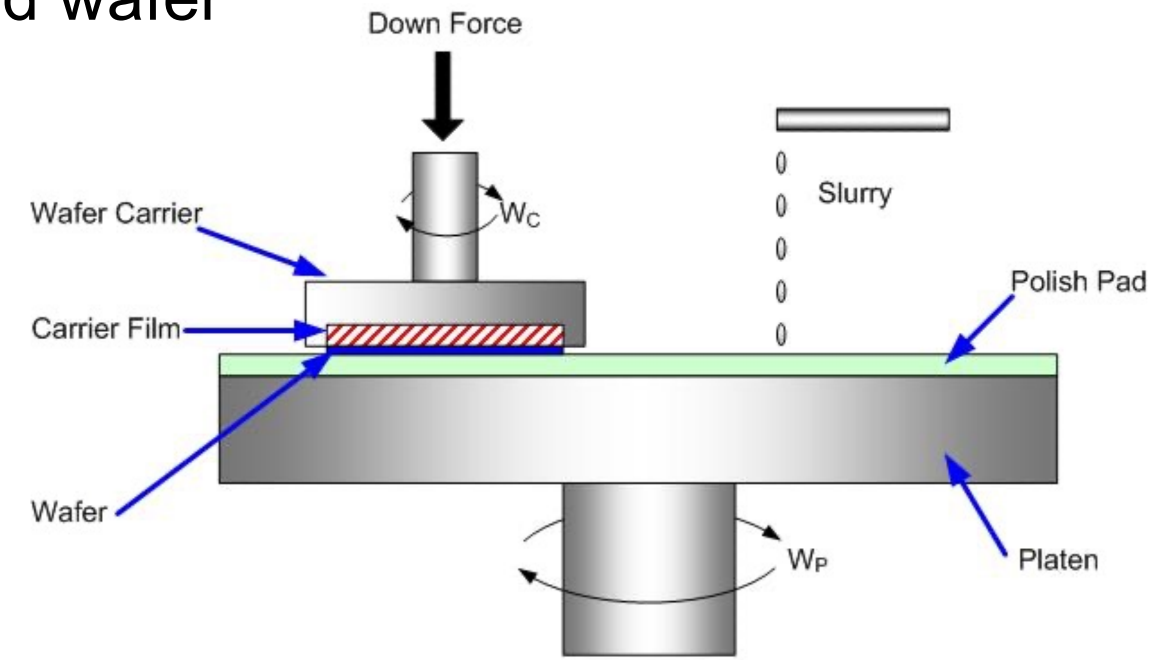
Dummy Fill Insertion

Outline

- Chemical-Mechanical Polishing (CMP)
- Filling Problem in fixed-dissection regime
- LP and Monte-Carlo (MC) approaches
- MC approach with Min-Fill objective
- Iterated MC method
- Computational experience
- Summary

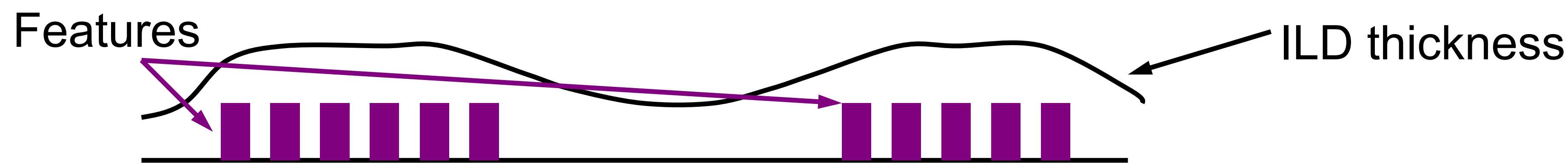
Chemical Mechanical Polishing

- For wafer surface planarization before adding next level of features
- Chemical Mechanical Polishing (CMP)
  - Chemically: abrasive slurry dissolves the wafer layer
  - Mechanically: a dynamic polishing head presses pad and wafer

