H_l-H_r-T_r)\cdot u}{W^2}$. Since D has the maximum density, we have

$$\frac{(H_r - H_l - T_l)\cdot u}{W^2} \geq 0 \Rightarrow H_r \geq H_l + T_l$$

and

$$\frac{(H_l - H_r - T_r)\cdot u}{W^2} \geq 0 \Rightarrow H_l \geq H_r + T_r$$

Therefore $T_l = T_r = 0$. This contradicts our assumption that at least one rectangle edge touches a window edge. Similar proof applies for y-direction. □

4. Density Calculation Algorithm

In this section, we propose an algorithm to solve the EDWA problems. The algorithm is based on the two theorems. We first outline the whole algorithm, then detail explanations are presented for each step. For convenience, we use a triple to represent a square region. For example, (x, y, B) refers to a BxB region whose left bottom corner is at (x, y).

Algorithm Density_Calculation(M, N, W, R, $Rects$)

1. $m = M/R$; $\quad n = N/R$;
2. $err = R/W - (R/2W)^2$;
3. $Rmap$ = Build_Rect_Map(R, $Rects$);
4. For each grid tile $(iR,\ jR,\ R)$
5.   Calculate the tile density $Rgrid[i][j]$;
6.
7. For each region $(iR,\ jR,\ W-R)$
8.   Calculate the region density $Rcenter[i][j]$;
9.
10. For each region $(iR,\ jR,\ W)$
11.   Calculate the window density $Rwin[i][j]$;
12.
13. $U = \max\{Rwin[i][j]\}$;
14. For $i = 0$ to $m-2$
15.   For $j = 0$ to $n-2$
16. $dmax = \max\{Rwin[i][j],\ Rwin[i+1][j],$ $Rwin[i][j+1],\ Rwin[i+1][j+1]\}$;
17.   if($dmax + err > U$)
18. U = Detail_Density ($iR, jR, R, Rcenter[i+1][j+1]$);
19. Output U;

In Density_Calculation, M and N are the x and y dimension of the given layout. W is the window size, and R is the sliding step. $Rects$ records all rectangles in the given layout.

4.1 Data Preparation

The algorithm starts from a fix-dissection density calculation with a sliding step R. After setting up a mxn grid, we define a center-window as a sliding-window with a window size $(W-R)$. As shown in Figure 6 (a), the blue solid window is a sliding-window, and the red dotted window is a center-window. For each grid tile, we calculate its density, and store the results in a $2D$ array $Rgrid$. Based on $Rgrid$, it is easy to get the center-window density and sliding-window density, and the values are saved in the $2D$ array $Rcenter$ and $Rwin$, respectively. Since each center-window is a part of a sliding-window, the center-window density calculation does not need extra running time. At the same time, the maximum sliding-window density U is derived. Lines $1 \sim 13$ finish these preparations.

To calculate the density of a region is a basic operation in this algorithm, and its first step is to identify rectangles which have overlap with the region. Therefore, we build a rectangle map to speed rectangle searching. For the given MxN layout, claim a two-dimension array $Rmap[M/R, N/R]$, the elements of $Rmap$ are a rectangle id list. For any rectangle $rect$, suppose (x_l, y_l) and (x_h, y_h) are the left bottom corner and the right upper corner of the rectangle, respectively. Then $rect$ will be recorded in $Rmap[\lfloor x_l/R \rfloor .. \lfloor x_h/R \rfloor, \lfloor y_l/R \rfloor .. \lfloor y_h/R \rfloor]$. Figure 7 shows an example. Figure 7 (a) gives a layout with 14 rectangles. Figure 7 (b) is the $Rmap$. $Rmap[i][j]$ is a rectangle id list.

The algorithm Build_Rect_Map is summarized as follows. R is the original sliding step, and $Rects$ is the rectangle list.

