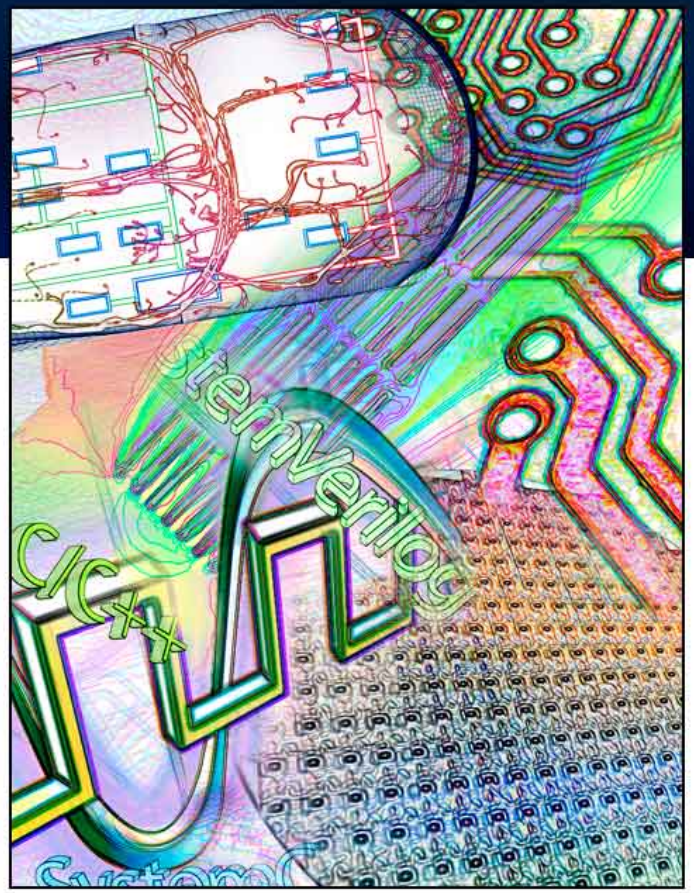
Scalable Hierarchical Floorplanning for Fast Physical Prototyping of Systems-on-Chip



Renshen Wang and Nimish Shah

Mentor Graphics Corporation

March 28th, 2012

Napa, CA

Mentor Graphics®

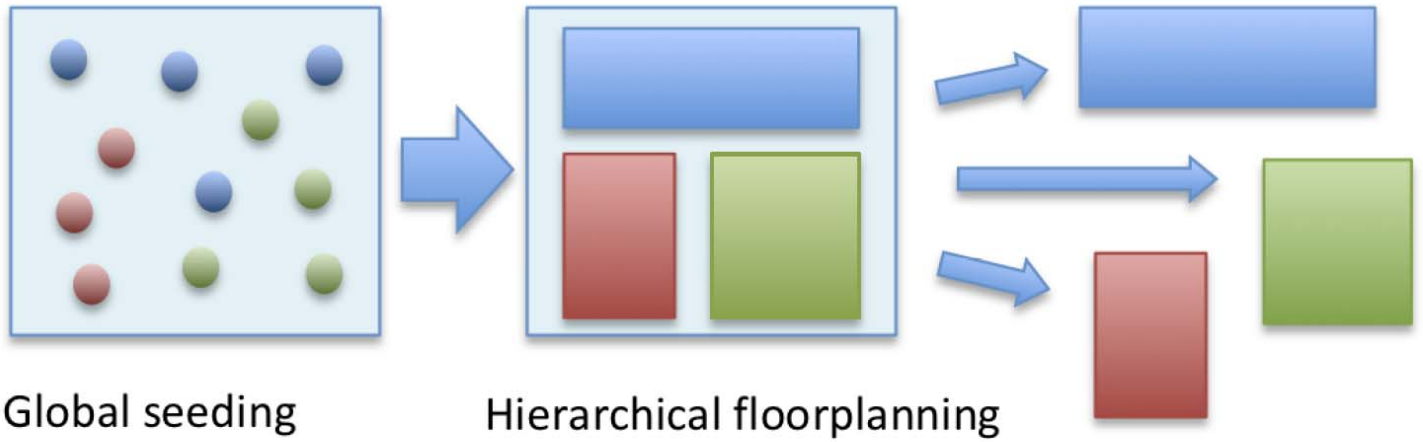
Scaling of VLSI Circuits and Design Cycles

