Incremental Physical Design

Jason Cong  
UCLA

Majid Sarrafzadeh  
Northwestern

Outline

- PART I: Introduction & Motivation
- PART II: Partitioning, Floorplanning, and Placement
- PART III: Routing
- PART IV: Conclusion
PART I:
Introduction & Motivation

Introduction & Motivation

- Concurrent optimization:
  - problems are getting complex
  - need quick evaluation of a number of alternatives
- Existing algorithms are “incremental”
  - what can we say about the quality of the final solution?
  - when to do, and how much, incremental “moves”?
- A more fundamental concept: Given a solution, assess its quality
- Design for change
- The right data structures
Current Methods

- Ad-hoc (low-temp annealing or rip-up and reroute)
- No or very little understanding of the solution quality
- Painful and slow updates
- May have undesired global impact

- There has been very little done on the topic
- (We provide a reasonable list of references as a starting points: data structures, layout algorithms, synthesis algorithms, etc)

- Today’s presentation is meant to serve as a motivator

PART II:
Partitioning, Floorplanning, and Placement
Incremental Partitioning

- An original (hyper)graph and its near-optimal partitioning results are given.
- An incrementally changed (hyper)graph is obtained by adding/deleting vertices/edges from the original graph.
- Incremental partitioning problem is to find the near-optimal partitioning for the changed (hyper)graph without doing from the scratch.
- Incremental partitioning should be much faster than the original partitioning.
- It is preferable to use the existing partitioning results for the original (hyper)graph.
Original Graph

One Vertex Changed (e.g., resized)
One Edge Added (logic restructuring)

3rd Vertex Changed
0-proximity Vertices

1-proximity Vertices
2-proximity Vertices

Partitioning Problem on the Original Graph
Partitioning Problem on the Changed Graph (inf-threshold)

0-Threshold Incremental Partitioning
1-Threshold Incremental Partitioning

Incremental Partitioning with 1% Change In the Original Graph

Net-cut results normalized to infinity-threshold results.
Incremental Partitioning with 1% Change In the Original Graph

Runtime normalized to infinity-threshold results.

Incremental Partitioning with 5% Change In the Original Graph

Net-cut results normalized to infinity-threshold results.
Incremental Partitioning with 5% Change In the Original Graph

Runtime normalized to infinity-threshold results.

Incremental Partitioning with 10% Change In the Original Graph

Net-cut results normalized to infinity-threshold results.
Incremental Partitioning with 10% Change In the Original Graph

![Bar chart showing runtime normalized to infinity-threshold results.](image)

Runtime normalized to infinity-threshold results.

Incremental Partitioning with 20% Change In the Original Graph

![Bar chart showing net-cut results normalized to infinity-threshold results.](image)

Net-cut results normalized to infinity-threshold results.
Incremental Partitioning with 20% Change In the Original Graph

![Graph showing runtime comparison for incremental partitioning with 20% change in the original graph.]

Runtime normalized to infinity-threshold results.