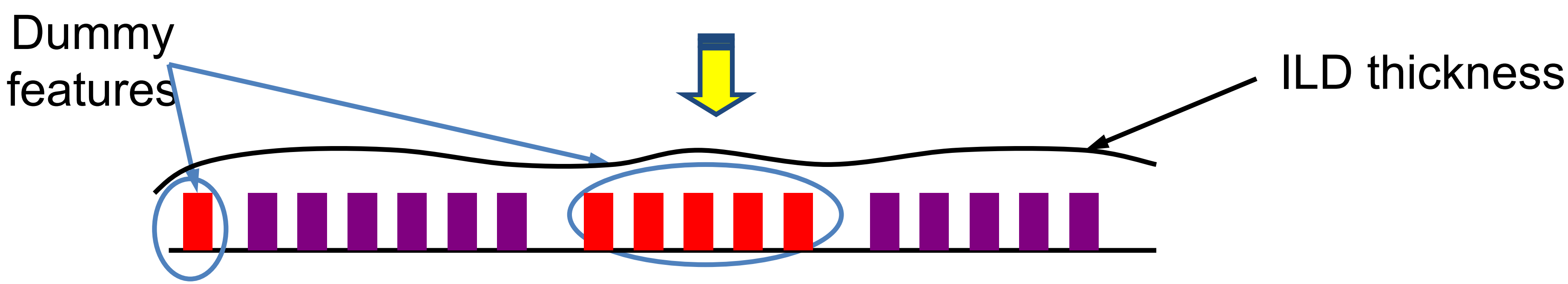


Source: www.ntu.edu.sg

CMP and Interlevel Dielectric Thickness



- Post CMP interlevel-dielectric (ILD) thickness is proportional to feature density
- Uneven features cause polishing pad to deform
- May insert dummy features to reduce variation



Objectives of Density Control

- Want to
  - minimize density variation to optimize post-CMP topography
  - minimize amount of fill to minimize impact on circuit performance
- Difficult to optimize both objectives simultaneously, so minimize one while keeping the other in check
- Objective for Manufacturability (i.e., Min-Var)

  minimize window density variation

  subject to upper bound on window density
- Objective for Design Performance (i.e., Min-Fill)

  minimize total amount of added fill features

  subject to upper bound on window density variation

Filling Problem

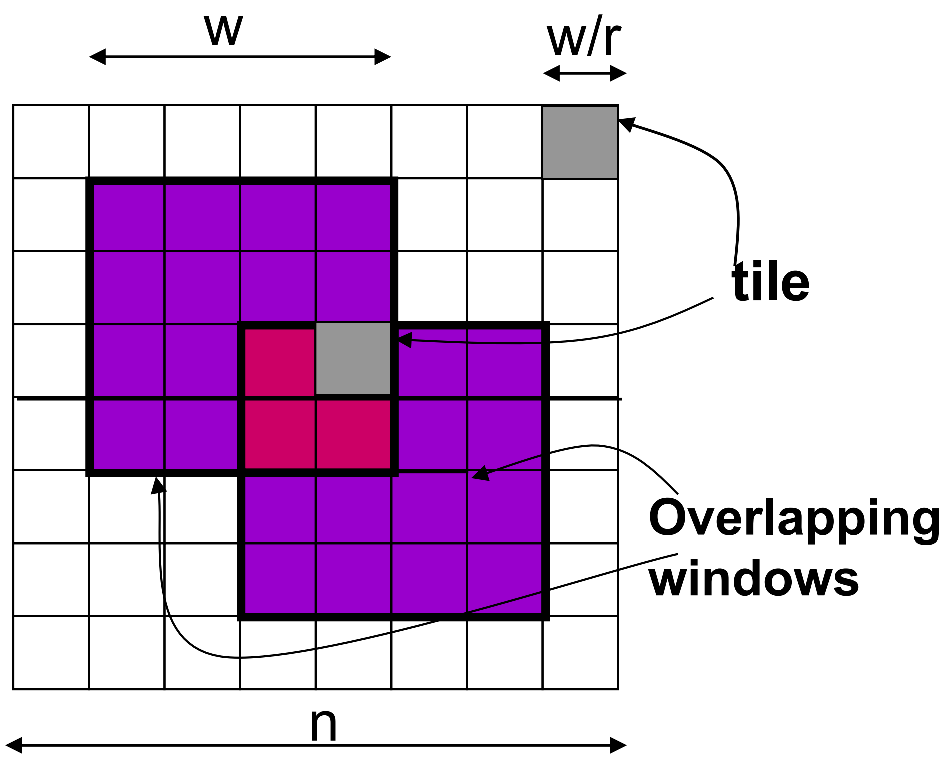
- **Given**
  - **rule-correct layout in $n \times n$ region**
  - **window size = $w \times w$**
  - **window density upper bound U**

- **Fill layout with Min-Var or Min-Fill objective**

  such that *no* fill is added
  - **within buffer distance B of any layout feature**
  - **into any overfilled window that has density $\geq U$**

Fixed-Dissection Regime

- Monitor only a fixed set of $w \times w$ windows
  - "offset" = w/r (example shown: w = 4, r = 4)
- Partition n x n layout with $nr/w \times nr/w$ fixed dissections
- Each $w \times w$ window is partitioned into r^2 tiles



