Solutions: Layout-Level Modifications for Performance and Yield

June 25, 1999
Session Overview

- Yield improvement by wire spacing
- Buffer insertion for performance
- Density improvement by filling
- Phase Shift Mask phase assignment
Routing Improvement

- Better yield by reducing the critical area
- Better electrical behavior by changed capacitance
- Insertion of buffer to improve speed
Technology Roadmap

- Scaling of x0.7 every three years
  - .25u .18u .13u .10u .07u .05u
  - 5metal 6metal 7metal 7metal 8metal 9metal
- Interconnect delay dominates system performance
  - consumes 70% of clock cycle
- Cross coupling capacitance is dominating
  - cross capacitance -> 100%, ground capacitance ->0%
  - 90% in .18u
  - harms signal integrity
New Materials Implications

- **Lower dielectric**
  - Reduces total capacitance
  - Doesn’t change cross/ground proportions
- **Copper metalization**
  - Reduces RC delay
  - Avoids electron migration
  - Thinner deposition reduces cross cap
- **Multi layers of routing**
  - Relative routing pitch may increase
  - Room for shielding
Routing Techniques to Improve Performance and Yield

- **Wire spacing**
  - smaller cross coupling capacitance
  - higher ground capacitance
  - smaller probability for shorts
  - better process uniformity

- **Wire widening**
  - smaller resistance
  - smaller probability for opens

- **Wire shielding**
  - lower cross coupling noise

- **Via strengthening**
  - smaller resistance
  - higher reliability
Wire Spacing & Widening Basics

- Routing can relax to manufacturing grid
- Doesn’t increase die area
- Law of big number works: long signals don’t change lengths
- short signals are immaterial
- Higher opportunities for shielding
- Higher opportunities for via strengthening
- Can compile together ECOs of spacing, widening, shielding and global optimization
- Drawbacks: design flow, later ECOs, layout stability
Wire Spacing Optimization
by Linear Optimization

- One dimensional optimization algorithm
- Attractive-repulsive approach
- Addresses spacing by constraints rather then by objective
- Notations (vertical spacing):
  - $h_i$ wire segment at location $y_i$
  - $(h_i, h_j)$ - ordered segments, usually adjacent ones
  - layout constrains (design rules, wire width):
    (1) $y_i - y_i >= R_{ij}$
  - $(h_k, h_l)$ - repulsive wires (signals)
    $a_{kl}$ - repulsive coefficient (signal separation)
    $f(y)$ - degree of proximity (net specifications),
    (2) $f(y) = \frac{\min (y_k-y_l)}{a_{kl}}$
  - $c_{ij}$ - coefficient of adjacent nets objective function:
    (3) $S = \sum_{I,j} c_{il} * (y_i-y_j)$
Problems: minimize (3), maximize (2), s.t. (1)
Addresses spacing indirectly by constraints rather than by objective
Good control on measures of layout (size, wire length)
Wire Spacing Optimization by Iterative Balancing

- One dimensional existence algorithm
- Propagate vacant area through entire layout
- Addresses spacing as an “objective”
- Notations (horizontal spacing):
  - hi wire segment at location xi
  - Li (Ri) - visible segments from left (right)
  - left spacing: ai = min (Li) (xi - x)
  - right spacing: bi = min (Ri) (xi - x)
  - spacing imbalance: mi = I ai - bi I
- Problem:
  find locations yi of all wire segments such that all wires are balanced, ie., mi = 0 for all wires
Wire Spacing Techniques

- Need for smart jogging
- Work on all layers simultaneously
- Move around wires and vias
- Don’t change relative position of wires and vias
- Maintain full hierarchy
- Handle all routing styles
- Handle full chip (striping, multi-processing)
- Support “don’t touch” signals (P/G, clocks)
- Support net specifications (clearance, width)
- Support pair-wise net proximity
Wire Spacing and Layout Methodology