Incremental Partitioning with 1% Change In the Original Graph

<table>
<thead>
<tr>
<th>Ckt</th>
<th>threshold = 0</th>
<th></th>
<th></th>
<th>threshold = 1</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>act. vtx</td>
<td>cut</td>
<td>time</td>
<td>act. vtx</td>
<td>cut</td>
<td>time</td>
</tr>
<tr>
<td>ibm01</td>
<td>127</td>
<td>291</td>
<td>.11s</td>
<td>1384</td>
<td>291</td>
<td>.55s</td>
</tr>
<tr>
<td>ibm02</td>
<td>196</td>
<td>285</td>
<td>.15s</td>
<td>2501</td>
<td>286</td>
<td>1.1s</td>
</tr>
<tr>
<td>ibm03</td>
<td>231</td>
<td>767</td>
<td>.18s</td>
<td>2421</td>
<td>766</td>
<td>1.5s</td>
</tr>
<tr>
<td>ibm04</td>
<td>275</td>
<td>617</td>
<td>.24s</td>
<td>2717</td>
<td>615</td>
<td>1.1s</td>
</tr>
<tr>
<td>ibm05</td>
<td>293</td>
<td>1845</td>
<td>.28s</td>
<td>4269</td>
<td>1841</td>
<td>2.5s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ckt</th>
<th>threshold = 2</th>
<th></th>
<th></th>
<th>threshold = ∞</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>act. vtx</td>
<td>cut</td>
<td>time</td>
<td>act. vtx</td>
<td>cut</td>
<td>time</td>
</tr>
<tr>
<td>ibm01</td>
<td>7846</td>
<td>290</td>
<td>5.2s</td>
<td>12752</td>
<td>315</td>
<td>10.8s</td>
</tr>
<tr>
<td>ibm02</td>
<td>15261</td>
<td>280</td>
<td>16.4s</td>
<td>19601</td>
<td>276</td>
<td>16.8s</td>
</tr>
<tr>
<td>ibm03</td>
<td>15679</td>
<td>819</td>
<td>18.4s</td>
<td>23136</td>
<td>780</td>
<td>24.5s</td>
</tr>
<tr>
<td>ibm04</td>
<td>17374</td>
<td>596</td>
<td>15.3s</td>
<td>27507</td>
<td>545</td>
<td>25.4s</td>
</tr>
<tr>
<td>ibm05</td>
<td>23962</td>
<td>1846</td>
<td>31.6s</td>
<td>29347</td>
<td>1855</td>
<td>39.3s</td>
</tr>
</tbody>
</table>
### Incremental Partitioning with 5% Change In the Original Graph

<table>
<thead>
<tr>
<th>Ckt</th>
<th>threshold = 0</th>
<th></th>
<th>threshold = 1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>act. vtx</td>
<td>cut</td>
<td>time</td>
<td>act. vtx</td>
</tr>
<tr>
<td>ibm01</td>
<td>637</td>
<td>495</td>
<td>.42s</td>
<td>5024</td>
</tr>
<tr>
<td>ibm02</td>
<td>980</td>
<td>293</td>
<td>.57s</td>
<td>8777</td>
</tr>
<tr>
<td>ibm03</td>
<td>1156</td>
<td>884</td>
<td>2.1s</td>
<td>8826</td>
</tr>
<tr>
<td>ibm04</td>
<td>1375</td>
<td>1015</td>
<td>0.96s</td>
<td>10291</td>
</tr>
<tr>
<td>ibm05</td>
<td>1467</td>
<td>2260</td>
<td>1.1s</td>
<td>13706</td>
</tr>
</tbody>
</table>

### Incremental Partitioning with 10% Change In the Original Graph

<table>
<thead>
<tr>
<th>Ckt</th>
<th>threshold = 2</th>
<th></th>
<th>threshold = ∞</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>act. vtx</td>
<td>cut</td>
<td>time</td>
<td>act. vtx</td>
</tr>
<tr>
<td>ibm01</td>
<td>11669</td>
<td>486</td>
<td>9.5s</td>
<td>12752</td>
</tr>
<tr>
<td>ibm02</td>
<td>18902</td>
<td>289</td>
<td>19.8s</td>
<td>19601</td>
</tr>
<tr>
<td>ibm03</td>
<td>21864</td>
<td>849</td>
<td>25.5s</td>
<td>23136</td>
</tr>
<tr>
<td>ibm04</td>
<td>25487</td>
<td>1011</td>
<td>31.4s</td>
<td>27507</td>
</tr>
<tr>
<td>ibm05</td>
<td>28124</td>
<td>2232</td>
<td>43.4s</td>
<td>29347</td>
</tr>
</tbody>
</table>
Incremental Partitioning with 20% Change In the Original Graph