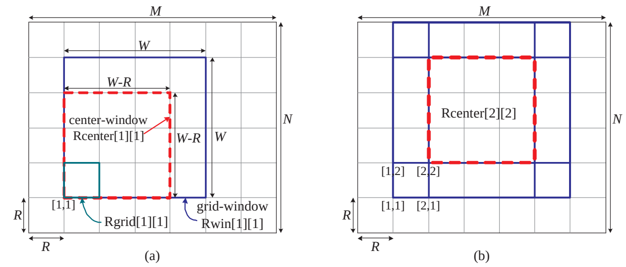


Figure 6: (a) Illustration of $Rgrid$, $Rcenter$ and $Rwin$. (b) Suppose the maximum sliding-window density of $Rwin[1][1]$, $Rwin[1][2]$, $Rwin[2][1]$, and $Rwin[2][2]$ is larger than U. Then the blue region $(1, 1, W+R)$ is selected for further density analysis. The center-window $Rcenter[2][2]$ is shared by the four sliding-windows.

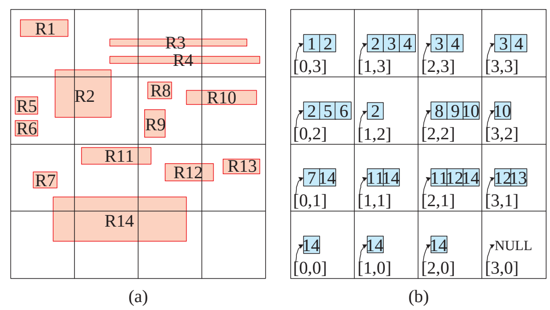


Figure 7: (a) A layout with 14 rectangles (b) The corresponding $Rmap$

Algorithm Build_Rect_Map(R, $Rects$)

Initialize $Rmap$ as $Rmap[i][j] = NULL$
For each $rect$ in $Rects$
    For $ii = \lfloor x_l/R \rfloor$ to $\lfloor x_h/R \rfloor$
        For $jj = \lfloor y_l/R \rfloor$ to $\lfloor y_h/R \rfloor$
            Insert $rect$ into $Rmap[ii][jj]$;

According to Theorem 1, we know that any window can be covered by four sliding-windows, and its density is bounded by $dmax$ (the maximum density of the four sliding-windows) plus err $(R/W - (R/2W)^2)$. Therefore, if $dmax + err \le U$, it means that any window inside the region that is covered by the four sliding-windows cannot have a density larger than U, and we do not need consider those windows any more. This step helps to prune many regions so that we only need focus on certain areas which will be handled by Detail_Density. In Figure 6 (b), suppose the maximum value of $Rwin[1][1]$, $Rwin[1][2]$, $Rwin[2][1]$, and $Rwin[2][2]$ is larger than U, then the blue region is selected for Detail_Density. Meanwhile, the center pink region is shared by these four sliding-windows, and $Rcenter[2][2]$ is fed into Detail_Density as an input.

4.2 Density Calculation

In this section, we present algorithms to identify the maximum density window in a given region. The main idea is to recursively apply the fix-dissection approach with smaller sliding steps. For each dissection, we can further prune regions based on Theorem 1. When the region size is small enough, we will call Exact_Density algorithm to report a maximum density window in the given region.

Algorithm Detail_Density(X, Y, B, *center_dens*)

1. If ($B + W < DSIZE$)
2. Then U = Exact_Density(X, Y, B, *center_dens*);
3. Return U;
4.
5. $\bar{R} = B/k$
6. For $i = 0$ to $2k$
7. For $j = 0$ to $2k$
8. if($i == k$ and $j == k$)
9. then $Rdens[i][j] = center\_dens$;
10. continue;
11. $region.x_l$ = Region_Point(X, i, $\bar{R}$);
12. $region.y_l$ = Region_Point(Y, j, $\bar{R}$);
13. $region.x_h$ = Region_Point(X, $i+1$, $\bar{R}$);
14. $region.y_h$ = Region_Point(Y, $j+1$, $\bar{R}$);
15. Calculate *region* density $Rdens[i][j]$;
16.
17. Calculate center-window density $Cdens[i][j]$ $(1 \le i, j \le k)$;
18. Calculate sliding-window density $Wdens[i][j]$ $(0 \le i, j \le k)$;
19.
20. $err = \bar{R}/W - (\bar{R}/2W)^2$;
21. For $i = 0$ to k
22. For $j = 0$ to k
23. $dmax$ = $\max\{Wdens[i][j],\ Wdens[i+1][j],\ Wdens[i][j+1],\ Wdens[i+1][j+1]\}$;
24. if($dmax + err > U$)
25. U = Detail_Density($i\bar{R}$, $j\bar{R}$, $\bar{R}$, $Cdens[i+1][j+1]$);

