Monte-Carlo Algorithms for Layout Density Control

Y. Chen, A. B. Kahng, G. Robins, A. Zelikovsky

(UCLA, UVA and GSU)

http://vlsicad.cs.ucla.edu

Supported by Cadence Design Systems, Inc.

ASPDAC, 2000, Japan

Outline

- CMP
- Filling problem
- Approaches to filling synthesis
- A new Monte-Carlo approach
- Synthesis of filling patterns
- Computational experience
- Summary and research directions

ASPDAC, 2000, Japan

CMP and Pattern Density Model

- Chemical -Mechanical Polishing (CMP)
  - Rotating pad polishes each layer on wafers to achieve planarized surfaces
  - Non-uniform feature density affects CMP:
    - overpolishing of empty areas (dishing) and underpolishing of dense areas
- Model for oxide planarization via CMP

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

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

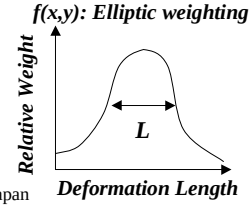
- Spatial Density Model

$$d(x,y) = \frac{featureArea}{layoutArea}$$

- Effective Density Model

$\rho(x,y) = IFFT [FFT [d(x,y)] \bullet FFT [f(x,y)]]$

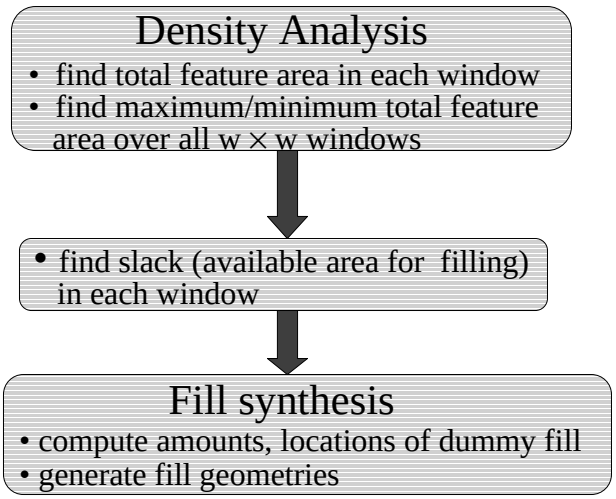
f(x,y): Elliptic weighting filter



Layout Density Control

- Density Rules
  - Foundry rules control local feature density for w ×w windows
    - e.g., for metal layer every 2000um × 2000um window must be between 35% and 70% filled
  - Filling = insertion of "dummy" features to improve layout density
    - typically via layout post-processing in PV / TCAD tools
    - affects design characteristics (e.g., RC extraction) ⇒ accurate knowledge of filling is required during physical design
- Objectives of the Layout Density Control
  - Min-Var: minimizing window density variation, given the upper bound of the window densities
  - Min-Fill: minimizing the total amount of filling features, given the range density variation

Layout Density Control Flow



ASPDAC, 2000, Japan

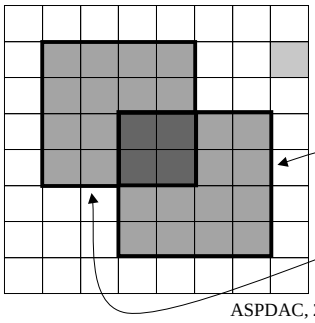
Filling Problem

- **Given** design rule-correct layout of *k* disjoint rectilinear features in $n \times n$ region, window size $w < n$ and window density upper bound U
- **Find** design rule-correct **filled** layout
  - no fill geometry is added within distance B of any layout feature
  - no fill is added into any window that has density $\geq U$
  - satisfy the given objective

ASPDAC, 2000, Japan

Fixed-Dissection Regime

- Feature density is imposed only for ***fixed*** set of $w \times w$ windows
- Layout partitioned by r^2 fixed dissections
- Each $w \times w$ window is partitioned in r^2 ***tiles***



fixed ***r***-dissection
with ***r*** = 4

Filling Problem in Fixed-Dissection Regime

- **Given**
  - fixed ***r***-dissection of layout
  - total feature area ***area***[T] in each tile T
  - ***slack***[T] = area available for filling in T
  - maximum window density ***U***
- **Find** total fill area p[T] to add in each T
  - any $w \times w$ window W has density $\leq U$
  - satisfy the given objective.

