Integrated Circuit Layout Reconstruction

Ian Besse, Patrick Campbell, Julianne Chung, Malena Espanol, Mark Iwen, Qingshuo Song

Mentor: Edward Keyes

IMA Mathematical Modeling in Industry Workshop August 10, 2005
Outline

- Project Overview
- GROS (GReedy Orthogonal Simplification) Algorithm
- Numerical Experiments
- Conclusions and Future Work
Project Overview
Goals

- Reduce the number of vertices to conserve storage space
- Alter vertices such that edges are horizontal, vertical, 45 or 135 degree diagonals
- Preserve accurate representation of circuit layout
Given a polygon $\mathbf{P}$, an approximation $\mathbf{Q} = (x_1, y_1), ..., (x_N, y_N)$ to $\mathbf{P}$ has energy:

$$E = N + G(\theta) + F(\mathbf{P}, \mathbf{Q})$$

- $N$ is the number of vertices in $\mathbf{Q}$
- $G(\theta)$ enforces the desirability of horizontal, vertical, 45 degree, and 135 degree lines.
- $F(\mathbf{P}, \mathbf{Q})$ represents $\mathbf{Q}$’s fidelity to $\mathbf{P}$
Heuristic Energy Minimization

Attempting to directly minimize the energy function would be much too slow.

The GROS Algorithm:

- We reduce the number of vertices while maintaining horizontal, vertical, 45 degree and 135 degree lines. This keeps $N$ and $G(?)$ small.

- We allow simplifications within an $e$ band of existing vertices. This keeps $F(P,Q)$ small.
Overall Approach

Select a polygon, $P$

Calculate $e$ for $P$.

Determine the longest “viable” sequence of vertices in $P$

Run repair algorithm clockwise around the polygon
Calculate $e$ for $P$

- Let $d =$ minimum line spacing of the circuit
- Given polygon $P$, find its neighbors
- Compute the minimum distance, $t$, between $P$ and its neighbors
- Let $e = \min (d, t)$
Repair Algorithm: Defining Sequences

\[
\min \left\{ |\Delta x_{in}|, |\Delta y_{in}|, \frac{|\Delta x_{in} - \Delta y_{in}|}{\sqrt{2}}, \frac{|\Delta x_{in} + \Delta y_{in}|}{\sqrt{2}} \right\} < \frac{\varepsilon}{2}
\]
Repair Algorithm: Defining Sequences

Have we exceeded all Bounding Bands?

- No: Check Next Vertex
- Yes: Horizontal, Vertical or Diagonal?
Repair Algorithm: Simplification

Have we exceeded all Bounding Bands?

Yes
Horizontal, Vertical or Diagonal?

No
Check Next Vertex

Simplify with Always-Fatten Rule

H or V
Always-Fatten Rule

- Safely increase the size of the polygons
- Prevent some self-intersections
More General Repair Algorithm: Joining To Unrepaired Segments
Repair Algorithm: U-Turns

- Long, thin segments can be repaired to nothing
- Need a better fidelity measure
Dealing with U-Turns

• Measure Euclidean distance between initial and terminal vertices

• Stop if decrease detected
Have we exceeded all Bounding Bands?

- Yes
  - Horizontal, Vertical or Diagonal?
    - H or V
      - Simplify with Always-Fatten Rule
    - Diag
      - Simplify with Diagonal Fattening Rule

- No
  - Check Next Vertex

Repair Algorithm: Simplification
Repair Algorithm: Simplification

Diagonal Fattening Rule
Repair Algorithm: Recap

1. Select a polygon, $P$
2. Calculate $e$ for $P$.
3. Determine the longest “viable” sequence of vertices in $P$.
4. Run repair algorithm clockwise around the polygon.
Numerical Results: Example 1
Numerical Results: Example 2

Before: 107 vertices; After: 86 vertices; Tolerance = 5
20 Perc Decrease in Vertices (61 Perc on Diag polygon), Time to Process: 0.14 sec
Numerical Results: A Comparison
Numerical Results: Diminishing Returns

<table>
<thead>
<tr>
<th>Tolerance</th>
<th># vertices before</th>
<th># of vertices after</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23838</td>
<td>18435</td>
<td>23%</td>
</tr>
<tr>
<td>2</td>
<td>23838</td>
<td>14486</td>
<td>39%</td>
</tr>
<tr>
<td>3</td>
<td>23838</td>
<td>7721</td>
<td>68%</td>
</tr>
<tr>
<td>4</td>
<td>23838</td>
<td>6435</td>
<td>73%</td>
</tr>
<tr>
<td>5</td>
<td>23838</td>
<td>5476</td>
<td>77%</td>
</tr>
<tr>
<td>6</td>
<td>23838</td>
<td>5122</td>
<td>79%</td>
</tr>
<tr>
<td>7</td>
<td>23838</td>
<td>4872</td>
<td>80%</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

- Our algorithm effectively reduced vertices and maintained fidelity
- Investigate using a separate safety factor for each polygon vertex
- Greedy sequence approach
- Implement an efficient method to find e
- Utilize energy function to measure effectiveness