In Detail_Density, X and Y are the coordinates of the left bottom corner of the input region. B is the original sliding step, and $\bar{R}$ is the new sliding step. k is a pre-defined division factor such that $\bar{R} = \frac{B}{k}$. Lines $5 \sim 18$ calculate sliding-window density. Region_Point is to get the four corners of a grid tile. The grid tile is illustrated as Figure 8 (c) and the details will be presented in Section 4.2.2. Lines $20 \sim 25$ prune regions based on Theorem 1.

4.2.1 *Region Properties*

As we notice that all regions processed by Detail_Density have a size less than $2W$x$2W$. Therefore, for any input LxL ($L < 2W$) region, we have the following observations:

1. $L = W + B$.
2. All WxW windows inside this region share a $(2W-L)$x$(2W-L)$ area, which is in the center of the region. Furthermore, the density of this area is *center_dens*.
3. The left bottom corner of any WxW window must fall in the BxB area on the left bottom corner of the region.
4. The number of sliding-windows within this region is $(k+1)^2$.

In Figure 8 (a), the center green area is covered by any WxW window inside the LxL region. And the left bottom corner of all windows must be within the pink area including the boundaries. If the sliding step is $\bar{R}$, then the total number of grid points inside the pink area is $(\frac{B}{\bar{R}}+1)^2$, i.e., $(k+1)^2$. In other words, there are $(k+1)^2$ sliding-windows in the given region.

4.2.2 *Region Partition*

The main idea of Detail_Density is to recursively apply the fix-dissection approach with smaller sliding steps. The running time of the fix-dissection approach is closely related to:

(1) the number of rectangles to be checked for each tile;
(2) the number of tiles.

To reduce the number of rectangles to be checked for each tile, we draw on *Rmap*. With the help of *Rmap*, we only need check a few rectangle lists instead of traversing all rectangles. For example, if a tile is (x, y, H). Then we only need check the rectangle lists registered in $Rmap[\lfloor \frac{x}{R} \rfloor .. \lfloor \frac{x+H}{R} \rfloor, \lfloor \frac{y}{R} \rfloor .. \lfloor \frac{y+H}{R} \rfloor]$, where R is the initial sliding step.

When we apply the fix-dissection algorithm, the region is partitioned into $\frac{L}{\bar{R}}$x$\frac{L}{\bar{R}}$ tiles as the grid in Figure 8 (b). From the region property (2), we know that the center green area is shared by all sliding-windows. Therefore, it is not necessary to divide this area into tiles. Instead, the whole green area should be treated as one tile. Furthermore, the density of this center area is already calculated in the last dissection when the sliding step is B, and the value is input to the new dissection as *center_dens*. Since the center area takes a large percent of a sliding-window, *center_dens* helps save a lot of rectangle-tile overlap checking. For example, in Figure 8 (b), $L/B = 8$. The center area takes more than half of the whole region, while we only need calculate the density of the tiles along the boundaries.