<table>
<thead>
<tr>
<th>Ckt</th>
<th>threshold = 0</th>
<th>threshold = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>act. vtx</td>
<td>cut</td>
</tr>
<tr>
<td>ibm01</td>
<td>2550</td>
<td>1104</td>
</tr>
<tr>
<td>ibm02</td>
<td>3920</td>
<td>369</td>
</tr>
<tr>
<td>ibm03</td>
<td>4627</td>
<td>1115</td>
</tr>
<tr>
<td>ibm04</td>
<td>5501</td>
<td>2320</td>
</tr>
<tr>
<td>ibm05</td>
<td>5869</td>
<td>3432</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ckt</th>
<th>threshold = 2</th>
<th>threshold = ∞</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>act. vtx</td>
<td>cut</td>
</tr>
<tr>
<td>ibm01</td>
<td>12551</td>
<td>701</td>
</tr>
<tr>
<td>ibm02</td>
<td>19472</td>
<td>292</td>
</tr>
<tr>
<td>ibm03</td>
<td>22923</td>
<td>782</td>
</tr>
<tr>
<td>ibm04</td>
<td>27225</td>
<td>1383</td>
</tr>
<tr>
<td>ibm05</td>
<td>29074</td>
<td>3246</td>
</tr>
</tbody>
</table>

Some more ideas

- Localized proximity
- Net-list dependent proximity
- New/ targeted algorithms (not “locked full partitioner”) for incremental partitioning
- Create an initial solution that is suitable for change
Incremental Floorplanning

- Do not resize the floorplan if not needed or if not needed "badly" (based on an enhanced sizing tree) [Crenshaw-Sarrafzadeh]
A slack based sizing tree
(for quick decision making)

Speed-up after a series of moves
(adding buffers, sizing, gate duplication)
(Floorplan) Design for Change

- Accurate data is not available in early phases:
  - Missing cells: Some cells have not been designed yet. Use estimates from previous design.
    - Formulate designer’s experience in estimation of the cell area.
  - Evolving (cell / IP) library: Final area of modules can be estimated from current area.
  - Geometric synthesis, timing improvement, ....
- Nostradamus: A Floorplanner of Uncertain Designs [Bazargan-Kim-Sarrafzadeh]

Problem Formulation

- Distribution lists for width/height:
  - \( W_i = \{(w_{i1}, p(w_{i1})), \ldots, (w_{i,m_i}, p(w_{i,m_i}))\} \quad \sum_j p(w_j) = 1 \)
  - \( H_i = \{(h_{i1}, p(h_{i1})), \ldots, (h_{i,n_i}, p(h_{i,n_i}))\} \quad \sum_j p(h_j) = 1 \)
  - Example:
    - \( W_1 = \{(5,0.3), (7,0.5), (8,0.2)\} \)
    - \( W_2 = \{(2,0.9), (3,0.1)\} \)
    - \( H_1 = \{(1,1), (2,2), (7,7)\} \)
    - \( H_2 = \{(4,4), (6,6)\} \)
  - We could also use:
    - \( L_i = \{(w_{i1}, h_{i1}, p_{i1}), \ldots, (w_{i,m_i}, h_{i,m}, p_{i,m})\} \)
Output Distributions

Input/Output Correlation
Experimental Results

Ratio of actual area of traditional methods to area of Nostradamus: 30% uncertainty

Conservative  Optimistic  Exp. Val.

Experimental Results

Ratio of actual area of traditional methods to area of Nostradamus: 10% uncertainty

Conservative  Optimistic  Exp. Val.
# Placement

## Placement Problem

**Input:**
- A set of cells and their complete information (a cell library).
- Connectivity information between cells (netlist information).

**Output:**
- A set of locations on the chip: one location for each cell.

**Goal:**
- Total chip area and wirelength, under timing constraints.

**Challenge:**
- Rapid growing circuit size (> 1 million).
- Multi objectives (timing constraint, routability)
Placement Problem

Input: Cells and Interconnections
Output: Locations on a 2-D plane

Predictors based on the given algorithm or based on absolute truth

A good placement

A bad placement
Placement Algorithms (both incremental, so do we understand incremental?)