- 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



Global seeding
(partitioning,
clustering,
seed placement)

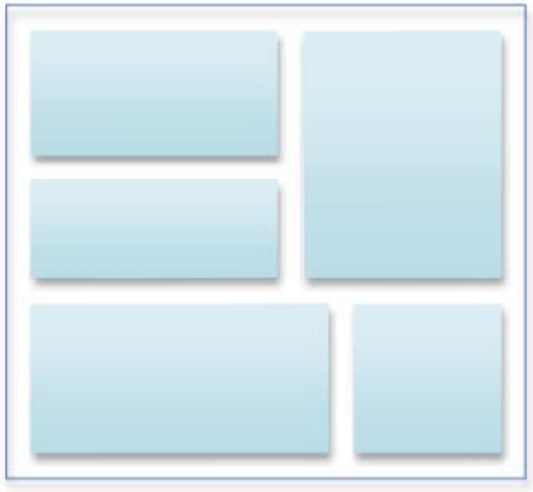
Hierarchical floorplanning
(region shaping,
channel sizing,
pin placement, etc)

Independent
block level
implementation

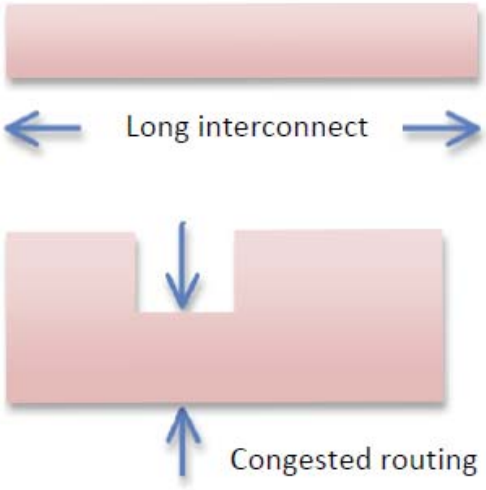
The Floorplanning Problem

- 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

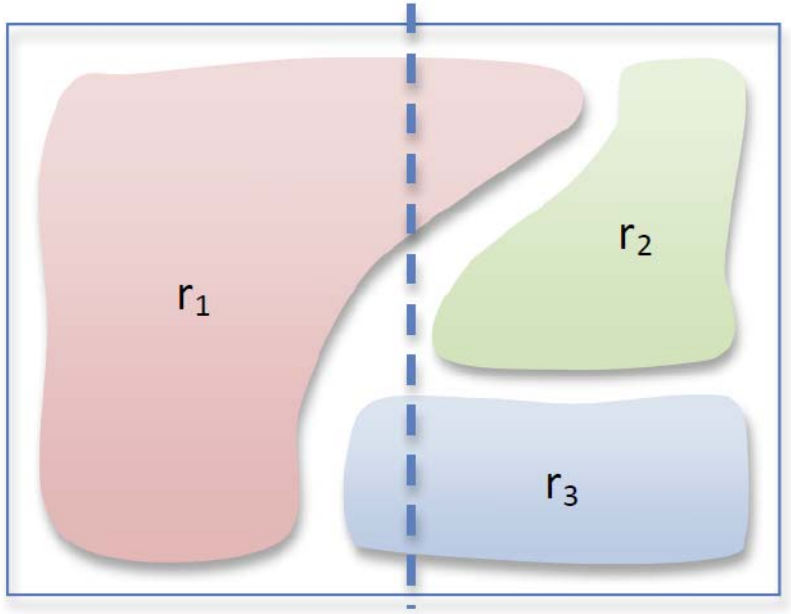
- Input: A seed placement on the target design
  - Rectangle die area D
  - Upper bound on region aspect ratio A_u
  - n regions r_1 ... r_n
  - Each region r_i has m_i cells, each cell $e_{i,j}$ placed at point $p(e_{i,j}) \in D$ (seed placement)
- Solution: A set of rectangles R_1 ... $R_n \subseteq D$ such that