Layout Density Models

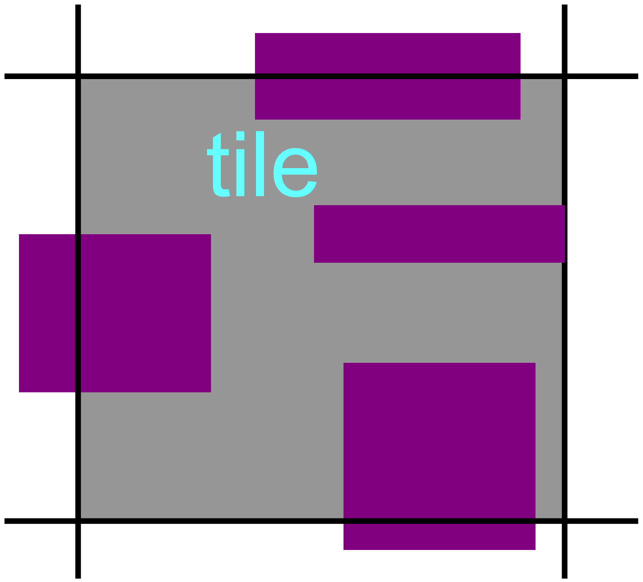
- **Spatial Density Model**

  window density ≈ sum of tiles' feature area

- **Effective Density Model (more accurate)**

  window density ≈ weighted sum of of tiles's feature area

  - **elliptical weights decrease from window center to boundaries**



Linear Programming Approach

- *Min-Var Objective*
  [Kahng+ TCAD99]
  – Maximize: M
  – Subject to:

    For any tile T

    $0 \leq p[T] \leq slack[T]$

    For any window W

    $\sum_{T \in W} (p[T]+area[T]) \leq U$

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

    $p[T]$ = fill area of tile T
  – spatial density model

- ***Min-Fill Objective***
  [Tian+ DAC00]
  - **Minimize:**

    Fill amount = $\sum p[T]$
  - **Subject to:**

    For any tile T

    $0 \leq p[T] \leq slack[T]$

    $LowerB \leq \rho_0(T) \leq UpperB$

    $UpperB - LowerB \leq \varepsilon$

  $\rho_0(T)$ = effective density for tile T
  - **effective density model**

Monte-Carlo Approach with Min-Var Objective [Chen+ ASPDAC00]

- Fill layout randomly
  - Pick the tile for next filling geometry probabilistically
  - Higher priority of a tile ⇒ higher probability to be filled (i.e., Monte-Carlo)
  - Lock a tile if any containing window is overfilled
- Different schemes for setting tile priorities
  - *Slack* of the tile: will fill uniformly randomly
  - *U − max density of any window containing the tile:* tend to fill region as much as possible at the end
  - *U − min density of any window containing the tile:* will fill most underfilled window first (outperforms the other two schemes experimentally)

Monte-Carlo Approach with Min-Var Objective

- Different schemes for amount of fill geometry added per iteration
  - Insert a single filling geometry into a tile (better results)
  - Insert maximum possible filling geometries into a tile (faster)
- 2 possibilities for updating priorities after each iteration
  - Update priorities of all affected tiles (slightly better results)
  - Update priorities only of tiles which belong to any newly locked window (faster)

Monte Carlo-based Filling Algorithm

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 **Do**
    Select a random tile T according to priorities
    $insert\_in(T) = insert\_in(T) + 1$; $slack(T) = slack(T) - unit\_fill$
    If $slack(T) < unit\_fill$ **Then** $priority(T) = 0$
    Else $priority(T) = priority(T) - unit\_fill$
    For each window W containing T **Do**
        $area(W) = area(W) + unit\_fill$
        For each tile $T' \in W$ **Do**
            Update $priority(T')$ according to $area(W)$
For each tile T **Do**
    Randomly perturb sequence of grid positions: $random(i) = 1, \ldots, slack(T)/unit\_fill$
    For $i = 1, \ldots, insert\_in(T)$ **Do**
        Insert a unit-fill geometry into the $random(i)^{th}$ grid position
Output the filled layout