Two Recent LP Approaches

- *Min-Var Objective* (Kahng et al's approach):

  * Objectives

    minimizing window density variation given the upper bound of the window densities, based on the spatial density model.

  * LP formulation

  – Maximize M (lower bound on window density)

  – Subject to:

    • For any tile T:

      $0 \le p[T] \le \textbf{pattern} \times slack[T]$

    • For any window W:

      $\sum_{T \in W} p[T] \le U \times w2$

      $M \le \sum_{T \in W} (p[T] + area[T])$

  (***pattern*** = max achievable pattern area density)

- *Min-Fill Objective* (Tian et al's approach):

  * Objectives

    minimizing the total amount of filling features, given the range density variation, based on the effective density model.

  * LP formulation

  – Minimize the total amount of filling features

  – Subject to:

    • For any tile T:

      $0 \le p[T] \le \textbf{pattern} \times slack[T]$

      $LowerBound \le \rho_0(T) \le UpperBound$

      $UpperBound - LowerBound \le \varepsilon$

    ($\rho_0(T)$ = the effective density of tile T)

    ($p(T)$ = the spatial density of tile T)

ASPDAC, 2000, Japan

Weakness of the LP Approaches

- Poor scaling properties of the LP formulation;
- Impractical running time of the LP methods when the size of layout is large;
  – The number of variables and the number of constraints in the linear program are both $O((n \cdot r/w)^2)$ (n: #rectangles, w/r: tile size).
- The solution for the given number r of dissections is not necessarily the optimal solution for 2r dissection;
- Rounding is another reason of deviation of the LP solution from the optimum: when the tile size is sufficient small, then the problem becomes an instance of integer programming and the rounding error becomes essential.

ASPDAC, 2000, Japan

Monte-Carlo Algorithm

Calculate the *slack(T)* of each tile and the density of each window;

For each tile T initialize

insert_in(T) = 0

priority(T) = f(U, slack(T), MaxWin(T))

While the sum of tile priorities is positive

Select a random tile *T* according to its priority;

If each window's density after insertion is less than *U*

Update *insert_in(T), priority(T)* and *slack(T);*

Update each window's density

Else

Update the priorities of tiles covered by the overfilled window;

For each tile Do

Randomly perturb sequence of grid positions;

Insert the *insert_in(T)* unit-fill geometries into the random grid position;

- *Key Decisions:* ***Priorities of tiles****;* ***Updating Scheme.***

ASPDAC, 2000, Japan

Monte-Carlo Priorities for Tiles

- **Priority of tile *T*** (probability of choosing *T* for next fill)
  - ***Slack priority*** proportional to slack of tile T

(Intuition: the tiles are selected with probability proportional to its empty area.)

  - ***Maximal priority*** proportional to *U - MaxWin(T)*

MaxWin(T) = maximum density of window containing T

(Intuition: first insert fill into tiles for which the upper density is less likely to constrain the filling.)

  - ***Minimal priority*** proportional to *U - MinWin(T)*

MinWin(T) = minimum density of window containing T

(Intuition: each insertion of a filling geometry increases the current minimum window density with high probability.)

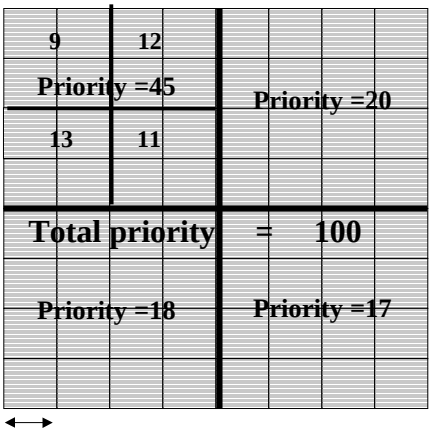
- **Higher Priorities**
  - easily accomplished, e.g., by raising priorities to the power 2 or 4 before normalizing them.
  - ***Monte-Carlo*** algorithm ⇒ ***Greedy*** algorithm