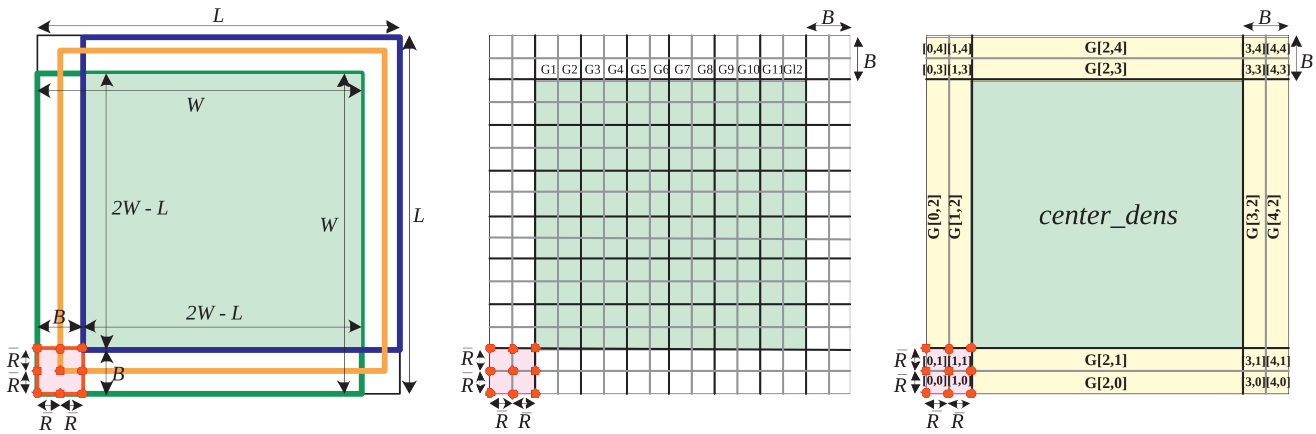
Meanwhile, the left bottom corner of sliding-windows can only fall in the pink region. It is not necessary to calculate density for grids G_i $(i = 1, .., 12)$ separately as shown in Figure 8 (b). Therefore, these twelve tiles can be merged, and be treated as one tile as $G[2,3]$ in Figure 8 (c). In this way, the number of tiles is reduced from 256 in Figure 8 (b) to 24 in Figure 8 (c). For a general case, if the size of the whole region is $W + B$, then the number of tiles can be reduced from $(\frac{W+B}{\bar{R}})^2$ to $(\frac{2B}{\bar{R}}+1)^2 = (2k+1)^2$. k is a given constant. Thus only a constant number of tiles need to be checked for a given region at each dissection. In the algorithm Detail_Density, Region_point is to get the coordinates of these tiles. Once the tile region is identified, its density can be easily derived.

4.2.3 *Exact Maximum Density Calculation*

Given a region (X, Y, L) $(L < 2W)$, we want to find a window with the maximum density. The total number of windows within this region is $(L - W + 1)^2$. For example, if $L = 2,200$ and $W = 2,000$, there are $40,401$ windows. Still, the searching spacing is pretty large. On the other hand, from Theorem 2, we know that at least one maximum density window has its two adjacent edges overlap with two rectangle edges, and the window/rectangle edge pair has the same direction. This motivates us to focus on the windows satisfying these conditions so that the searching space can be significantly reduced.

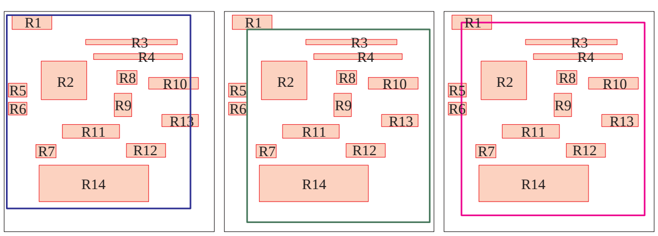
In Figure 9 (a), the window satisfies all the above requirements. The left window edge overlaps the left edge of R_5 and R_6, and the upper window edge touches the upper edge of R_1. On the other hand, although the left and upper window edge in Figure 9 (b) also overlap with an edge of R_5, R_6 and R_1, their directions are different. The left window edge overlap with the right edge of R_5 and R_6, and the upper window edge overlap with the bottom edge of R_1. Therefore, this kind of windows dose not satisfy the criteria. In Figure 9 (c), none of the window edges touch any rectangle edges, and this window won't be considered.