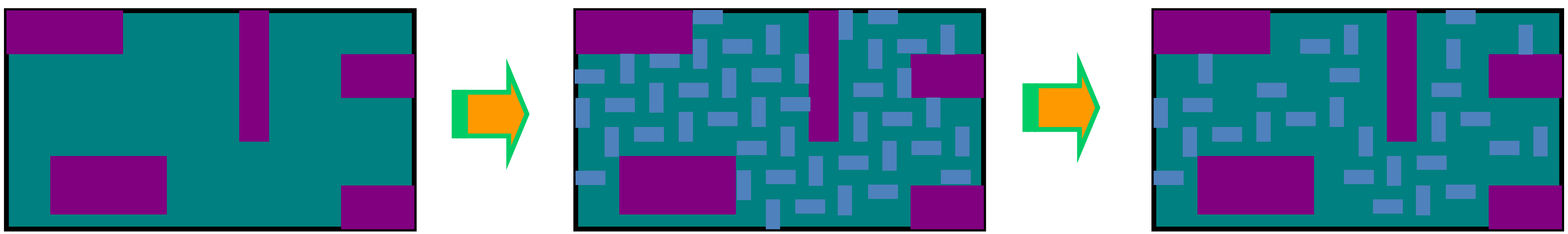
LP vs. Monte-Carlo

- LP
  - large runtime for large layouts
  - r-dissection solution may be suboptimal for 2r dissections
  - essential rounding error for small tiles
- Monte-Carlo
  - very efficient: $O((nr/w)\log(nr/w))$ time
  - scalability: handle large values of *r*
  - accuracy: reasonably high comparing with LP
  - drawback: excessive amount of fill features for Min-Var

- Remark: If we always choose the tile with the highest priority for filling instead, we get a greedy algorithm instead of MC. MC performs better than greedy on average.

Monte-Carlo Approach with Min-Fill Objective [Chen+ DAC00]

- Delete excessive fill
- Delete as much fill as possible while maintaining min window density ≥ *L*



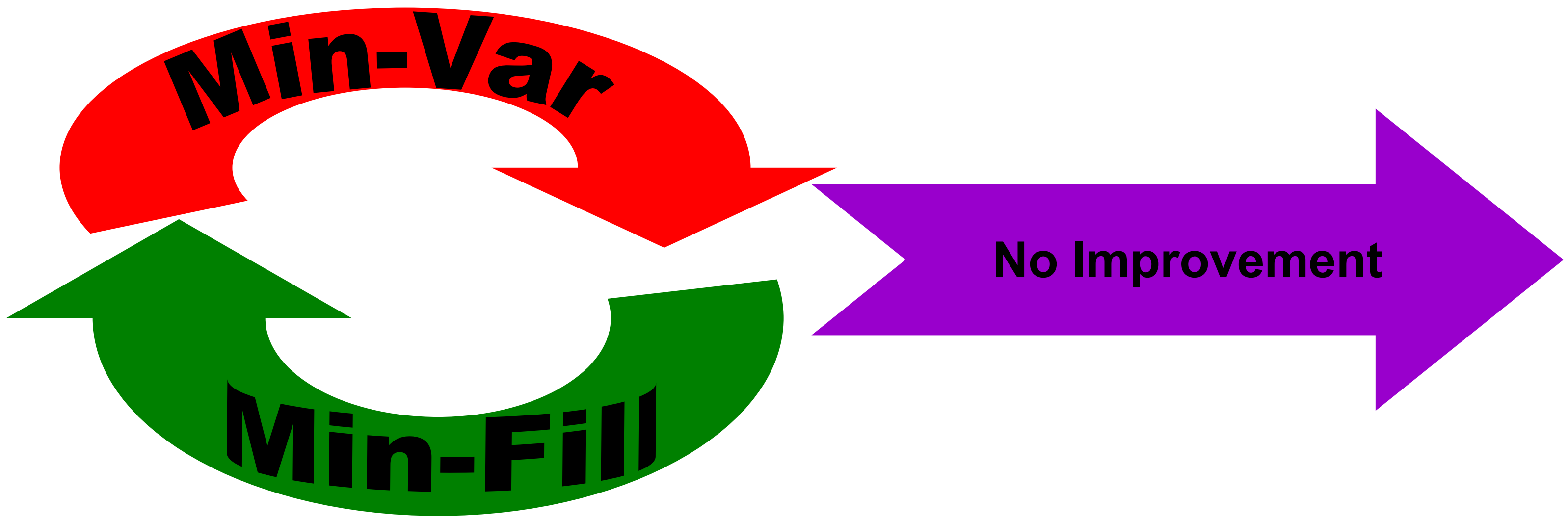
Monte-Carlo Approach with Min-Fill Objective

- Priority: *min density of any window containing the tile − L*

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

Iterated Monte-Carlo Approach

- Repeat forever [Chen+ DAC00]
  - run Min-Var Monte-Carlo with maximum window density *U*
  - exit if no change in minimum window density
  - run Min-Fill Monte-Carlo Algorithm with minimum window density *M*



Computational Experience

- Testbed
  - GDSII input
  - hierarchical polygon database
  - C++ under Solaris
  - open-source code