ASPDAC, 2000, Japan

Updating Priorities

- *(H1):* update priorities of all affected tiles
- *(H2):* update priorities only of ***locked*** tiles
  - contained in window of density = *U*
- data structure: *quadrisection*
  - recursively partitions design into 4 quadrants
  - region priority = sum of its quadrant priorities
    = sum of its tiles priorities

Priority Update

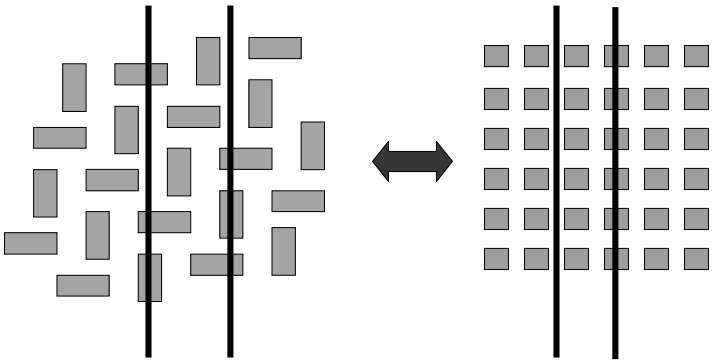


O(log (# tiles)) per insertion

Synthesis of Filling Patterns

- Given area of filling pattern p, insert filling pattern into tile T ***uniformly*** over available area
- Desirable properties of filling pattern
  - uniform coupling to long conductors
  - either grounded or floating

Basket-Weave Pattern



Each vertical/horizontal crossover line has same overlap capacitance to fill

Experimental Testbed

- GDSII → CIF
- Internally developed geometric processing engines
- All runtimes are CPU seconds, Sun Ultra-10
- Test cases

testcase	L1	L2	L1x4	L2x4
layout size	125,000	112,000	250,000	224,000
# rectangles	49,506	76,423	198,024	305,692

Performance of varying tile selection priorities and updating schedules

Layout Info			HEURISTIC I				HEURISTIC II					
			Min_Pri		Max_Pri		Min_Pri		Max_Pri		SLK_Pri	
T/W/r	Max	Min	Min	CPU	Min	CPU	Min	CPU	Min	CPU	Min	CPU
L1/31/2	0.2020	0.1054	**0.1935**	3.30	**0.1933**	2.21	**0.1934**	1.01	**0.1932**	1.00	**0.1925**	0.97
L1/31/5	0.2144	0.0909	**0.1995**	57.0	**0.1987**	32.3	**0.1987**	2.08	**0.1984**	2.07	**0.1967**	2.08
L1x4/31/2	0.2107	0.0873	**0.1513**	13.3	**0.1513**	10.0	**0.1512**	4.72	**0.1504**	4.66	**0.1494**	4.64
L1x4/31/5	0.2146	0.1010	**0.1845**	187	**0.1844**	109	**0.1833**	7.67	**0.1830**	7.65	**0.1816**	7.57
L2/28/2	0.1807	0.0506	**0.1135**	4.70	**0.1132**	3.98	**0.1130**	1.67	**0.1141**	1.68	**0.1130**	1.67
L2/28/5	0.2376	0.0720	**0.1712**	100	**0.1670**	78	**0.1653**	3.92	**0.1658**	3.91	**0.1683**	3.99
L2x4/28/2	0.2232	0.0501	**0.1721**	44.5	**0.1647**	34	**0.1671**	13.8	**0.1645**	13.9	**0.1612**	13.5
L2x4/28/5	0.2377	0.0500	**0.1684**	516	**0.1648**	373	**0.1653**	21.3	**0.1628**	21.3	**0.1603**	21.0

- Tradeoff between runtime and accuracy:
  - the fastest slack priority has the lowest accuracy
  - the slowest min priority has the highest accuracy.
- The best choice seems to be maximal priority:
  - almost as accuracy as the minimal priority
  - considerably fast