- Routing tools do not optimize for spacing
  - Need awareness of layout designers to specific signals
- Stand-alone spacing
  - layout (GDSII/DEF) -> layout (GDSII/DEF)
- Need tight interface to extraction and timing simulation
- Future: built-in extraction and timing estimates
Data Aspects of Post Layout Optimization

- Need to preserve user’s hierarchy
- Huge data needs striping
  - Minor loss of optimality for large stripes
  - Need work across hierarchy
  - Fix boundary location, “look” beyond cut-line
  - Need propagate net information
- Must support multi-processing for reasonable TAT
Wire Spacing and Shielding

- Pre routing specification
  - Convenient, handled by router
  - Robust but conservative
  - May consume big area
- Post routing specification
  - Area efficient, shield only where really needed
  - Ease task of router
  - Complex design flow
Opportunities for Via Strengthening

- Add cut holes where possible
  - Wire widening may need larger vias
  - “Non square” via cells
- Increase metal-via overhang
  - Non uniform overhang
Wire spacing example

before spacing

after spacing
Economy of Yield

- Enormous cost of mega fabs
- Cycle of under and over manufacturing capacity
  - Full price for every die when under capacity
  - Silicon price in over capacity
- Performance distribution
  - Mean at design point - ordinary parts
  - High (low) tail - high (low) performance parts
  - Common in mP’s and DSP’s
- Global wire spacing shifts mean rightwards
  - Very high return in high volume manufacturing
Defect yield and wire spacing
[Maly, Proc. IEEE, 1990]

- Notations:
  - $Y_i$ - yield of layer $i$
  - $D_i$ - defect density in layer $i$
  - $A_{ci}$ - critical(effective) area of layer $i$
  - yield in layer $i$: $Y_i = \exp(-D_i \cdot A_{ci})$
  - full chip yield: $Y = Y_1 \times \ldots \times Y_n$
  - $r$ - random variable of defect size
    - $f(r)$ - probability density function of $r$
    - $A_{ci}(r)$ - critical area due to defect of size $r$
    - critical area calculation:
      $$A_{ci} = \int (r) (A_{ci}(r) \cdot f(r)) \, dr$$

- Consequence:
  - space as much as possible (post routing)
  - trade off die area and spacing
Session Overview

- Yield improvement by wire spacing
- **Buffer insertion for performance**
- Density improvement by filling
- Phase Shift Mask phase assignment
Performance yield enhancement

- Reducing cross coupling capacitance
  - Increase clock speed
  - Reduce power consumption
  - Smaller capacitive load enables smaller devices
Buffer insertion in distributed RC tree

- Propagation delay increases quadratically with wire length
  - Difficult for synthesis to account for physical layout
  - Dropping buffer resources “almost everywhere” is costly
- Problems:
  - Where to insert a buffer along branches
  - What size of buffer to use at any insertion
- Pre routing timing simulation
  - Use placement information to extract more accurate trees
  - Simulate critical signals before routing
  - Apply optimal buffer insertion algorithm
  - Insert buffer cells into placement
Optimal buffer insertion - single source
[L.P.P.P. van Ginneken, ISCAS, 1990]

- Source (driver) is root, sinks (receivers) are leaves
- Use Elmore delay model:
  - i - left (sink)
  - k - internal node
  - Cj - cap at node j
  - Rj - resistance of branch to node j
  - Pi (Pk) - path from root to node I (k)
  - Rki - sum(j in Pi and Pk) Rj
- Delay from source to sink i:
  - \( D_i = \sum(k) R_{ki} \times C_k \)
  - Denote:
    - \( L_k \) - total cap loaded at node k
  - Then:
    - \( D_i = \sum(j \text{ in } Pi) R_k \times L_k \)
Optimal buffer insertion algorithm
[L.P.P.P. van Ginneken, ISCAS, 1990]

- Notations:
  - $T_i$ - arrival time at sink $i$
  - $T_0$ - departure time at source
    $T_0 = \min(I) \ (T_i - D_i)$
- Goal: inset buffer at branches such that $T_0$ is maximized
- Algorithm paradigm:
  - Bottom up (leaves to root) pass to compute dominant pairs of arrival
time and capacitive load at any node
  - Select at root the pair with maximal $T_0$
  - Top down (root to leaves) pass to retrieve at every node the appropriate pair, implying specific buffer insertion
How to accommodate buffer insertion and wire sizing?

