Simultaneous Block and I/O Buffer Floorplanning for Flip-Chip Design

Chih-Yang Peng, Wen-Chang Chao, and Yao-Wen Chang

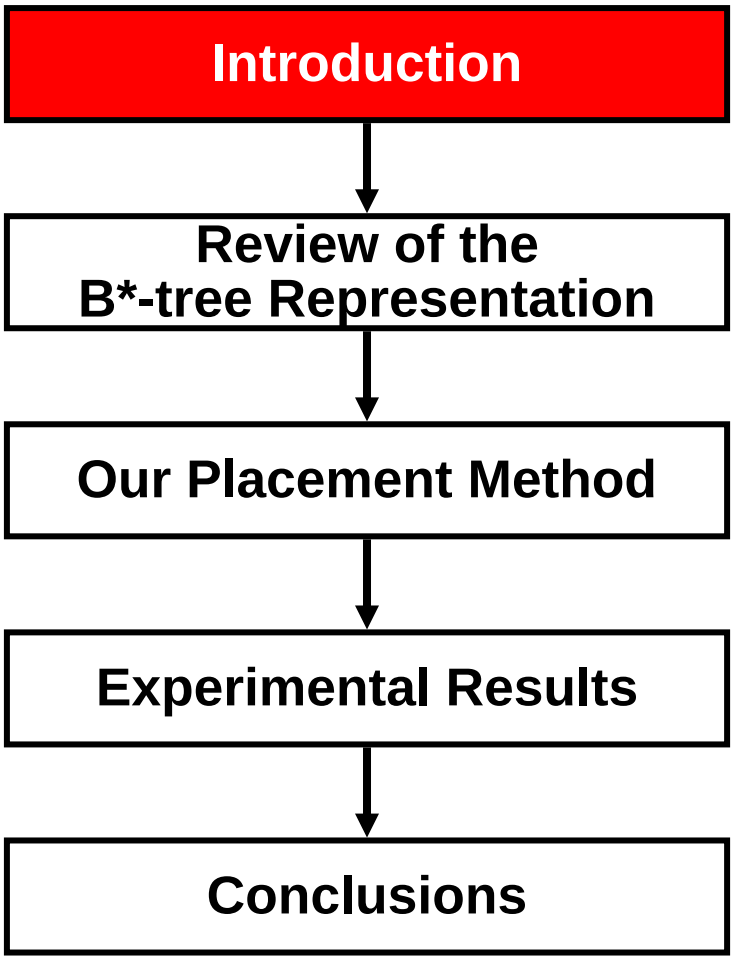
Department of Electrical Engineering,

National Taiwan University, Taipei, Taiwan

Jyh-Herng Wang

Faraday Technology Corporation, Hsinchu, Taiwan

Outline

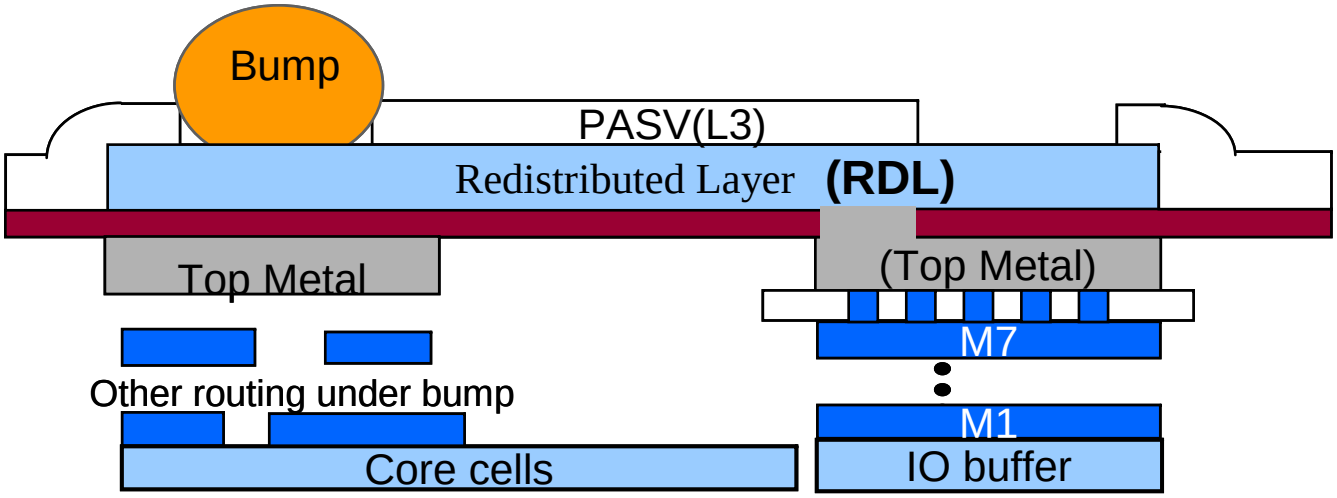

Introduction
Review of the
B*-tree Representation
Our Placement Method
Experimental Results
Conclusions

Introduction

- Develop a hierarchical method for placing the blocks and I/O buffers of flip-chip designs in order to minimize the path delay and skew simultaneously.
- Integrate simulated annealing, partitioning, and clustering based on the B*-tree representation.
- Reduce the average cost by 48% in less runtime compared with the flat B*-tree based floorplanner.

Flip Chip Structure

- The flip-chip package gives the highest chip density of any packaging method to support the pad-limited ASIC design.
- The top metal layer of flip-chip, called Re-Distributed Layer (RDL), connects I/O buffers to bump balls.
- Bump balls are placed on RDL and use RDL to connect to I/O buffers.

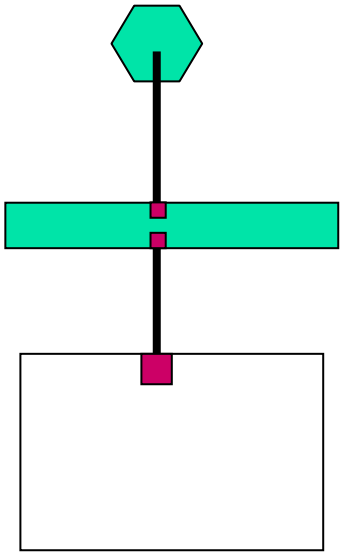


Cross section of the chip

Flip Chip Structure (cont'd)

- The bump balls can overlap with I/O buffers and blocks
- The I/O buffers can be placed anywhere inside a chip unlike the traditional peripheral I/O buffers.
- The signals or power could be imported from the signal bumps or power bumps distributed on the whole chip.

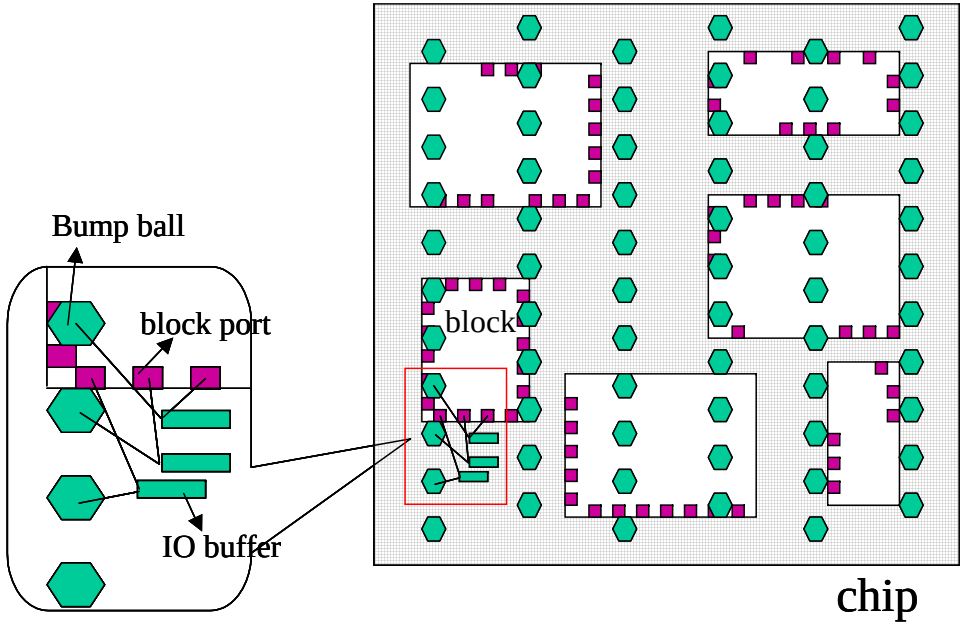
path



Bump ball

I/O buffer

Block



Previous Work

- Hsieh and Wang [ISCAS 2005]
  - Use an analytical method
  - Use sum of skew in the objective function
    - The individual skews may differ a lot.

