Scalable Hierarchical Floorplanning for Fast Physical Prototyping of Systems-on-Chip

Renshen Wang and Nimish Shah

Mentor Graphics Corporation

March 28th, 2012

Napa, CA
Scaling of VLSI Circuits and Design Cycles

- Moore’s law continues forward
- Design size growing
- Approaching, exceeding 500 million gates on a chip

- Challenges for existing EDA software

- Space
  - Large virtual memory with 64-bit address
  - But typical machines have 128GB of RAM
  - “Memory wall”, memory speed trailing behind

- Time
  - CPU speed up < Problem size increase
  - Effort on parallel computing, but many algorithms hard to parallelize
Hierarchical Design Flow

- **Divide-and-conquer**
  - Divide the large (typically modular) design into regions
  - Keep each region size in a sweet spot of the implementation tool
  - Allocate chip area for each region, i.e., floorplanning
  - Implementation of the regions

- **Floorplanning in the hierarchical flow**

![Diagram showing the process of hierarchical floorplanning](image)
The Floorplanning Problem

- Classical floorplanning harmful? [Kahng2000]
  - Preoccupation with packing driven (macro placement?)
  - Lack of attention to scalability
  - Lack of attention to the overall RTL-down methodology context
  - Etc.

- Our hierarchical floorplan
  - Fixed die area
  - Flexible shape (under constraints)
  - Seed placement for connectivity quality
  - Slicing tree for scalability

(a) Floorplan with good shapes
(b) Examples of bad shapes
Formulation for Scalable Floorplanning

- **Input:** A seed placement on the target design
  - Rectangle die area $D$
  - Upper bound on region aspect ratio $A_u$
  - $n$ regions $r_1 \ldots r_n$
  - Each region $r_i$ has $m_i$ cells, each cell $e_{i,j}$ placed at point $p(e_{i,j}) \in D$

- **Solution:** A set of rectangles $R_1 \ldots R_n \subseteq D$ such that
  \[
  \text{area}(R_i) \geq \sum_j \text{area}(e_{i,j}), \quad \text{(adequate area)}
  \]
  \[
  R_i \cap R_j = \emptyset \text{ for } i \neq j, \quad \text{(non-overlapping)}
  \]
  \[
  \max\left(\frac{l_1(R_i)}{l_2(R_i)}, \frac{l_2(R_i)}{l_1(R_i)}\right) \leq A_u, \quad \text{(good shape)}
  \]

- **Objective:** Rectangles that match up best to the seed placement
  \[
  \text{Maximize } \sum_i \left\{ \sum_j \text{area}(e_{i,j}) : p(e_{i,j}) \in R_i \right\}
  \]
Slicing and Partitioning

- Grid map $(a)$ for cell distributions, and cumulative map $(c)$
  \[ c_{ij}(r_k) = \sum_{u=1}^{i} \sum_{v=1}^{j} a_{uv}(r_k) \]

- Recursive slice-and-partition
  - Find a cut line with minimum cost, assign regions accordingly
  - Recursive calls on both sides of the cut line

Slicing a floorplan by a vertical cut line, on the left side: $r_1$, on the right side: $r_2, r_3$

Cost of this cut line
$= area_{right}(r_1) + area_{left}(r_3)$
Backtracking and Dynamic Programming

- Packing may fail under aspect ratio constraints
- How to backtrack without explosive computation?
- Cache solutions of subsets \( (S_r, [x_{\text{low}}, x_{\text{high}}], [y_{\text{low}}, y_{\text{high}}]) \)
  — Intermediate states of dynamic programming
- Take \( c_g \) (center of gravity) of each region’s seed cells
- Subset \( S_r \) can be decided by picking 4 regions’ \( c_g \) as left, bottom, right, top and bottom boundary (Theorem 1)
- \( O(n^4) \) states
Scalability

- Time and Space upper bound (in theory)
  - $m \times m$ grid, $n$ regions: $O(m^2n^3)$ time, $O(m^2n^2)$ space
  - Take $m \sim n$: $O(n^5)$ time, $O(n^4)$ space

- **Real runtime much lower**, depending on constraints

- Reduction from super-exponential to polynomial
  - Seed placement
  - Preserved “order” among regions

- Quality?
  - Connectivity optimized seed placement
  - Seeking a floorplan that matches the seed

- Seed generation is also scalable
  - Global placement
  - Netlist clustering for coarse seed placement
Experiments

- Runs fast when aspect ratio upper bound > 1.2
  - High design utilization may impose stronger constraints
  - Real world designs usually allow $A_u$ to be 1.2 or higher