- Constructive Placement
  *e.g. Analytical optimization*
  Build up placement from scratch.
  Fast, but most of them lack of solution quality.

- Iterative Improvement
  *e.g. Simulated annealing*
  Gradually improve existing solutions.
  Slow, but with good solution quality, easy to consider multi-objectives.

Incremental Placement

Caused by:
- Adding cells/nets
- Meeting timing constraints
  *while minimizing total chip area.*
- Reducing congestion
  *A post processing algorithm is more effective*
Proximity In Placement

Topological Proximity

Cell A and B are topological/proximate

Geometrical Proximity

Cell A and B are geometrical/proximate

In a good placement, topological proximity and geometrical proximity generally correlate well

Preliminary studies

Reducing length of long net

Helps meeting timing constraints
Shrinking long nets

Pick up cells which are located on the edge of the Bounding box, switch them with other cells.

Experimental setup

Given a placement produced by a fresh placement run, find the longest nets, Let its length be L, then find all the nets which are longer than (1-x%)*L, reduce all these long nets to (1-x%)*L or less.

Data reported are number of long nets and x%
Experiment results

<table>
<thead>
<tr>
<th>Circuit</th>
<th>#cells</th>
<th>Length of long nets are reduced to (1-x)% of longest net</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>5%</td>
</tr>
<tr>
<td></td>
<td># nets</td>
<td>WL</td>
</tr>
<tr>
<td>Primary2</td>
<td>2907</td>
<td>1</td>
</tr>
<tr>
<td>industry2</td>
<td>12142</td>
<td>2</td>
</tr>
<tr>
<td>industry3</td>
<td>15059</td>
<td>1</td>
</tr>
<tr>
<td>avqs</td>
<td>21854</td>
<td>1</td>
</tr>
<tr>
<td>avql</td>
<td>25114</td>
<td>1</td>
</tr>
</tbody>
</table>

It is not difficult to reduce the longest net by up to 20% without considerably increasing the total wirelength.

Experiment results

It’s a greedy, simple approach. More effective algorithms are to be studied.
PART III: Incremental Routing

Incremental Routing Problem

- Basic operations:
  - Net removal (easy)
  - Net addition (focus of this talk)
- Objective
  - Preserve as much previous results as possible
- Applications
  - Changes of the netlist due to functionality modification
  - Modification of placement due to performance consideration
  - Change of a route (width, spacing, possible buffer insertion) after post-layout extraction and simulation
  - ...

ISPD 2000: April 10-12

Jason Cong
Majid Sarrafzadeh
Sub-problems in Incremental Routing

- Single Net Routing
  - No change of existing routes
- Rip-up and Reroute
  - Change some existing routes but no change of placement/floorplan
- Incremental Floorplan and Placement Update
  - Invoke when Rip-up & re-route fails

Increasing flexibility
Increasing runtime

Single Net Routing Problem

- Problem Formulation
  Given
  - a set of existing routes/obstacles
  - the width and spacing of a given net
  Find a valid route for this net
- Need for gridless (GLS) routing
  - Variable wire width for delay optimization
  - Variable wire spacing for noise constraints
- Challenges
  - Construction of GLS routing graph
  - Representation of GLS routing graph
  - Search on GLS routing graph
Existing Approaches to GLS SNR

- Obstacle expansion
  - Reduced to zero-width routing
- Construction of routing graph
  - Tile-based approach
  - Point-based approach
    - Uniform grid graph
    - Non-uniform grid graph
- Search on the routing graph

Obstacle Expansion for Gridless Routing

- Expansion of Obstacles According to Width Rule and Spacing Rule to Reduce the Routing Problem into Zero Width Routing

\[
W_w = 4, \quad S_w = 2, \quad E_w = W_w / 2 + S_w
\]
Tile-based Approach
[Margarino CAD’87; Sechen ISPD’98]

1. Tile Partition
2. Corner-Stitching
   Data Structure
   [Ousterhout, TCAD’84]
3. A Tile-to-Tile
   Searching

Uniform Grid Approach [Lee’61]