- **Pre Placement**
  - Easy to handle, straightforward implementation by P&R
  - Inaccurate

- **Post placement - Pre Routing (recommended)**
  - Good estimate of net topology
  - Buffer can inserted easily

- **Post Routing**
  - Most accurate
  - Difficult to handle, needs compaction, wire widening and spacing
Generalization of optimal buffer insertion

• Same algorithmic paradigm as single source
• Multi sources [Lillis-Cheng, TCAD, 1999]
  – Use for busses
  – Use bidirectional buffer
• Combine with wire sizing [Lillis-Cheng-Lin, JSSC, 1996]
  – Optimizes power-delay trade-off
Session Overview

• Yield improvement by wire spacing
• Buffer insertion for performance
• Density improvement by filling
• Phase Shift Mask phase assignment
Density Rules

- Modern foundry rules specify layout density bounds to minimize impact of CMP on yield
- Density rules control local feature density for $w \times w$ windows
  - e.g., for metal layer every $2000\text{um} \times 2000\text{um}$ 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 vital design characteristics (e.g., RC extraction)
  - accurate knowledge of filling is required during physical design and verification
**Layout Density Control Flow**

**Density Analysis**
- find total feature area in each window
- find maximum/minimum total feature area over all $w \times w$ windows

**Fill synthesis**
- find slack (available area for filling) in each window
- compute amounts, locations of dummy fill
- generate fill geometries
Exact Max-Density Window Analysis

• For each pivot rectangle \( R \) do
  – find density of \( w \times w \) window \( W \) that abuts \( R \) on top and right
  – while \( W \) intersects \( R \) do
    • slide \( W \) right till intersection with other rectangle edge
    • record changes in density

• \( O(k^2) \) algorithm for \( k \) rectangles
Fixed $r$-Dissection Regime

- Feature area density bounds enforced only for fixed set of $w \times w$ windows
- Layout partitioned by $r^2$ distinct fixed dissections
- Each $w \times w$ window is partitioned in $r^2$ tiles
Drawbacks of Fixed $r$-Dissection Analysis

• If all $w \times w$ windows of fixed $r$-dissection have density $\leq U$, there may be floating $w \times w$ window with density $\min\{1, U + 1/r - 1/(4r^2)\}$

• Fixed-dissection algorithm is inaccurate

• Exact algorithm is slow $= O(k^2)$
Shrunk and Bloated Windows

- **Standard window** = fixed $r$-dissection $w \times w$ window
- **Floating window** = arbitrary $w \times w$ window
- **Bloated window** = standard window bloated by one tile
- **Shrunken window** = standard window shrunk by one tile
- Any floating window is contained in one bloated window and contains one shrunk window
Multilevel Approach

• **Estimation:**
  – max floating window density ≤ max bloated window density
  – min floating window density ≥ min shrunk window density

• **Zooming:**
  – remove standard windows in underfilled bloated windows
  – subdivide remaining tiles and find area of new bloated windows

• Terminate subdivision when either:
  – # of rectangles is small (run exact density analysis), or
  – \((\text{max bloated density})/(\text{max standard density}) \leq \varepsilon\) (say, \(\varepsilon=1\%\))
Multilevel Algorithm

**Tiles** = list of all windows (r = 1)

**Accuracy** = \( \infty \)

While **Accuracy** > 1 + \( \varepsilon \)