As we notice that the region size L is always less than $2W$. In section 4.2.1, we know that all windows inside such a region share the center $(2W - L)$x$(2W - L)$ area. Therefore, we do not need consider the rectangles inside the center area, and get its density from the input *center_dens* instead. Furthermore, the window edge distribution has its own range. For the left window edge, its x-



(a) (b) (c)

Figure 8: (a) All windows inside the *LxL* region share the center green area. The left bottom corner of all sliding windows fall in the pink region. (b) Full dissection with *RxR* tiles. The dark grid is the previous dissection with a sliding step *B*, where $B = 2\bar{R}$. Totally there are 256 tiles that need density calculation. (c) Only the 24 yellow tiles need density calculation. The center green tile gets its density from the input *center_dens*.



(a) (b) (c)

Figure 9: (a) The left and upper edges of the window overlap with a left rectangle edge and an upper rectangle edge. (b) A window has two adjacent edges overlapping with two rectangle edges. But the rectangle/window edges have different directions. (c) A window has no edge overlapping with any rectangle edges.

coordinate must fall in the range $[X, X + (L - W)]$. Therefore, we only need record the left rectangle edges satisfying this constraint. If the *x*-coordinate of a left rectangle edge is larger than $X + (L - W)$, no feasible window matches this rectangle edge with its left window edge. We have similar constraints for the right, upper and bottom window edges.

Figure 10 shows an example. In Figure 10 (a), there are 14 rectangles in the given *LxL* region. The center $(L - 2B)$x$(L - 2B)$ area is shared by all windows inside the region. Since the left window edge can only fall within the first column, we only need consider the left edge of R_1, R_5 and R_6 as the blue and green lines in Figure 10 (b). Although the right edges of R_5 and R_6 are also within the first column, they are not considered. Similarly, only the upper edge of R_1, the right edge of R_{10} and R_{13}, and the bottom edge of R_{14} are selected. When two of these edges are selected, e.g., the upper edge of R_1 and the right edge of R_{10}, one window is fixed as shown in Figure 10 (c), and this window satisfies all the constraints. Therefore, for each selected edge, a dotted line is created as shown in Figure 10 (d). The distance between an edge and the corresponding dotted line is W. In this way, we set up a grid over the given region, and all windows that satisfy the constraints have their edges on this grid. The solid red dots in Figure 10 (d) are the possible locations for the left bottom corner of a window. In this example, we only need check 20 windows. The algorithm is summarized as follows. X and Y are the coordinates of the left bottom corner of the given region. B is the sliding step of the previous dissection, and *center_dens* is the density of the center area.

Algorithm Exact_Density(X, Y, B, *center_dens*)

For all rectangles *rect* in the given region
    If $rect.x_l \in [X, X + B]$, add $rect.x_l$ to *Llist*;
    If $rect.x_h \in [X + W, X + W + B]$, add $rect.x_h$ to *Rlist*;
    If $rect.y_l \in [Y, Y + B]$, add $rect.y_l$ to *Blist*;
    If $rect.y_h \in [Y + W, Y + W + B]$, add $rect.y_h$ to *Ulist*;

Setup a *grid* based on *Llist*, *Rlist*, *Blist* and *Ulist*;

Calculate density for each grid tile except the center tile;
Set the density of the center tile as *center_dens*;

For each grid point (i, j) in the region (X, Y, B)
    WinDens = density of the window whose left bottom corner is on $grid[i][j]$;
    If ($Windens > U$)
        U = *Windens*;
Return U;

When Exact_Density is called, the region size is very close to the window size. So the number of rectangle edges within the boundary tiles is very limited, which means that the grid size in Exact_Density won't be big. Therefore, only a small amount of windows need to be checked, and this guarantees a short running time.