- Robust with any distribution of seed placement

<table>
<thead>
<tr>
<th>$A_u$</th>
<th>Design$_1$</th>
<th>Design$<em>2(U</em>{low})$</th>
<th>Design$<em>2(U</em>{high})$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T(s)$</td>
<td>#Int.Sol</td>
<td>$T(s)$</td>
</tr>
<tr>
<td>1</td>
<td>125</td>
<td>1905666</td>
<td>4.5</td>
</tr>
<tr>
<td>1.01</td>
<td>99</td>
<td>1444349</td>
<td>4.0</td>
</tr>
<tr>
<td>1.05</td>
<td>3.2</td>
<td>22565</td>
<td>2.5</td>
</tr>
<tr>
<td>1.1</td>
<td>3.2</td>
<td>29244</td>
<td>1.6</td>
</tr>
<tr>
<td>1.21</td>
<td>2.6</td>
<td>1313</td>
<td>1.5</td>
</tr>
<tr>
<td>1.33</td>
<td>2.6</td>
<td>412</td>
<td>1.5</td>
</tr>
<tr>
<td>1.46</td>
<td>2.6</td>
<td>936</td>
<td>1.5</td>
</tr>
<tr>
<td>1.61</td>
<td>2.6</td>
<td>549</td>
<td>1.5</td>
</tr>
<tr>
<td>1.77</td>
<td>2.6</td>
<td>38</td>
<td>1.5</td>
</tr>
<tr>
<td>1.95</td>
<td>2.6</td>
<td>44</td>
<td>1.5</td>
</tr>
</tbody>
</table>
Measurements

- Quality of a floorplan?
  - $WL_{\text{flat}}$: wire length of a flat placement
  - $WL_{\text{floorplan}}$: wire length of placement with cells placed in regions
  - Seed placement on original netlist and clustered netlist

- $WL_{\text{flat}} < WL_{\text{floorplan}}$ as expected
  - difference is usually small enough
  - effect of clustering to be further studied

<table>
<thead>
<tr>
<th>Test case</th>
<th>$WL_{\text{flat}}$</th>
<th>Placement seed</th>
<th>$WL_{\text{floorplan}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design1</td>
<td>$3.04 \times 10^{11}$</td>
<td>flat</td>
<td>$3.91 \times 10^{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 1</td>
<td>$3.58 \times 10^{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 2</td>
<td>$3.49 \times 10^{11}$</td>
</tr>
<tr>
<td>Design2</td>
<td>$1.39 \times 10^{11}$</td>
<td>flat</td>
<td>$1.52 \times 10^{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 1</td>
<td>$1.52 \times 10^{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 2</td>
<td>$1.44 \times 10^{11}$</td>
</tr>
<tr>
<td>Design3</td>
<td>$7.58 \times 10^{11}$</td>
<td>flat</td>
<td>$7.72 \times 10^{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 1</td>
<td>$8.59 \times 10^{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 2</td>
<td>$8.60 \times 10^{11}$</td>
</tr>
<tr>
<td>Design4</td>
<td>$1.55 \times 10^{12}$</td>
<td>flat</td>
<td>$1.66 \times 10^{12}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 1</td>
<td>$1.74 \times 10^{12}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 2</td>
<td>$1.74 \times 10^{12}$</td>
</tr>
<tr>
<td>Design5</td>
<td>$5.15 \times 10^{11}$</td>
<td>flat</td>
<td>$5.21 \times 10^{11}$</td>
</tr>
<tr>
<td>Design6</td>
<td>$2.05 \times 10^{11}$</td>
<td>flat</td>
<td>$2.23 \times 10^{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 1</td>
<td>$2.32 \times 10^{11}$</td>
</tr>
<tr>
<td>Design7</td>
<td>$1.42 \times 10^{11}$</td>
<td>flat</td>
<td>$1.44 \times 10^{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 1</td>
<td>$1.50 \times 10^{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 2</td>
<td>$1.55 \times 10^{11}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>cluster level 3</td>
<td>$1.63 \times 10^{11}$</td>
</tr>
<tr>
<td>Design8</td>
<td>$2.65 \times 10^{12}$</td>
<td>cluster level 1</td>
<td>$3.03 \times 10^{12}$</td>
</tr>
</tbody>
</table>
Conclusions and Future Work

- A preliminary validation of hierarchical floorplanning flow
- Place interface pins on regions
  - smaller, independent sub-designs
    - Implemented in parallel
    - Faster turn around time

Future enhancement
- fast and high quality seed placement (clustering, placement)
- automatically deciding logic hierarchies of regions
- extending floorplan’s slicing-tree structure
- channel size/shape optimization
- etc.