- find area in each bloated and standard window
- \( \text{MAX} = \text{max area of standard window} \)
- \( \text{BMAX} = \text{max area of bloated window} \)

refine **Tiles** = list of tiles from bloated windows of area \( \geq \text{MAX} \)

subdivide each tile in **Tiles** into 4 subtiles

**Accuracy** = \( \text{BMAX} / \text{MAX} \)

Output max standard window density = \( \text{MAX}/ w^2 \)
Runtime of Multilevel Algorithm

- Each iteration decreases difference in area between bloated and standard window by half
- Original difference is $3w^2$
- Main loop terminates after $t$ iterations:
  \[ \frac{3w^2}{2^t} \leq 2\epsilon \]
- Maximum $t$ is $O(\log(w/\epsilon))$
- Runtime is $O((n/w \cdot \log(w/\epsilon))^2)$
Filling Problem

- **Given** design rule-correct layout of $k$ disjoint rectilinear features in $n \times n$ region
- **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 $\geq U$
  - minimum window density in the filled layout is maximized (or has density $\geq$ lower bound $L$)
Filling Problem in Fixed-Dissection Regime

• **Given**
  – fixed \( r \)-dissection of layout
  – feature *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 \( T \) s.t.
  any \( w \times w \) window \( W \) has density \( \leq U \) and
  \( \min_{W} \bigcup_{T \in W} (\text{area}[T] + p[T]) \) is maximized
Fixed-Dissection LP Formulation

- Maximize $M$ (lower bound on window density)
- subject to:
  - For any tile $T$: $0 \leq p[T] \leq \text{pattern} \times \text{slack}[T]$
  - For any window $W$:
    $$\sum_{T \in W} p[T] \leq U \times w^2$$
    $$M \leq \sum_{T \in W} (p[T] + \text{area}[T])$$
($\text{pattern} = \text{max achievable pattern area density}$)
Fixed-Dissection LP Formulation

- One variable and two constraints per tile
- Two constraints per window
Multilevel LP Formulation

- Use multilevel density analysis in LP
- $Tiles[r] =$ list of fixed r-dissection tiles from bloated windows of area $\geq$ MAX
- $Saved\ tiles =$ subdivided $Tiles[r]$ minus $Tiles[r+1]$
- $Saved\ windows =$ all standard windows $W$ for which area is found
- Multilevel LP uses only constraints for $saved\ tiles$ and $saved\ windows$
Multilevel LP Formulation

Saved tiles have different sizes: tiles with more feature area are more subdivided

ML LP has one variable and two constraints per tile and two constraints per window
Floating Deviation LP Formulation

- **Floating deviation** = the difference between max and min floating window density
- Floating deviation
  \[ \leq \text{max bloated window density} - \text{min shrunk window density} \]
- **Floating deviation LP:**
  - For any **bloated** window \( W \):
    \[ p[T] \leq U \times w^2 \]
  - For any **shrunk** window \( W \):
    \[ M \leq \sum_{T \in W} (p[T] + \text{area}[T]) \]
Floating Deviation LP Formulation

- one variable and two constraints per tile
- one constraint per shrunken window
- one constraint per bloated window
Hierarchical Density Control

- Hierarchical filling = master cell filling
Hierarchical LP Formulation

• For any cell instance \( C \) of master cell \( \mathcal{C} \) and tile \( T \),
  \( \gamma[C,T] \) is portion of slack[\( C \)] in intersection of \( C \) with \( T \):
  \( \gamma[C,T] = \text{slack}(C \cap T)/\text{slack}[C] \)

• New variable \( d[C] \) per each master cell \( \mathcal{C} \):
  \( d[C] = \text{filling per master cell } \mathcal{C} \)

• New constraints:
  – For total amount of filling added into tile \( T \):
    \[ p[T] = \bigcap \gamma[C,T] \]
  – For amount of filling added into each master cell \( \mathcal{C} \):
    \[ 0 \leq d[C] \leq \text{pattern} \times \text{slack}[C] \]
Synthesis of Filling Patterns