REMARK In this paper, we focus on identifying the maximum density window. In reality, we may also want to find regions whose density is larger than the given density bound. By changing U to the given density bound, our algorithm can be used to serve this task as

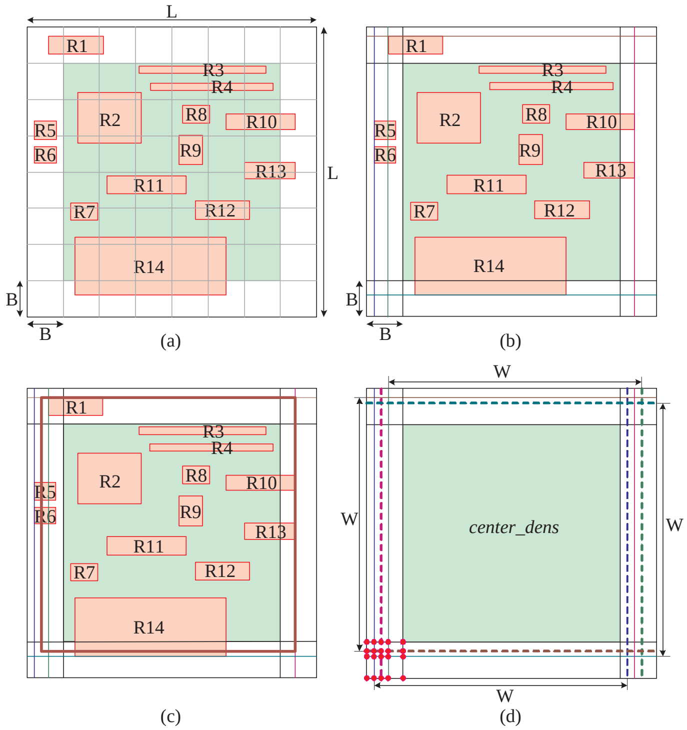


Figure 10: (a) A region with 14 rectangles. (b) The left edges of R_5, R_6 and R_1 are within the first column. Therefore, two lines (the blue line and the green line) are created. Similarly, one line is created for the upper edge of R_1, one is for the right edge of R_{10} and R_{13}, and one is for the bottom edge of R_{14}. (c) When the upper edge of R_1 and the right edge of R_{10} are selected, one window is fixed, and it satisfies all the constraints. (d) A grid is set up for Exact_Density. The distance between a solid line and the corresponding dotted line is W.

well.

5. Experimental Results

We implemented our algorithm in C on an AIX workstation (1.2GHz) with 2GB memory. The test cases are derived from industry designs. For comparison purpose, we also implemented the two algorithms in [6]. One is ALG3 which is the fastest exact algorithm in [6]. The other is "Multilevel Density Analysis" (MDA) which is an approximate algorithm and it terminates when the reported density is within an error bound of the actual maximum density. We set the error bound as 2%. Table 1 summarizes the layout dimension and the number of rectangles in each test case. Table 2 and 3 show the test results with two window sizes $24um$ and $32um$, respectively. The algorithm starts with a sliding step of $\frac{W}{4}$, and set the recursive partition factor $k = 4$. For ALG3, although it can identify the exact maximum density window, the running time is very long, measured by hours or days. Such a long running time is not acceptable in real designs. We killed the tests when the running time was longer than 24 hours. (To verify the correctness of our implementation, we also finished the test case *Test*3. When the window size was $32um$, it took more than 2 days to finish.) On the other hand, the algorithm MDA is fast, but it cannot find the exact maximum density, and the running time will increase with smaller error bounds. For all test

Table 1: Test casess

Testcase	Layout Area (um^2)	#rectangles
Test1	576x576	191,967
Test2	576x576	360,799
Test3	512x512	449,828
Test4	1248x1216	762,412
Test5	512x512	1,375,605
Test6	992x992	3,106,559
Test7	992x992	4,632,445
Test8	992x992	5,033,242
Test9	1216x1216	5,287,136
Test10	992x992	5,583,589

cases, our algorithm can report the exact maximum density, and its running time is even shorter than that of MDA.