Representing the region
using uniform grids
Graph can be very large!
Minimal Routing Graph Approach
[Wu T-Comp’87; Zheng TCAD’96]

Extend the boundaries of expanded rectangles
Sparser than uniform-grid but irregular

Limitations of Existing Approaches for Incremental Routing

- Pre-expansion of obstacles - very costly
  - Need to re-expand if routing a net of different width and/or spacing
- Pre-construction of routing graph - very costly
  - Incremental route may use only a small portion of the graph (but cannot be determined a priori)
Non-Uniform Grid Graph Approach
[Cong-Fang-Khoo, ICCAD’99]

- Given a set of obstacles and a source $s$ and sink $t$
- $G_S$ is an orthogonal grid graph

Implicit Representation of $G_S$

- Store $x$, $y$ locations in arrays
- Space Pre-construction
- Can be maintained *incrementally* ($O(\log N)$ time)
Searching on $G_S$

- Determine the Location of Neighboring Node: Easy
- Determine If the Location Is Valid: Difficult

No Pre-Expansion of Obstacles

- Point enclosure query $\rightarrow$ rectangle intersection query
Our 2-D Query Data Structure

Experimental Results

- Examples
  
<table>
<thead>
<tr>
<th>Ex.</th>
<th>Dimension</th>
<th>Layers</th>
<th>Pins</th>
<th>Cells</th>
<th>Rects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eco-1</td>
<td>1372.5 x 1593.3</td>
<td>3</td>
<td>3</td>
<td>6417</td>
<td>232309</td>
</tr>
<tr>
<td>Eco-2</td>
<td>1372.5 x 1593.3</td>
<td>3</td>
<td>2</td>
<td>6417</td>
<td>232453</td>
</tr>
<tr>
<td>Eco-3</td>
<td>1372.5 x 1593.3</td>
<td>3</td>
<td>5</td>
<td>6417</td>
<td>232004</td>
</tr>
<tr>
<td>Eco-4</td>
<td>1372.5 x 1608.6</td>
<td>3</td>
<td>2</td>
<td>6417</td>
<td>232633</td>
</tr>
<tr>
<td>Eco-5</td>
<td>1556.6 x 1676.8</td>
<td>3</td>
<td>2</td>
<td>7542</td>
<td>196947</td>
</tr>
<tr>
<td>Eco-6</td>
<td>2926.1 x 1676.8</td>
<td>3</td>
<td>2</td>
<td>13959</td>
<td>372240</td>
</tr>
<tr>
<td>Eco-7</td>
<td>2741.5 x 3192.8</td>
<td>3</td>
<td>2</td>
<td>25668</td>
<td>701172</td>
</tr>
</tbody>
</table>

- Comparison
  - Explicit Uniform Grid Graph
  - Iroute from Magic Layout Editor [Arnold 88]
Comparison of Memory Usage
Unit: MB

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Eco-1</td>
<td>160.2</td>
<td>10.9</td>
<td>32.7</td>
</tr>
<tr>
<td>Eco-2</td>
<td>160.2</td>
<td>10.9</td>
<td>32.7</td>
</tr>
<tr>
<td>Eco-3</td>
<td>160.2</td>
<td>7.2</td>
<td>32.6</td>
</tr>
<tr>
<td>Eco-4</td>
<td>161.7</td>
<td>10.9</td>
<td>32.6</td>
</tr>
<tr>
<td>Eco-5</td>
<td>191.0</td>
<td>12.7</td>
<td>35.2</td>
</tr>
<tr>
<td>Eco-6</td>
<td>359.4</td>
<td>15.9</td>
<td>52.6</td>
</tr>
<tr>
<td>Eco-7</td>
<td>641.1</td>
<td>43.6</td>
<td>84.7</td>
</tr>
<tr>
<td>Ratio</td>
<td>14.3</td>
<td>1.00</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Comparison of Runtime and ECO Routing Quality