Problem Formulation

- Given bump ball positions, blocks, IO buffers, and netlist
  - All blocks and buffers are rectangular
- Minimize the objective function: $\Gamma = \alpha\phi_1 + \beta\phi_2$

where

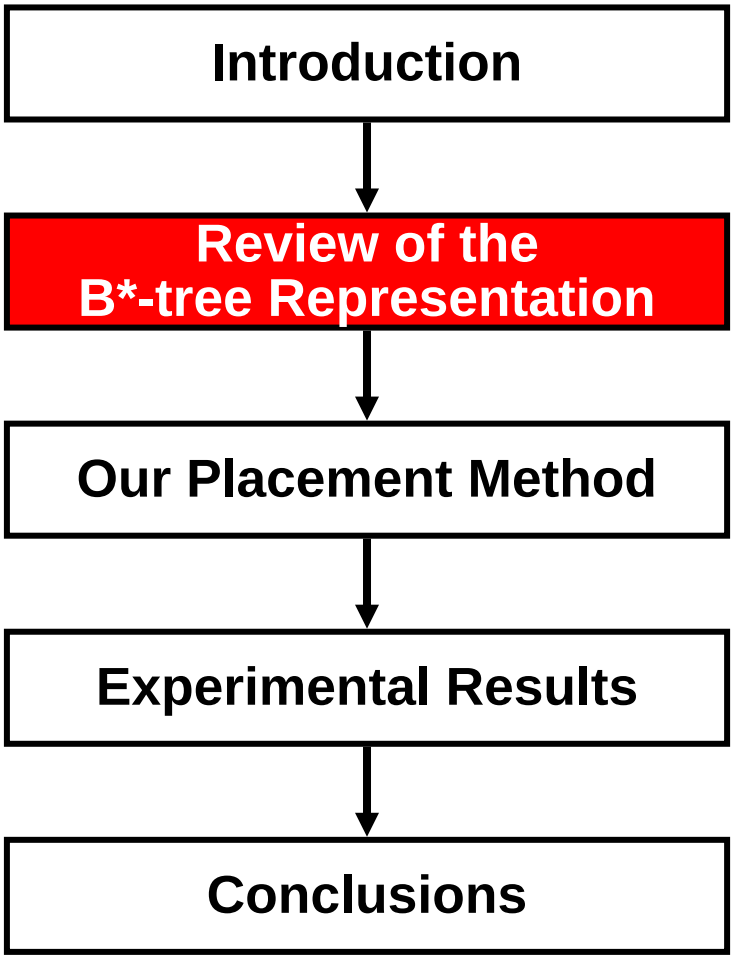
$$\phi_1 = \sum_{j=1}^{n1} d_j^i + \sum_{j=1}^{n2} d_j^o$$ **(sum of path delays)**

$$\phi_2 = \left(\max_{1 \le j \le n1} d_j^i - \min_{1 \le j \le n1} d_j^i \right)^2 + \left(\max_{1 \le j \le n2} d_j^o - \min_{1 \le j \le n2} d_j^o \right)^2$$

(sum of the squares of input/output skews)

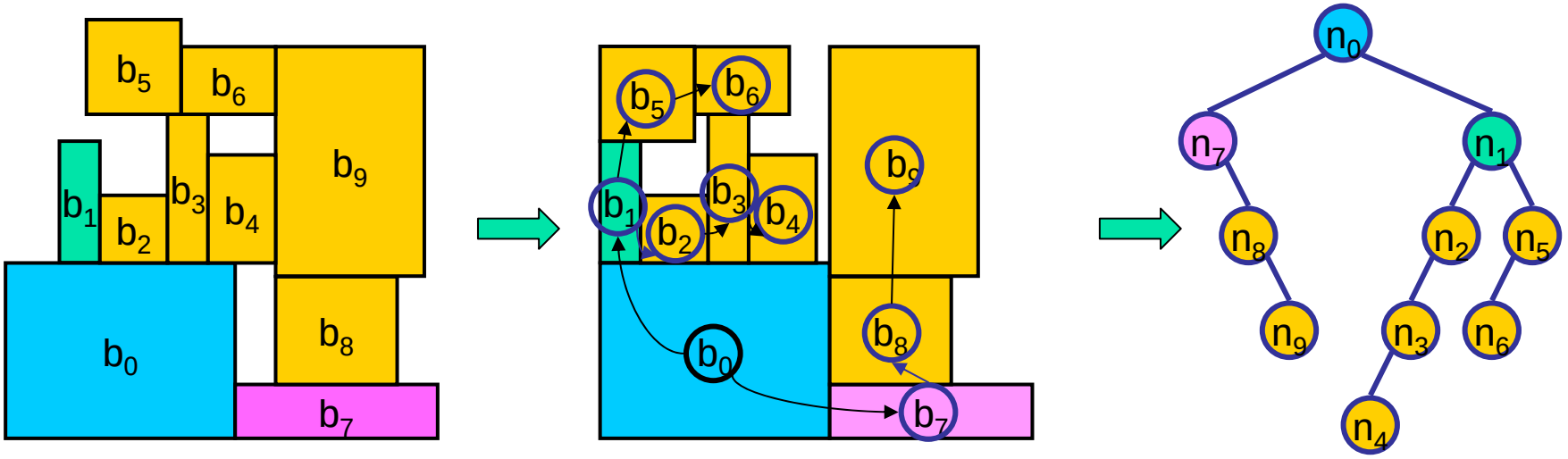
- α and β are user-specified weighting factors.
- d_j^i and d_j^o are the path delays of the jth input signal and the jth output signal respectively.

Outline


Introduction
Review of the
B*-tree Representation
Our Placement Method
Experimental Results
Conclusions

Review of the B*-Tree Floorplan Representation

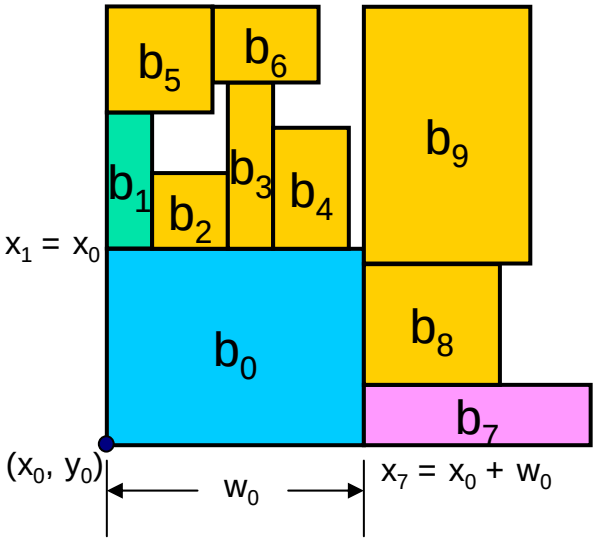
- Chang et al., “B*-tree: A new representation for non-slicing floorplans,” DAC-2k.
  - Compact modules to left and bottom.
  - Construct an ordered binary tree (B*-tree).

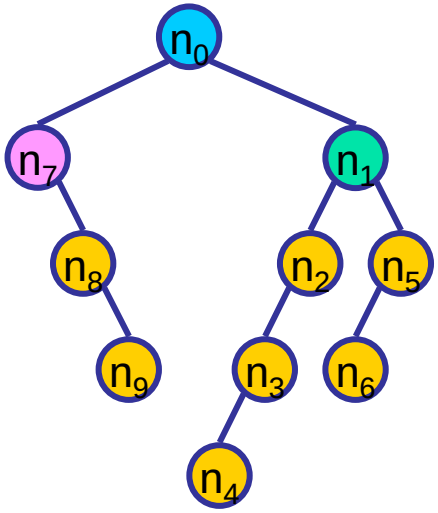


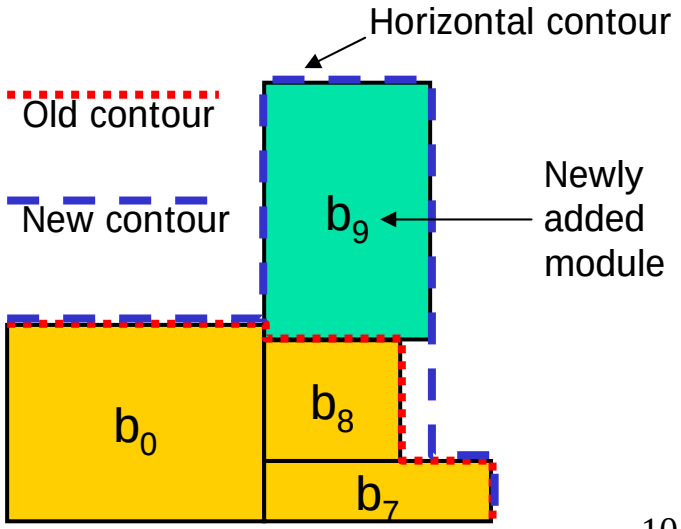
A slicing/non-slicing floorplan Compact to left and down B*-tree

B*-Tree Packing

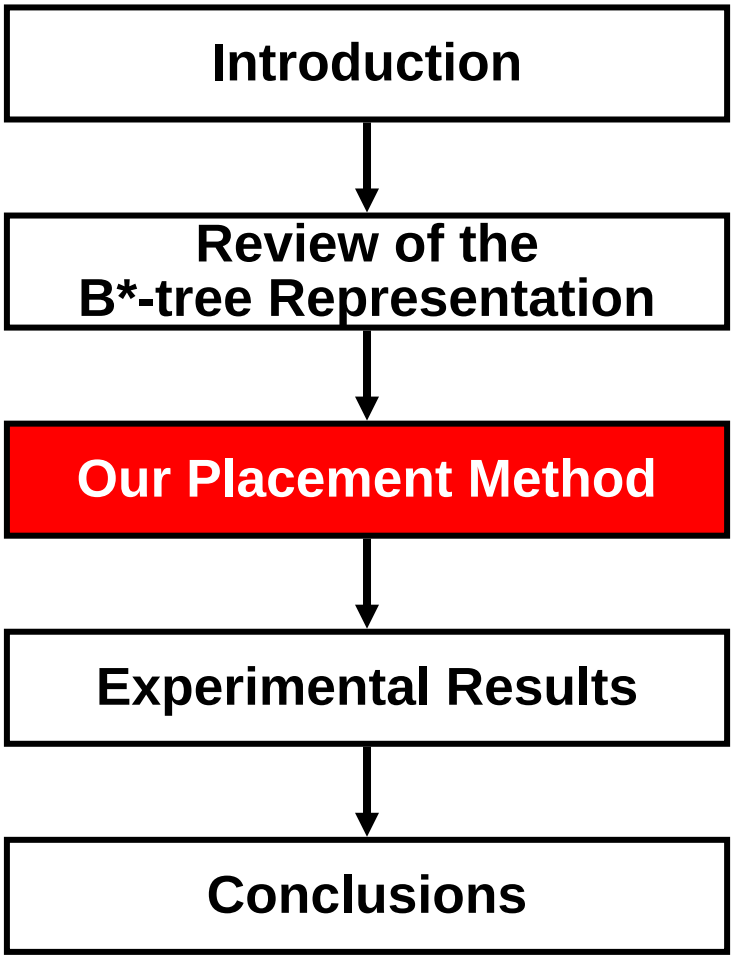
- X-coordinates: Be determined by the tree structure.
  - Left child: the lowest, adjacent block on the right ($x_j = x_i + w_i$).
  - Right child: the first block above, with the same x-coordinate ($x_j = x_i$).
- Y-coordinates: Can be computed in amortized O(1) time by maintaining the contour structure.