$$area(R_i) \geq \sum_j area(e_{i,j}), \qquad \text{(adequate area)}$$
$$R_i \bigcap R_j = \phi \text{ for } i \neq j, \qquad \text{(non-overlapping)}$$
$$max(\frac{l_1(R_i)}{l_2(R_i)}, \frac{l_2(R_i)}{l_1(R_i)}) \leq A_u \qquad \text{(good shape)}$$

- Objective: Rectangles that match up best to the seed placement

$$\text{Maximize} \sum_i \{\sum_j area(e_{i,j}) : p(e_{i,j}) \in R_i\}$$

Slicing and Partitioning



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
= $\text{area}_{\text{right}}(r_1) + \text{area}_{\text{left}}(r_3)$

- Grid map (a) for cell distributions, and cumulative map (c)

$$c_{ij}(r_k) = \sum_{u=1}^{i} \sum_{v=1}^{j} a_{uv}(r_k)$$

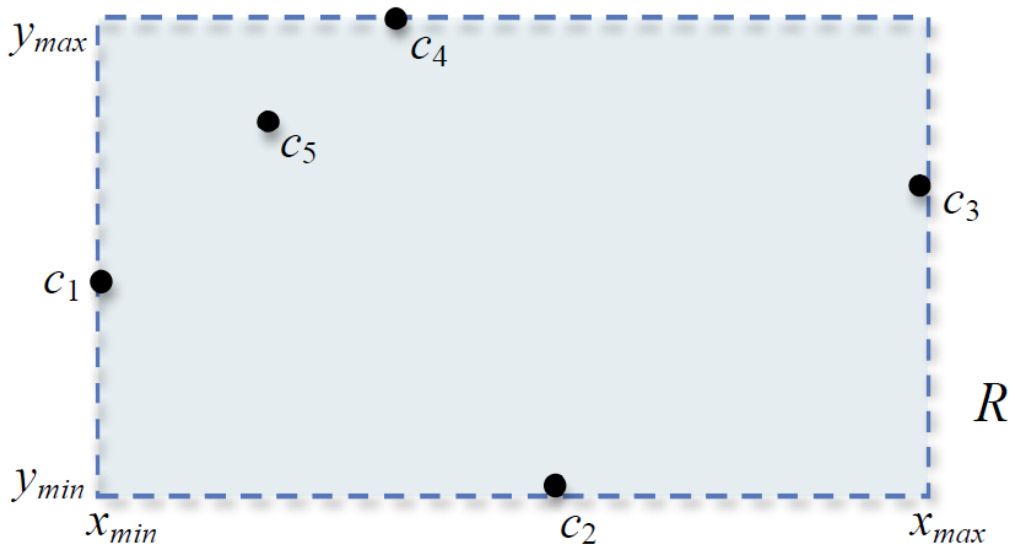
- Recursive slice-and-partition
  - — Find a cut line with minimum cost, assign regions accordingly
  - — Recursive calls on both sides of the cut line

Backtracking and Dynamic Programming

- Packing may fail under aspect ratio constraints
- How to backtrack without explosive computation?
- Cache solutions of subsets (S_r, $[x_{low}, x_{high}]$, $[y_{low}, y_{high}]$)
  - 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 x m grid, n regions: $O(m^2n^3)$ time, $O(m^2n^2)$ space
  - Take m ~ 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