- Given area of filling pattern $p[i,j]$, insert filling pattern into tile $T[i,j]$ 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
Grounded Pattern

Fill with horizontal stripes, then span with vertical lines
Session Overview

- Yield improvement by wire spacing
- Buffer insertion for performance
- Density improvement by filling
- Phase Shift Mask phase assignment
The Phase Assignment Problem in PSM

Assign 0, 180 phase regions such that
- (dark field) feature pairs with separation $< B$ have opposite phases
- (bright field) features with width $< B$ are induced by adjacent phase regions with opposite phases

$b \equiv$ minimum separation or width, with phase shifting
$B \equiv$ minimum separation or width, without phase shifting
Phase Conflict and the Conflict Graph

- Vertices: features (or phase regions)
- Edges: “conflicts” (necessary phase contrasts)
  (feature pairs with separation < B)
Odd Cycles in Conflict Graph

- Self-consistent phase assignment is not possible if there is an odd cycle in the conflict graph.
- Phase-assignable $\equiv$ bipartite $\equiv$ no odd cycles.

Diagram:

- 0 phase
- 180 phase
- ??? phase
Phase Conflict and the Conflict Graph

- Self-consistent phase assignment is not possible if there is an odd cycle in the conflict graph
- Phase-assignable = bipartite = no odd cycles
- Breaking odd cycles: must change the layout!
  - change feature dimensions, and/or change spacings
  - degrees of freedom include layer reassignment for interconnects
Breaking Odd Cycles

- Must change the layout:
  - change feature dimensions, and/or
  - change spacings
  - PSM phase-assignability is a layout, not verification, issue

DAC99 Tutorial

June 25, 1999
Phase Assignment - Dark Field

- Dark Field (odd-cycle breaking formulation)

- dotted matching line
- green 180-shift
- red conflicts
Phase Assignment - Bright Field

- Bright Field (dense criticality regime)

[Diagram with labeled features: black boundaries, b/w 0 and 180 areas (to be deleted), red odd degree, blue features, green 180-shift]
Phase Assignment: Key Technologies

- Key technologies: incremental gridless routing, incremental compaction
- Issues for custom, hierarchical and reuse-based layout methodologies
Conflict Edge Weight

- Which conflict edges are cheapest to break?
- Critical paths (e.g., in compactor) in x- and y-directions define layout area
- Conflict edges not on critical path: break for free
Minimum-Perturbation PSM Layout

- **Input:** layout in, e.g., .25um design rules
- **Goal:** adjust feature sizes and spacings to new PSM design rules, e.g., .15um
  - keep topology as close to original as possible
- **Application to new design and to migration**
  - assumes existence of a starting layout
  - hope to attain fewer violations in verification, require less manual cleanup of output layout
Compaction-Oriented Approach

- Analyze input layout
- Determine constraints for output layout
  - new PSM-induced (shape, spacing) constraints
- Compact (e.g., solve LP) with min perturbation objective
  - e.g., minimize sum of differences between old and new positions of each edge
- Key: Minimize the set of new constraints, i.e., break all odd cycles in conflict graph by deleting a minimum number of edges.
The T-join Problem

- How to delete minimum-cost set of edges from conflict graph G to eliminate odd cycles?
- Construct geometric dual graph $D = \text{dual}(G)$
- Find odd-degree vertices $T$ in $D$
- Solve the T-join problem in $D$:
  - find min-weight edge set $J$ in $D$ such that
    - all T-vertices has odd degree
    - all other vertices have even degree
- Solution $J$ corresponds to desired min-cost edge set in conflict graph $G$
Optimal Odd Cycle Elimination

Blue features/red conflicts

Conflict graph $G$

T-join odd degree nodes in $D$

Dual graph $D$

DAC99 Tutorial

June 25, 1999
Optimal Odd Cycle Elimination

Assign phases:
only green conflicts left

T-join odd degree nodes in D

conflict graph