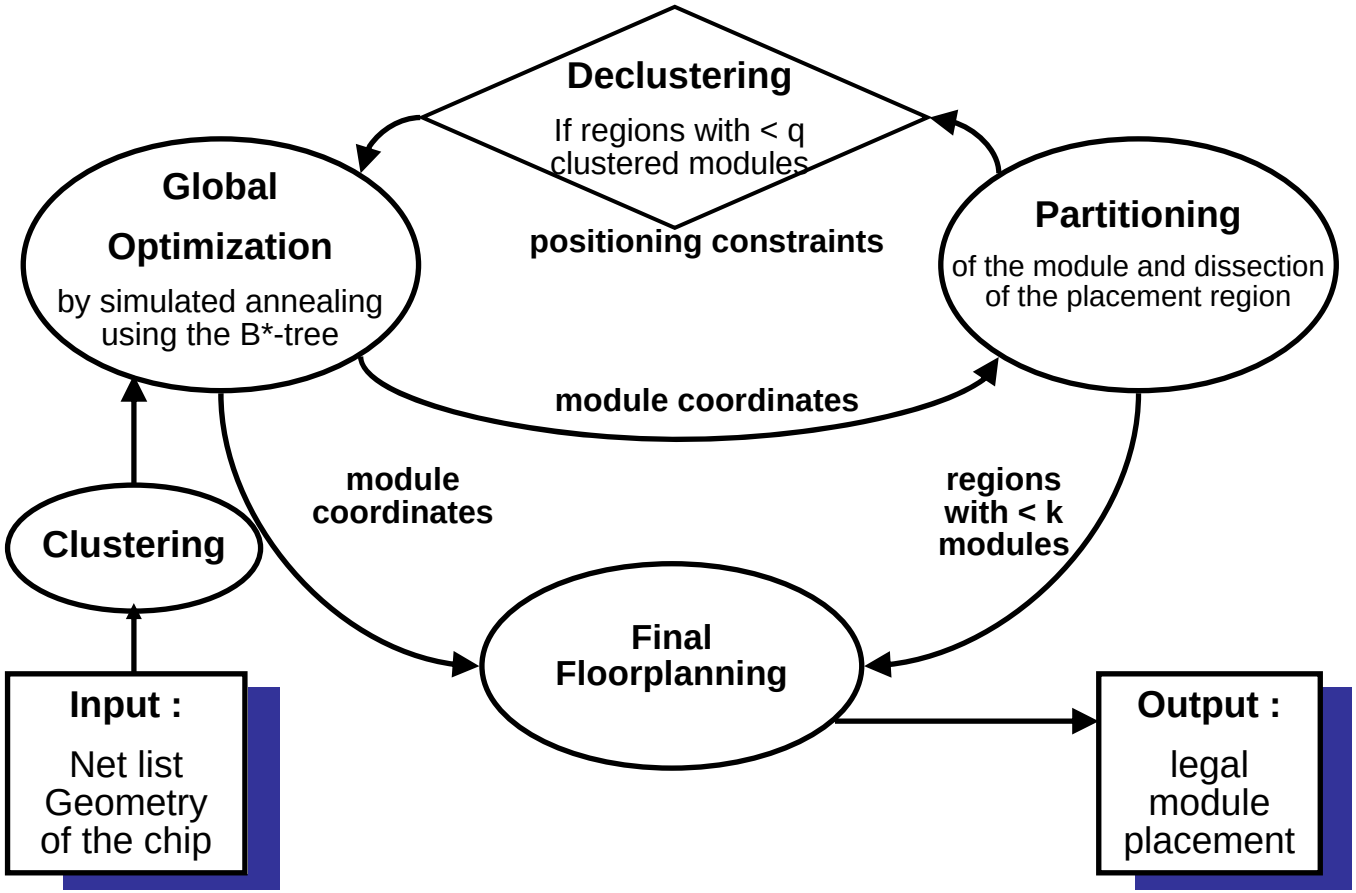




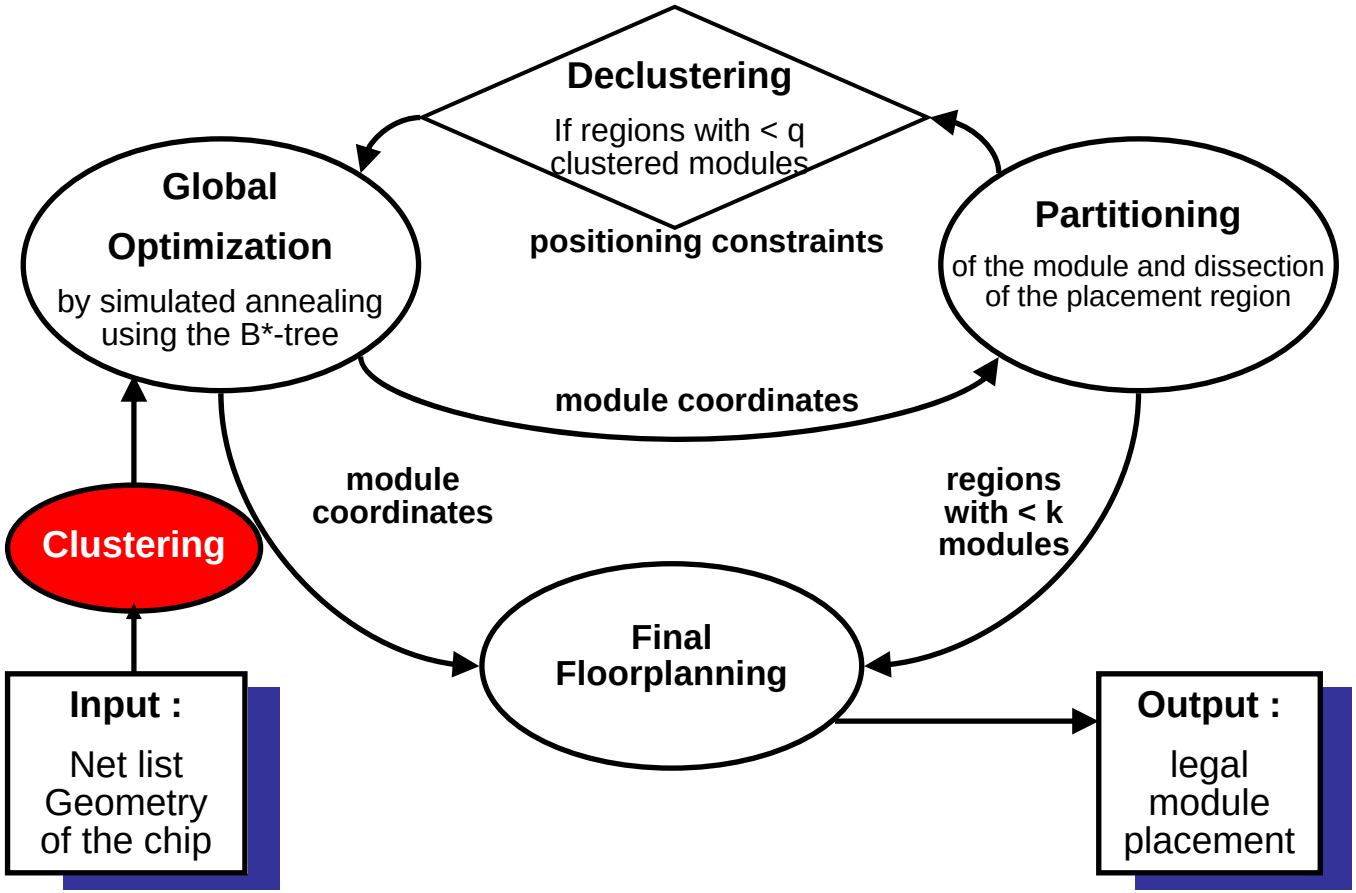
Outline


Introduction
Review of the
B*-tree Representation
Our Placement Method
Experimental Results
Conclusions