One big advantage of our algorithm is that it fully utilizes the results from previous iterations. For each region processed by Detail_Density or Exact_Density, the center area gets its density information directly from the previous iteration, and this saves a lot of computations. Especially, when the sliding step becomes smaller, the percentage of the center area dominates the whole region, and the rectangles falling on the boundaries are very limited. The MDA [6] also adopts the recursive partition approach. But in each iteration, a region is partitioned into smaller tiles with a finer grid, and the density calculation has to be called for each grid tile. None of the previous density calculation can be reused.

Table 2: Test results with a window size 32*um*

Test	ALG3[6]		MDA[6] ($err \leq 2\%$)		Our Alg	
	Max Dens	CPU (s)	Max Dens	CUP (s)	Max Dens	CPU (s)
Test1	57.54%	22027	58.41%	300	57.54%	2
Test2	42.83%	83254	43.26%	224	42.83%	4
Test3	28.99%	51h 30m	29.32%	170	28.99%	42
Test4	84.48%	46231	85.52%	821	84.48%	3
Test5	-	$> 24h$	19.61%	197	19.35%	110
Test6	-	$> 24h$	56.33%	136	55.57%	39
Test7	-	$> 24h$	47.95%	687	47.34%	195
Test8	-	$> 24h$	26.93%	138	26.64%	73
Test9	-	$> 24h$	86.88%	135	85.90%	15
Test10	-	$> 24h$	39.30%	346	38.96%	74

Table 3: Test results with a window size 24*um*

Test	ALG3[6]		MDA[6] ($err \leq 2\%$)		Our Alg	
	Max Dens	CPU (s)	Max Dens	CUP (s)	Max Dens	CPU (s)
Test1	67.23%	10587	68.06%	436	67.23%	1
Test2	47.40%	42289	48.02%	817	47.40%	3
Test3	29.82%	93242	30.20%	201	29.82%	32
Test4	84.42%	22876	85.57%	1421	84.42%	5
Test5	-	$> 24h$	21.05%	166	20.94%	88
Test6	-	$> 24h$	58.62%	270	57.56%	28
Test7	-	$> 24h$	50.82%	779	50.04%	96
Test8	-	$> 24h$	28.49%	128	28.08%	64
Test9	-	$> 24h$	88.24%	104	86.84%	15
Test10	-	$> 24h$	43.51%	213	42.92%	46

Moreover, Theorem 1 greatly reduces the number of regions that need to be considered for density calculation. For example, if the original sliding step is $\frac{W}{4}$, and the partition factor $k = 4$. Then in the 3^{rd} dissection, the sliding step is $\frac{W}{4^3} = \frac{W}{64}$, and the error bound from Theorem 1 is $\frac{1}{64} - (\frac{1}{128})^2 \approx 1.556\%$. In other words, after two dissections, we only need consider windows whose density is very close to the actual maximum density (the density difference is less than 1.556%). This is extremely effective when the density distribution has some density hot spots, which is a must check in the post-design stage. The experiments are to identify maximum density windows, while the minimum density windows can be handled similarly.

6. Conclusion

Density calculation is a fundamental operation in many manufacturing processes. As the device shapes keep shrinking, the minimum feature size will become even smaller, while the chip size continues to grow, the demand on the accuracy of density calculation will keep increasing. In this paper, we propose a fast exact algorithm to identify the maximum density for a given layout. (Minimum density checking can be extended similarly.) Compared with the existing exact algorithms, our algorithm reduces the running time from days/hours to a few minutes/seconds. And it is even faster than the existing approximate algorithm in the literature.

7. References

[1] B. Biswas, Modeling in-die process variation with accuracy, http://www.eetimes.com/story/OEG20040115S0027.

[2] L. Deng, K. Chao, H. Xiang, and D.F. Wong, Coupling-aware dummy metal insertion for lithography, Proc. Asia and South Pacific Design Automation Conf., Jan. 2006.

[3] A. B. Kahng, G. Robins, A. Singh, H. Wang and A. Zelikovsky, Filling and Slotting: Analysis and Algorithm, Proc. ACM/IEEE Intl. Symp. on Physical Design, pp. 95-102, April 1998.