<table>
<thead>
<tr>
<th>Ex.</th>
<th>Eco-1</th>
<th>Eco-2</th>
<th>Eco-3</th>
<th>Eco-4</th>
<th>Eco-5</th>
<th>Eco-6</th>
<th>Eco-7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time(s)</td>
<td>19.1</td>
<td>6.3</td>
<td>34.5</td>
<td>24.0</td>
<td>12.3</td>
<td>24.7</td>
<td>38.2</td>
</tr>
<tr>
<td>Non-Uniform Result (w/l/vc)</td>
<td>17374 / 66</td>
<td>11332 / 44</td>
<td>34736 / 110</td>
<td>21760 / 122</td>
<td>27061 / 54</td>
<td>37858 / 70</td>
<td>35690 / 74</td>
</tr>
<tr>
<td>Time(s)</td>
<td>42.15</td>
<td>26.58</td>
<td>68.70</td>
<td>57.39</td>
<td>43.14</td>
<td>74.29</td>
<td>79.79</td>
</tr>
<tr>
<td>Iroute Result (w/l/vc)</td>
<td>22629 / 89</td>
<td>13162 / 62</td>
<td>36593 / 103</td>
<td>24385 / 153</td>
<td>34543 / 12</td>
<td>56423 / 20</td>
<td>47591 / 20</td>
</tr>
</tbody>
</table>
Sub-problems in Incremental Routing

- Single Net Routing
  - No change of existing routes
- Rip-up and Reroute
  - Change some existing routes but no change of placement/floorplan
- Incremental Floorplan and Placement Update
  - Invoke when Rip-up & re-route fails

Rip-up and Reroute

- Invoke when net-by-net routing fails
- Objectives
  - Complete all the nets
  - Minimize number of rip-ups
- Challenges
  - Gridless nature of routing problem makes routing resource difficult to model and manage
Overview of Existing R&R Approaches

- Maintain Design-Rule Correctness
  - Pros: always has a valid partial solution
  - Cons: too restrictive; net order dependent

- Allow Temporary Design-Rule Violation
  - Pros: allow cross and touch [Kawamura ICCAD’90], more flexibility
  - Cons:
    - Difficult to model routing resources in gridless routing
    - Lack of global picture

- Planning Based Ripup and Re-route
  - Preferred approach
    - Can model routing resources with flexible resolution
    - Allow a more global view
    - Can be applied iteratively

Planning Graph Construction
Planning Based Rip-up & Re-route

- Advantages
  - More Global Picture
  - Reasonably Accurate Routing Region Information
  - Very efficient on the planning graph
  - Similar to global routing, but only serve as a guide
- Re-planning Approaches
  - Local Refinement
  - Ripup and Re-plan

Local Refinement
Local Refinement
Ripup & Re-planning

Sub-problems in Incremental Routing

- Single Net Routing
  - No change of existing routes

- Rip-up and Reroute
  - Change some existing routes but no change of placement/floorplan

- Incremental Floorplan and Placement Update
  - Invoke when Rip-up & re-route fails

Increasing flexibility
Increasing runtime
Incremental Floorplan and Placement Update

- Problem Formulation
  - When rip-up and re-route fails, determine how to complete the design with minimum changes of the floorplan and/or placement
- Need efficient heuristics to decide what routing regions to enlarge
- May apply compaction/de-compaction techniques
- Earlier discussions on incremental floorplan and placement also apply

Conclusions

- Design iteration is unavoidable
- Incremental PD is an important enabling technology to allow design convergence
- Trade-off of efficiency, preservability, and design quality is the key issue in incremental designs
- Much work is needed in this area
- Need to start thinking about “Design for Incremental Changes” (DFI)
Acknowledgements

- NSF, DARPA, and Fujitsu Laboratories
- Jie Fang and Kei-Yong Khoo from UCLA
- Kiarash Barzagan, Maogang Wang, and Xiaoqian Yang
  from Northwestern Univ.