Our Algorithm



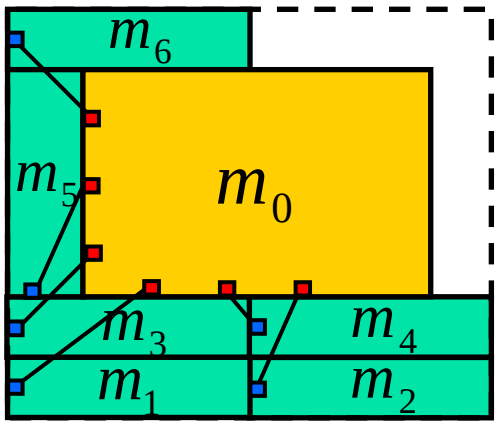
Clustering

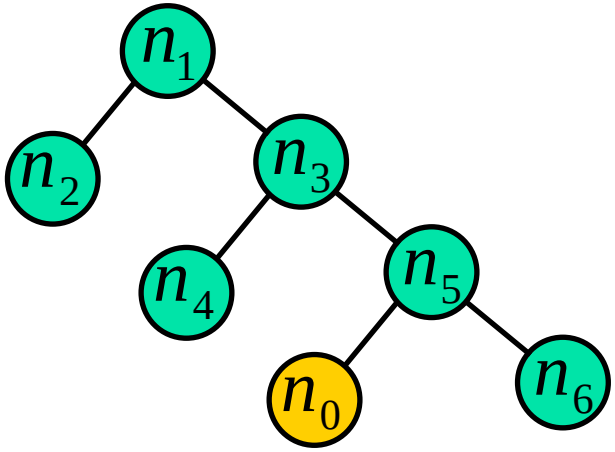

Declustering
If regions with < q clustered modules
Global Optimization
by simulated annealing using the B*-tree
positioning constraints
Partitioning
of the module and dissection of the placement region
module coordinates
module coordinates
regions with < k modules
Clustering
Final Floorplanning
Input :
Net list
Geometry
of the chip
Output :
legal
module
placement

Clustering

- Clustering method: simulated annealing using the B*-tree
- Objective function: $\phi = \alpha s + \beta c$
  - α and β : user-defined parameters.
  - s: area
  - c: the path delays between the block ports and the buffers ports

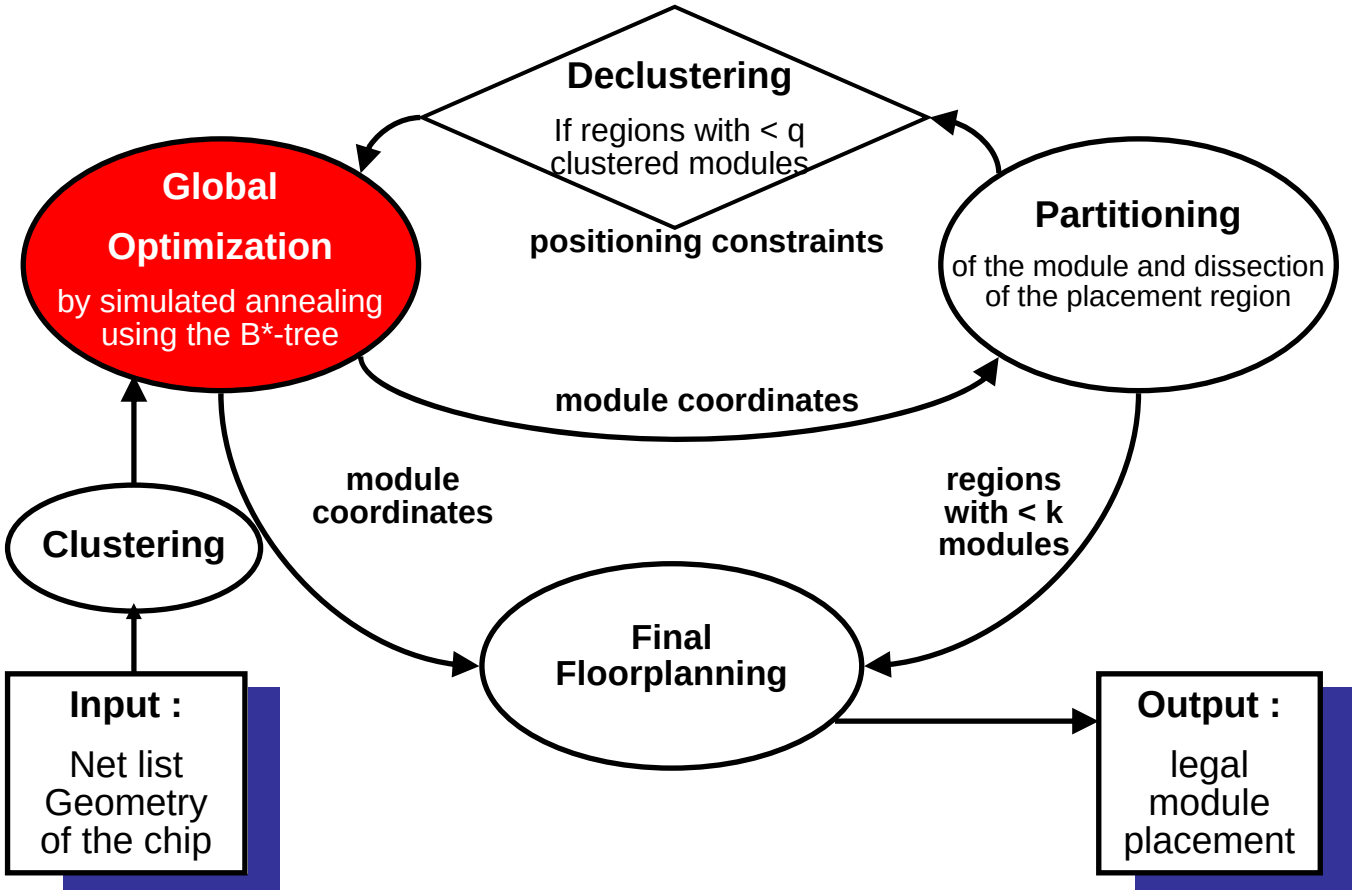
- **block port**
- **buffer port**





m_0 is the block and { m_1, m_2, …, m_6 } are buffers

Global Optimization



Global Optimization

- Blocks are placed by simulated annealing using B*-tree in each subregion
- Two stages in global optimization steps
  - Before declustering, the objective function is Γ
  - After declustering, we modify the objective function as follows:

$$\Gamma' = \Gamma + \gamma \Phi$$

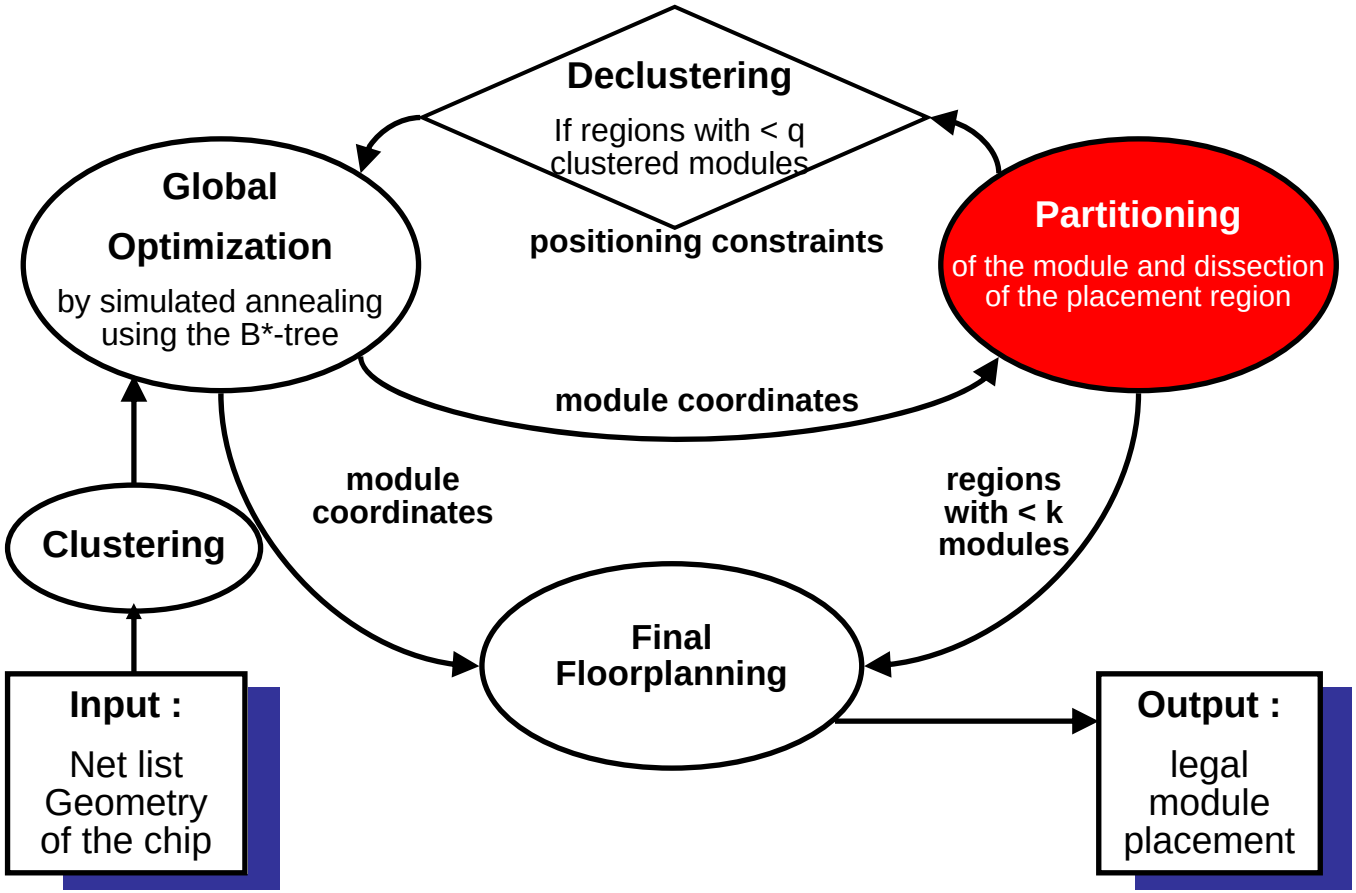
where

$$\Phi = \max(0, W_\rho^m - W_\rho^r) + \max(0, H_\rho^m - H_\rho^r)$$

(fixed-outline constraint)