[4] A. B. Kahng, G. Robins, A. Singh and A. Zelikovsky, New Multilevel and Hierarchical Algorithms for Layout Density Control, Proc. Asia and South Pacific Design Automation Conf., pp. 221-224, Jan. 1999.

[5] A. B. Kahng, G. Robins, A. Singh and A. Zelikovsky, New and Exact Filling Algorithms for Layout Density Control, Proc. IEEE Intl. Conf. on VLSI Design, pp. 106-110, Jan. 1999.

[6] A. B. Kahng, G. Robins, A. Singh and A. Zelikovsky, Filling Algorithms and Analyses for Layout Density Control, IEEE Transactions on Computer-Aided Design 18(4), pp. 445-462, 1999.

[7] Y. Chen, A. B. Kahng, G. Robins, and A. Zelikovsky, Monte-Carlo Algorithms for Layout Density Control, Proc. Asia and South Pacific Design Automation Conf., pp. 523-528, Jan. 2000.

[8] Y. Chen, A. B. Kahng, G. Robins and A. Zelikovsky, Practical Iterated Fill Synthesis for CMP Uniformity, Proc. ACM/IEEE Design Automation Conf., pp. 671-674, June 2000.

[9] Y. Chen, A. B. Kahng, G. Robins and A. Zelikovsky, Hierarchical Dummy Fill for Process Uniformity, Proc. Asia and South Pacific Design Automation Conf., pp. 139-144, Jan. 2001.

[10] Y. Chen, A. B. Kahng, G. Robins and A. Zelikovsky, Closing the Smoothness and Uniformity Gap in Area Fill Synthesis, Proc. ACM/IEEE Intl. Symp. on Physical Design, pp. 137-142, April 2002.

[11] Y. Chen, A. B. Kahng, G. Robins, and A. Zelikovsky, Area Fill Synthesis for Uniform Layout Density, IEEE Trans. on CAD, vol 21, No. 10, pp 1132-1147, Oct. 2002.

[12] Y. Chen, P. Gupta, and A. B. Kahng, Performance-Impact Limited Dummy Fill Insertion, Proc. SPIE Conf. on Design and Process Integration for Microelectronic Manufacturing, pp. 75-86, Feb. 2003.

[13] Y. Chen, A. B. Kahng, G. Robins, A. Zelikovsky, and Y. H. Zheng, Data Volume Reduction in Dummy Fill Generation, Proc. Design Automation and Testing in Europe, pp. 868-873, March 2003.

[14] Y. Chen, A. B. Kahng, G. Robins, A. Zelikovsky, and Y. Zheng, Area Fill Generation With Inherent Data Volume Reduction, Proc. Design Automation and Testing in Europe, Munich, Germany, pp. 868-873, March 2003.

[15] P. Gupta, A. B. Kahng, Y. Kim, D. Sylvester, Self-Compensating Design for Focus Variation, Proc. Design Automation Conf., pp. 365-368, June, 2005.

[16] A. B. Kahng, IC Layout and Manufacturability: Critical Links and Design Flow Implications, IEEE Intl. Conf. on VLSI Design, pp. 100-105, Jan. 1999.

[17] R. Tian, D. F. Wong, R. Boone, Model-based dummy feature placement for oxide chemical-mechanical polishing manufacturability, Proc. Design Automation Conf, pp 667-670, 2000.

[18] R. Tian, X. Tang, D. F. Wong, Dummy feature placement for chemical-mechanical polishing uniformity in a shallow trench isolation process, Proc. International Symposium on Physical Design, pp 118-123, 2001.

[19] K. Veelenturf, The road to better reliability and yield embedded DFM tools, Proc. of the conference on design, automation and test in Europe, pp.67-69, 2000.

[20] X. Wang, C. C. Chiang, J. Kawa and Q. Su, A Min-Variance Iterative Method for Fast Smart Dummy Feature Density Assignment in Chemical-Mechanical Polishing, Proc. International Symposium on Quality Electronic Design, pp. 258-263, 2005.