Effect of the power 2 and 4 priorities

LAYOUT INFO			MIN_PRI	MIN_PRI^2	MIN_PRI^4	MAX_PRI	MAX_PRI^2	MAX_PRI^4
T/W/r	MaxDen	MinDen	MinDen	MinDen	MinDen	MinDen	MinDen	MinDen
L1/31/2	0.20201	0.10548	**0.19393**	**0.19546**	**0.19463**	**0.19345**	**0.19271**	**0.18247**
L1/31/3	0.20712	0.09683	**0.19223**	**0.19265**	**0.19245**	**0.19205**	**0.19251**	**0.19216**
L1/31/4	0.21248	0.09369	**0.19814**	**0.19893**	**0.19661**	**0.19643**	**0.19655**	**0.19592**
L1/31/5	0.21449	0.09097	**0.19869**	**0.19961**	**0.19855**	**0.19793**	**0.19895**	**0.19664**
L2/28/2	0.18076	0.05065	**0.11150**	**0.11198**	**0.11213**	**0.11437**	**0.11400**	**0.11673**
L2/28/3	0.22651	0.05125	**0.14697**	**0.15168**	**0.15715**	**0.14625**	**0.14816**	**0.15261**
L2/28/4	0.21827	0.08072	**0.17688**	**0.17777**	**0.17800**	**0.17677**	**0.17737**	**0.17799**
L2/28/5	0.23764	0.07203	**0.16627**	**0.16671**	**0.16842**	**0.16675**	**0.16539**	**0.16889**

- Increases the accuracy of filling result
- Get the negligible effect on the runtime.

Comparison to LP-Based Approach

			LP METHOD		HEURISTIC_I				HEURISTIC_II			
Layout Info					Filling_1		Filling_2		Filling_1		Filling_2	
T/W/r	Max	Min	Min	CPU	Min	CPU	Min	CPU	Min	CPU	Min	CPU
L1/31/2	0.2020	0.1054	**0.2011**	0.3	**0.1935**	3.30	**0.1803**	0.1	**0.1934**	1.01	**0.1783**	0.17
L1/31/5	0.2144	0.0909	**0.2032**	1.78	**0.1995**	57.0	**0.1921**	0.5	**0.1987**	2.08	**0.1963**	0.49
L1/8/2	0.2696	0.0208	**0.1596**	0.2	**0.1586**	6.32	**0.1324**	0.2	**0.1586**	2.29	**0.1486**	0.36
L1/8/5	0.2721	0.0276	**0.1490**	0.8	**0.1482**	67.4	**0.1356**	9.2	**0.1482**	3.90	**0.1367**	1.42
L1/4/2	0.2825	0.0054	**0.1673**	0.1	**0.1677**	7.31	**0.1176**	0.9	**0.1677**	2.79	**0.1114**	3.97
L1/4/5	0.2823	0.0039	**0.1292**	0.6	**0.1238**	88.9	**0.1137**	22	**0.1238**	8.58	**0.1123**	45.8
L1x4/31/2	0.2107	0.0873	**0.1584**	0.2	**0.1513**	13.3	**0.0934**	0.2	**0.1512**	4.72	**0.0902**	0.87
L1x4/31/4	0.2148	0.1077	**0.1981**	0.6	**0.1919**	90.5	**0.1147**	2.3	**0.1902**	6.5	**0.1127**	3.59
L1x4/31/5	0.2146	0.1010	**N/A**	1.2	**0.1845**	187	**0.1124**	3.2	**0.1833**	7.67	**0.1094**	4.84

- The accuracy of the Monte-Carlo method is very high
  - **The resulting variation is no more than 5% larger than the LP method**
- The Monte-Carlo method is much faster than the LP method
  - when the window size is small and/or the number of fixed dissections is large, the LP method becomes impractical.

Summary of Contributions

- New Monte-Carlo method for filling problem
- Comparison of several variations of Monte-Carlo method based on different priority schemes
- Fast implementation with quad data structures
- Experimental confirmation of drastic improvement over LP-methods in:
  - runtime
  - scalability
  - accuracy

ASPDAC, 2000, Japan

Ongoing Research Directions

- Improved heuristics/exact algorithms for both CMP models
- Solutions to the Filling Problem in the continuous (as opposed to fixed-dissection) regime
- Calibration of our methods with data and density control requirements from industry partners
- Finding a more accurate slack computation method
- Achieving layout density control in a truly hierarchical context.

ASPDAC, 2000, Japan