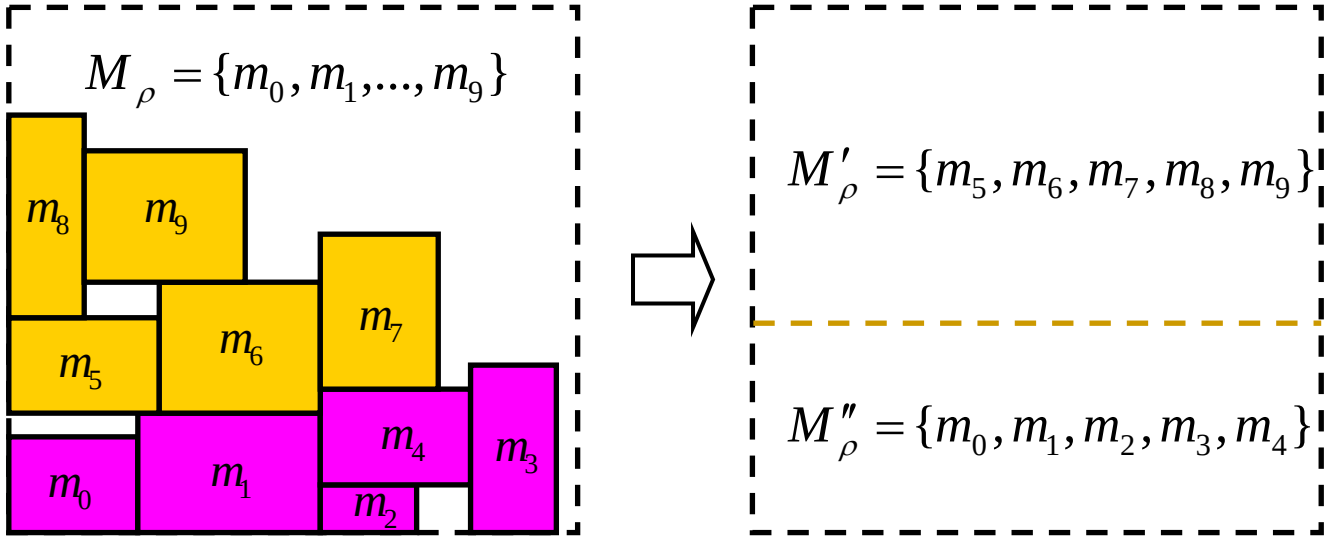
  - Cost Φ is used to force blocks to be packed into the chip.
  - W_ρ^r (H_ρ^r) is the width (height) of regions ρ, and W_ρ^m(H_ρ^r) is the width (height) of the floorplan in region ρ.
  - In order to satisfy the fixed-outline constraint, γ is set to a huge constant

Partitioning



Partitioning

- After the global optimization process, each region is dissected into two subregions.
- Blocks are divided into two groups according to their coordinates.



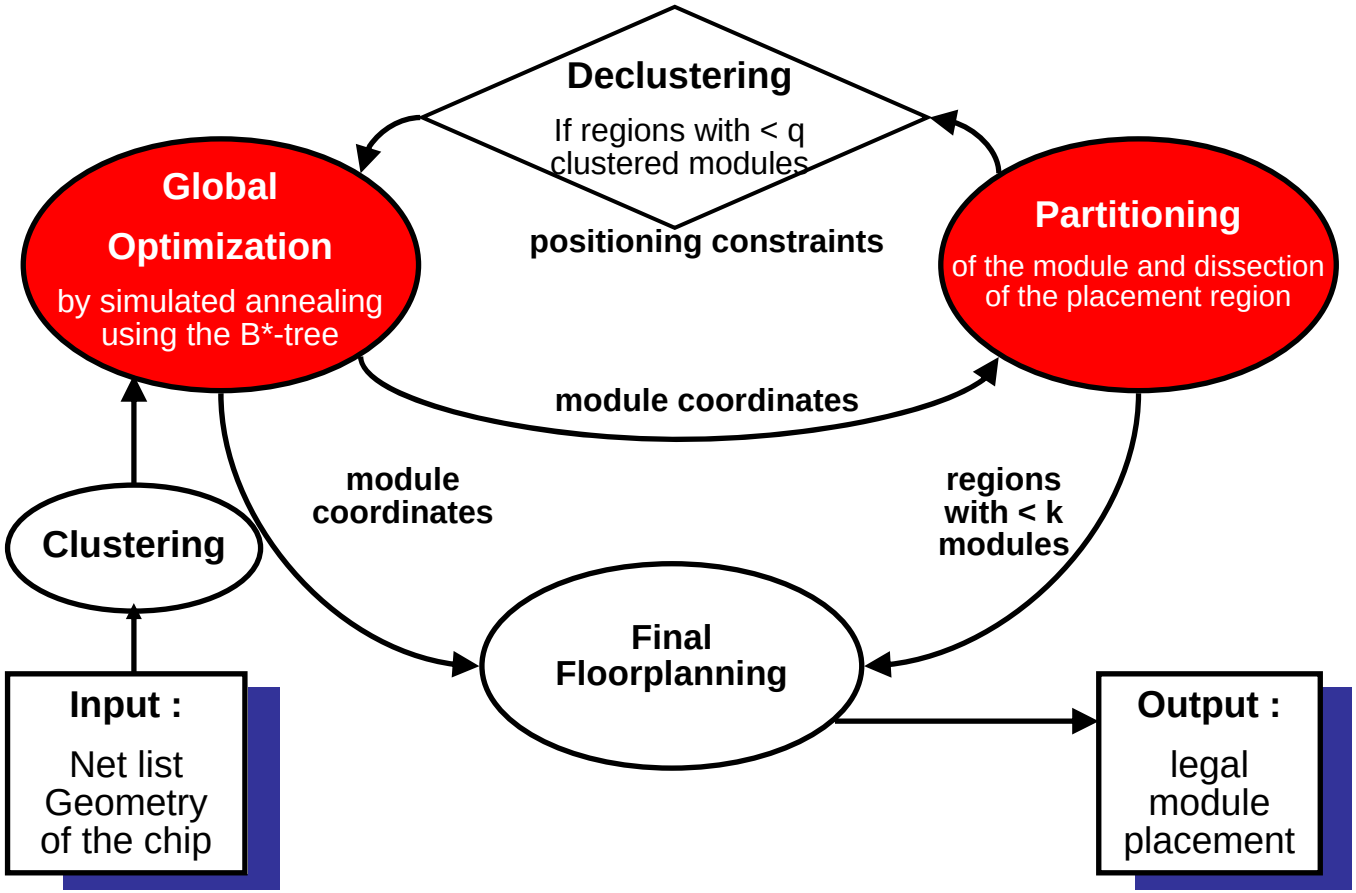
Partitioning (cont'd)

- The summation of the module areas of M'_ρ and M''_ρ are approximately the same, and the area of subregion is dissected correspondingly.

- If $W^r_\rho > H^r_\rho$ 
  - Sort blocks according to the x-coordinates.
  - Cut the region vertically.

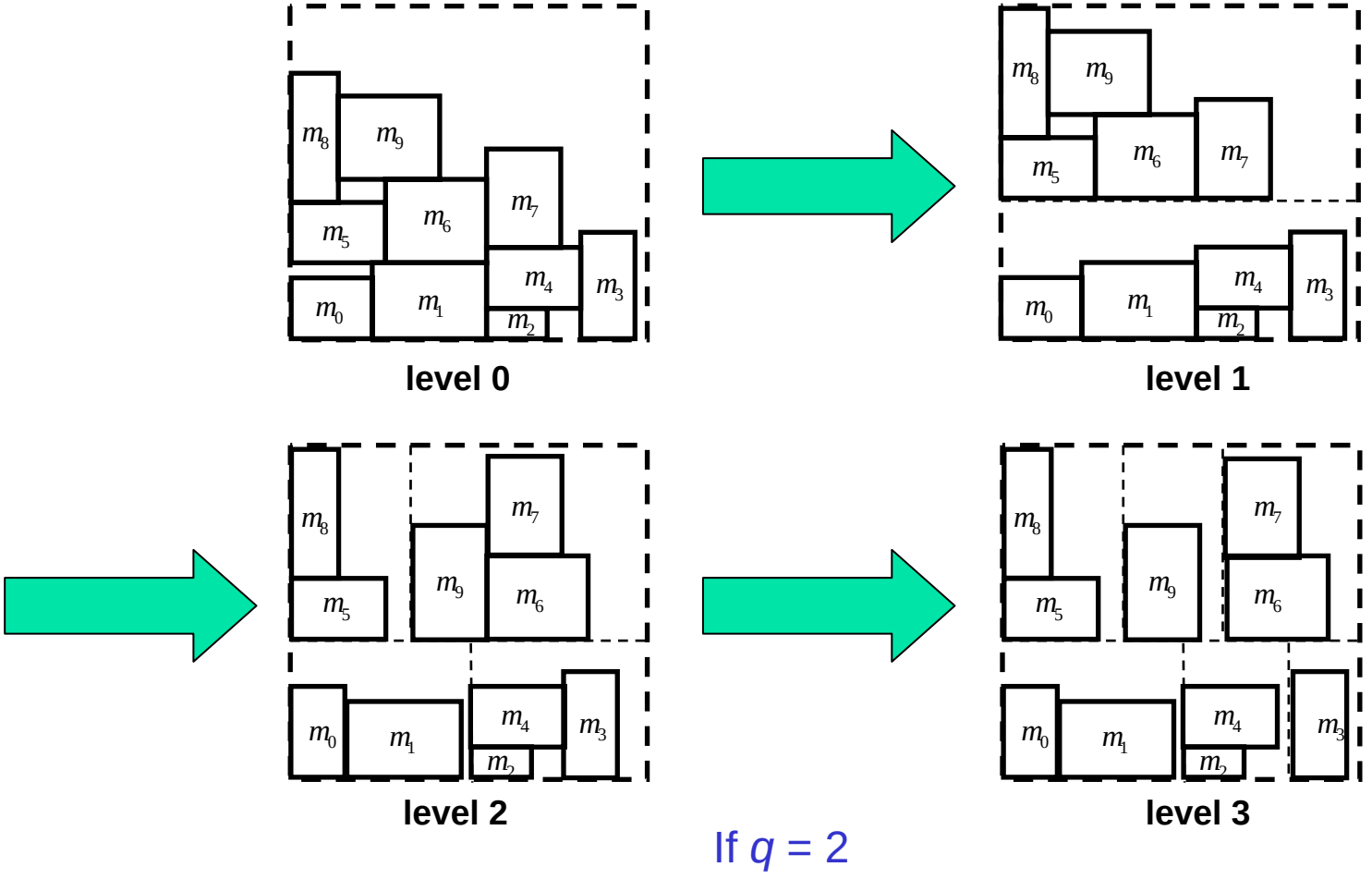
- If $H^r_\rho > W^r_\rho$ 
  - Sort blocks according to the y-coordinates.
  - Cut the region horizontally.