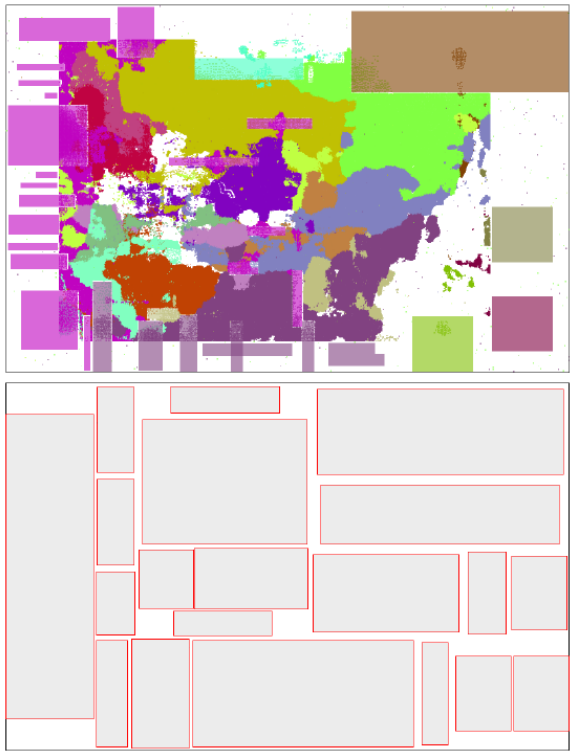


Table 3: Runtimes and cache sizes

A_u	Design_1		$\text{Design}_2(U_{low})$		$\text{Design}_2(U_{high})$	
	T(s)	#Int.Sol	T(s)	#Int.Sol	T(s)	#Int.Sol
1	125	1905666	4.5	64763	1753	9386249
1.01	99	1444349	4.0	60189	2055	9536067
1.05	3.2	22565	2.5	22288	2501	12985300
1.1	3.2	29244	1.6	8309	282	2554501
1.21	2.6	1313	1.5	326	57.6	756856
1.33	2.6	412	1.5	172	3.6	40680
1.46	2.6	936	1.5	145	1.6	9508
1.61	2.6	549	1.5	123	1.5	3551
1.77	2.6	38	1.5	109	1.5	1523
1.95	2.6	44	1.5	95	1.5	806

Measurements

- Quality of a floorplan?
  - WL_{flat}: wire length of a flat placement
  - $WL_{floorplan}$: wire length of placement with cells placed in regions
  - Seed placement on original netlist and clustered netlist
- $WL_{flat} < WL_{floorplan}$ as expected
  - difference is usually small enough
  - effect of clustering to be further studied

Table 4: Wire length results (unit: Å)

Test case	WL_{flat}	Placement seed	$WL_{floorplan}$
$Design_1$	3.04×10^{11}	flat	3.91×10^{11}
		cluster level 1	3.58×10^{11}
		cluster level 2	3.49×10^{11}
$Design_2$	1.39×10^{11}	flat	1.52×10^{11}
		cluster level 1	1.52×10^{11}
		cluster level 2	1.44×10^{11}
$Design_3$	7.58×10^{11}	flat	7.72×10^{11}
		cluster level 1	8.59×10^{11}
		cluster level 2	8.60×10^{11}
$Design_4$	1.55×10^{12}	flat	1.66×10^{12}
		cluster level 1	1.74×10^{12}
		cluster level 2	1.74×10^{12}
$Design_5$	5.15×10^{11}	flat	5.21×10^{11}
$Design_6$	2.05×10^{11}	flat	2.23×10^{11}
		cluster level 1	2.32×10^{11}
$Design_7$	1.42×10^{11}	flat	1.44×10^{11}
		cluster level 1	1.50×10^{11}
		cluster level 2	1.55×10^{11}
		cluster level 3	1.63×10^{11}
$Design_8$	2.65×10^{12}	cluster level 1	3.03×10^{12}

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.

www.mentor.com