- **Testcases**

  Metal layers from industry standard-cell layouts

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

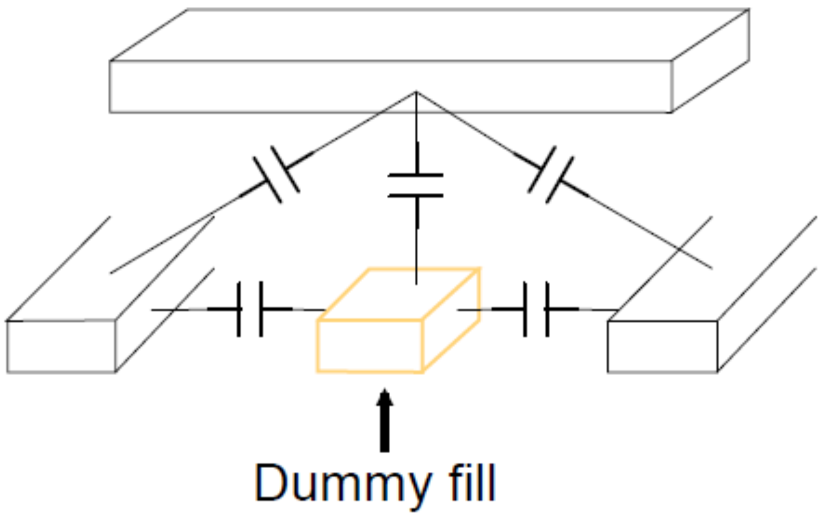
Computational Experience

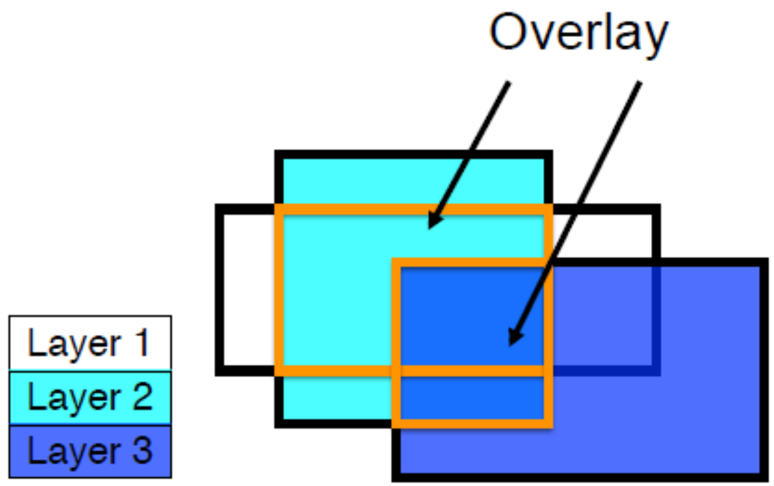
	Orig Density		LP		MC		IMC	
Spatial Density Model								
Test case	**Max**	**Min**	**Min**	**CPU**	**Min**	**CPU**	**Min**	**CPU**
L1/32/8	0.21447	0.10414	**0.19864**	41.5	0.19221	17.3	0.19871	24.8
L2/32/8	0.22648	0.07039	**0.14467**	43	0.13565	24.4	0.14463	68.6
L1x4/32/8	0.21693	0.09657	0.18643	255.7	0.18282	72.3	**0.18648**	111.9
L2x4/32/8	0.22226	0.05776	0.14647	532.6	0.13824	117.7	**0.14655**	469.7
Effective Density Model								
L1/32/8	0.41625	0.16255	**0.3197**	32.4	0.31994	22.3	0.31994	23.9
L2/32/8	0.53585	0.07249	**0.34777**	66.8	0.31153	38.3	0.33858	68.9
L1x4/32/8	0.4327	0.14665	0.28487	171.5	0.28505	90.7	**0.28505**	100.9
L2x4/32/8	0.52179	0.04467	0.34176	637.4	0.30799	165.6	**0.33524**	435.9

- Iterated Monte-Carlo (IMC) approach is more accurate than standard MC approach and faster than LP approach

Extension

- Multi-layer
  - minimize overlay between adjacent layers to reduce coupling capacitance





References

- [Kahng+ TCAD99] Filling Algorithms and Analyses for Layout Density Control
- [Tian+ DAC00] Model-Based Dummy Feature Placement for Oxide Chemical-Mechanical Polishing Manufacturability
- [Chen+ ASPDAC00] Monte-Carlo Algorithms for Layout Density Control
- [Chen+ DAC00] Practical Iterated Fill Synthesis for CMP
- [Lin+ DAC15] High Performance Dummy Fill Insertion with Coupling and Uniformity Constraints