Alternate Global Optimization and Partitioning Steps

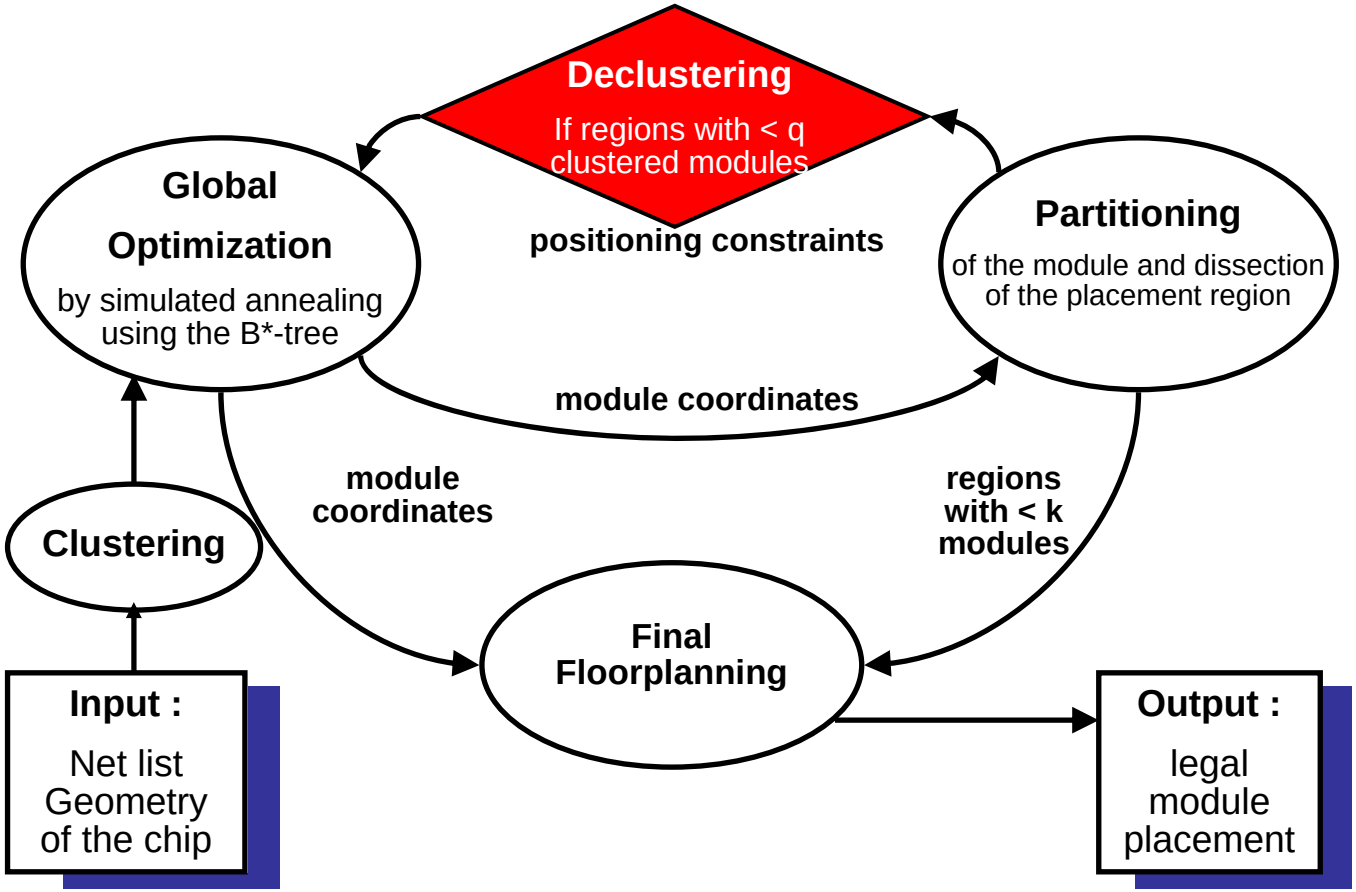


Alternate Global Optimization and Partitioning Steps

- Alternate global optimization and partitioning steps until the number of blocks in every region is smaller than q.

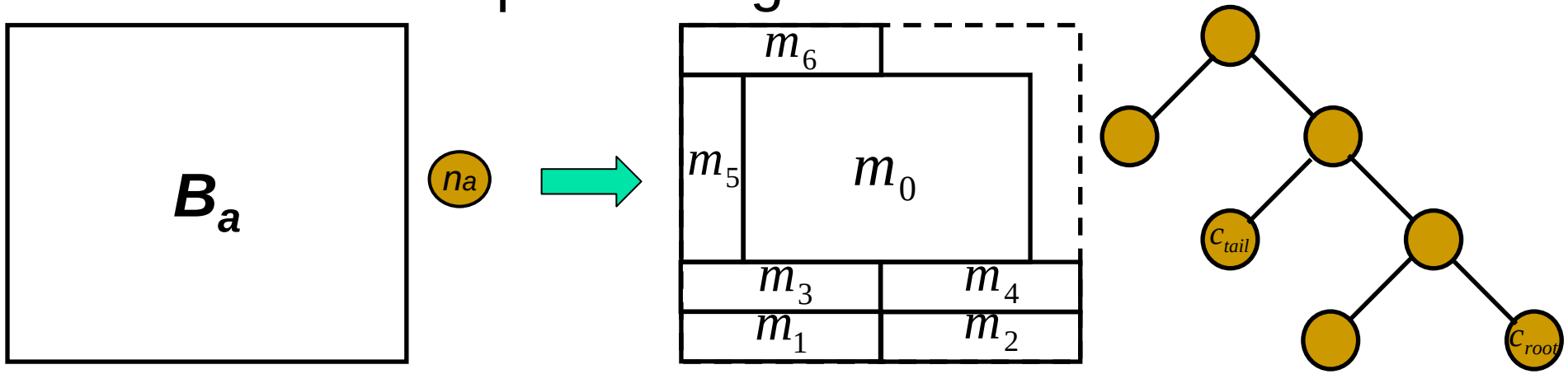


Declustering



Declustering

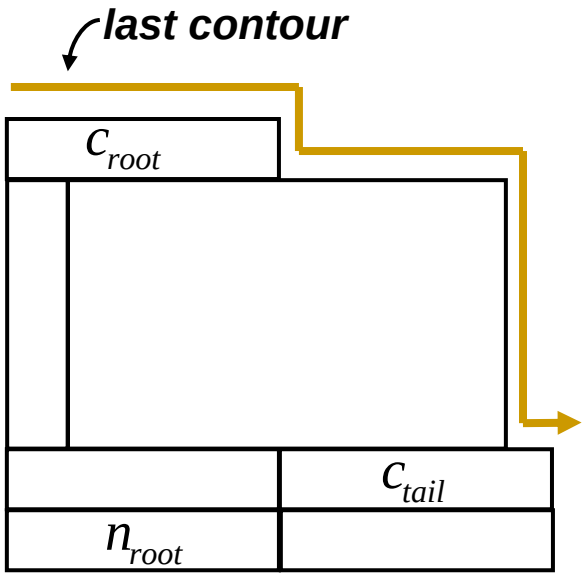
- Expand each *block node* in the B*-tree into a subtree representing the clustered blocks.



- Construct a new larger B*-tree using these subtrees.
- There are two kinds of relations between two connected nodes in the original tree.
  - Left relation (left child) and right relation (right child)

Declustering – Last Contour

- Record the last contour when performing B*-tree packing
  - The root node of the contour c_{root} represents the left- and top-most cell of the clustered block.
  - The tail node of the contour c_{tail} represents the right- and top-most cell of the clustered block.



a clustered block

Denomination:

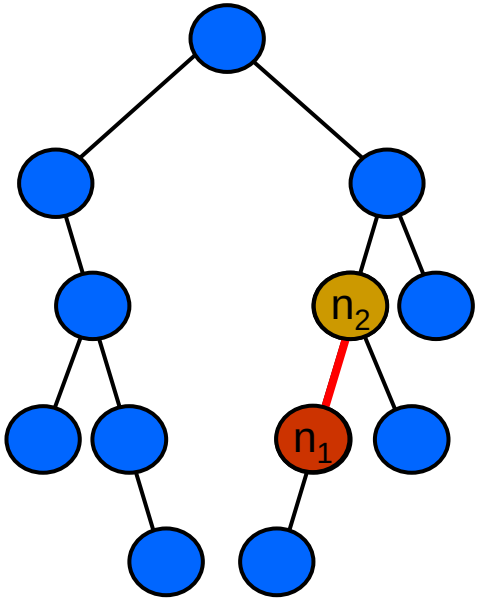
c_{root} : the root node of the last contour

c_{tail} : the last node of the last contour

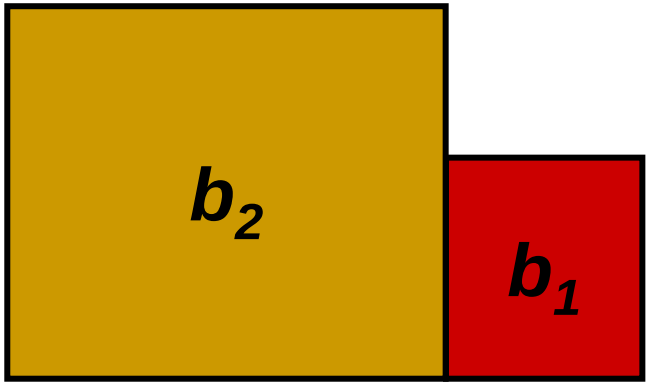
n_{root} : the root node of the subtree

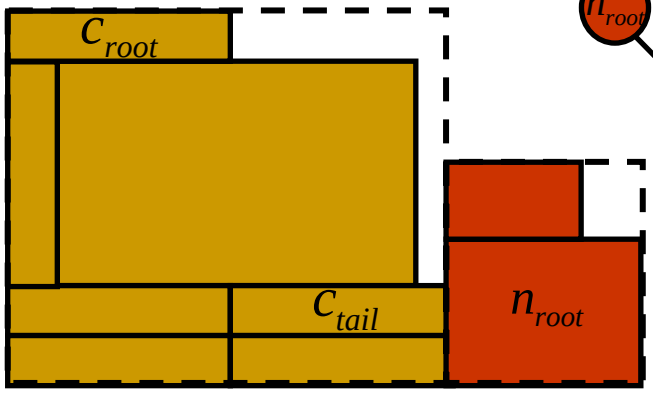
Declustering – Left Relation

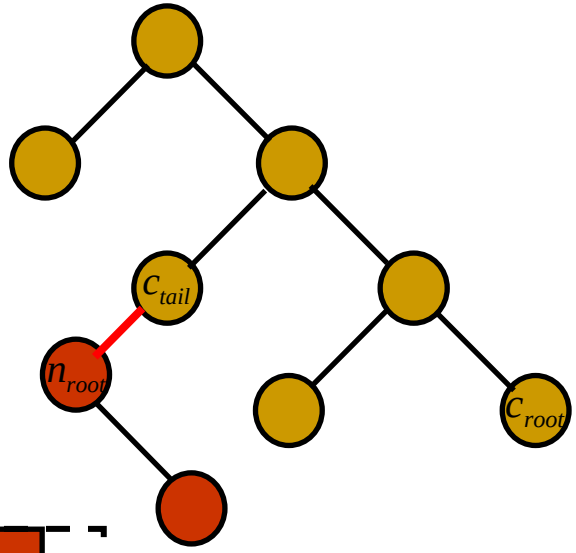
- If the block node n_1 is the left child of the block node n_2
  - Connect the root of n_1 subtree to the left child of the node c_{tail}



The original tree

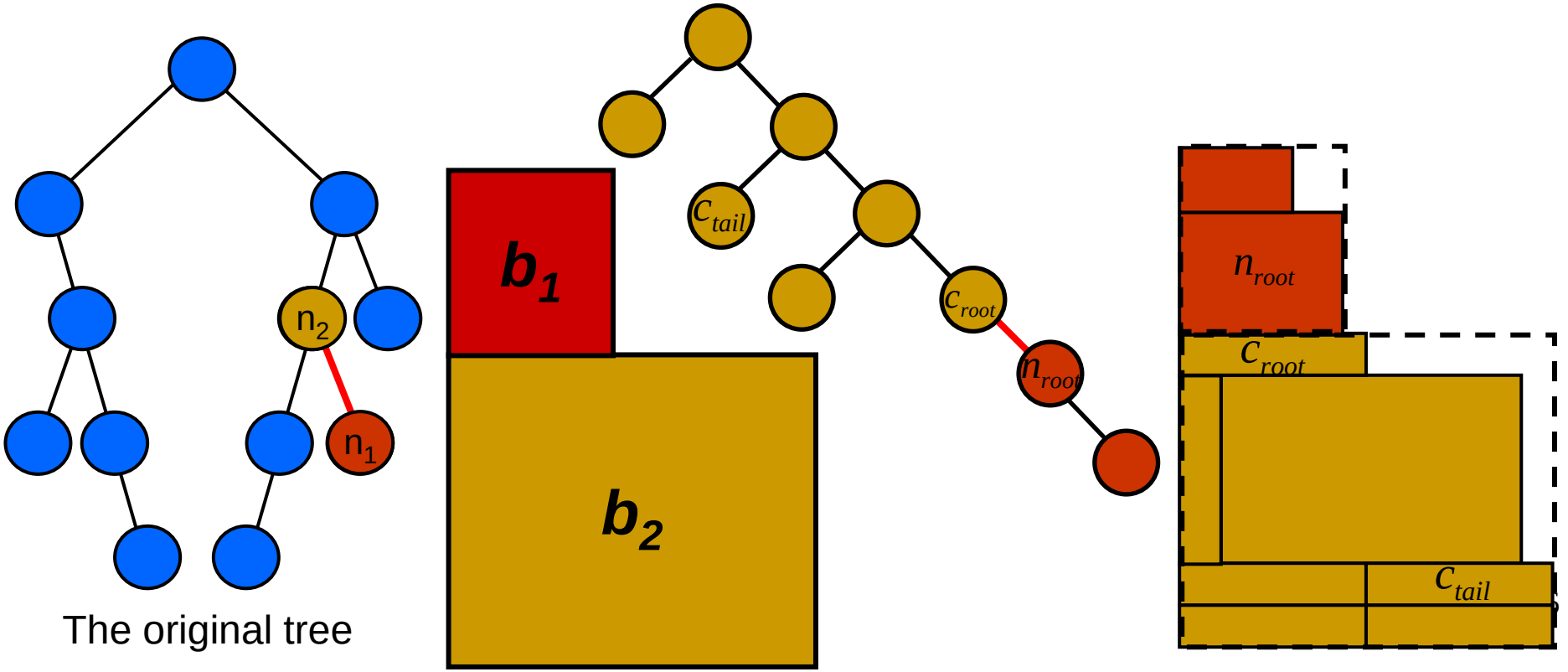






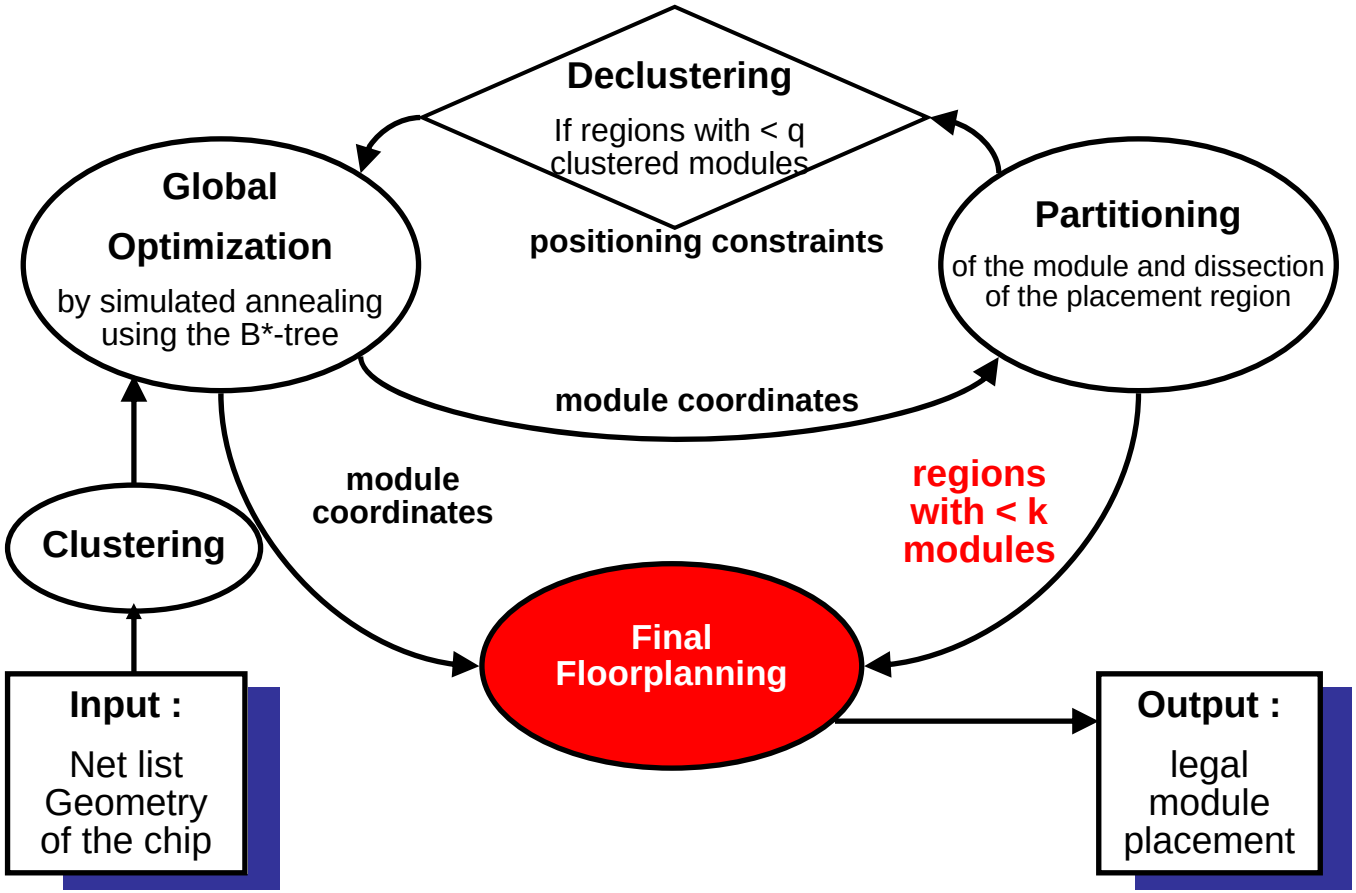
Declustering – Right Relation

- If the block node n_1 is the right child of the block node n_2
  - Connect the root of n_1 subtree to the right child of the node c_{root}



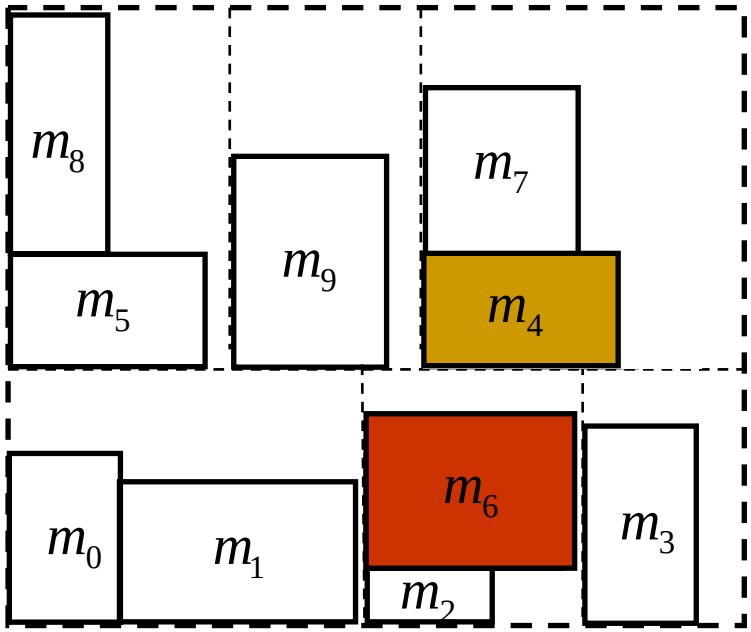
The original tree

Final Floorplanning



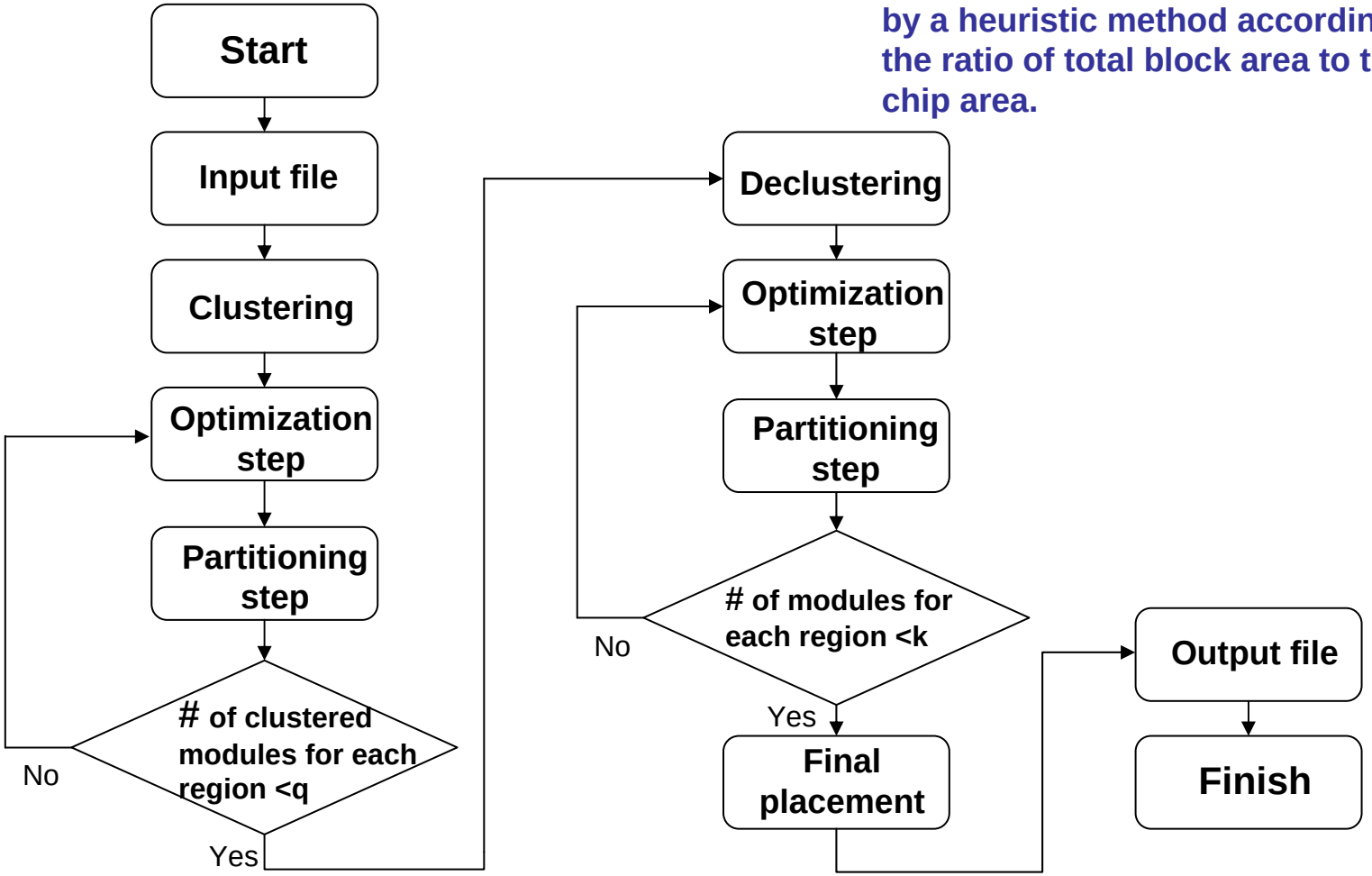
Final Floorplanning

- Select two blocks randomly and swap them.
- Use the same objective function as the one in the global optimization step.
- Accept the solution only if it does not violate the outline constraint.
- Re-compute the coordinates of blocks at the changed subregions.

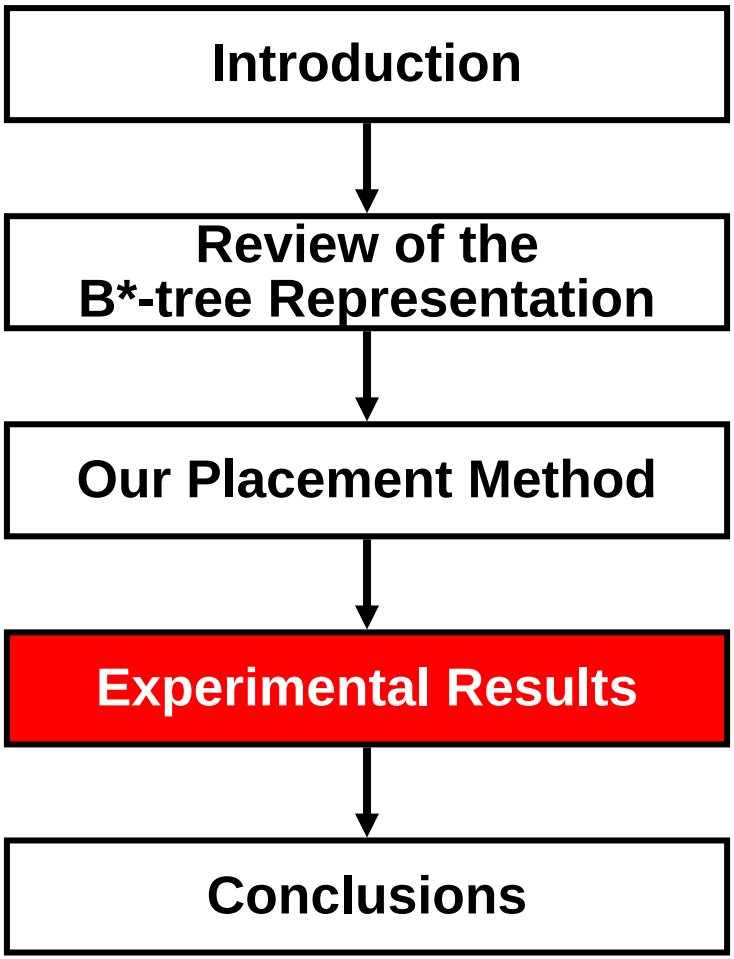


Summary of Our Algorithm

The parameters *q* and *k* are defined by a heuristic method according to the ratio of total block area to the chip area.



Outline


Introduction
Review of the
B*-tree Representation
Our Placement Method
Experimental Results
Conclusions

Experimental Setup

- Platform
  - 1.2GHz SUN Blade 2000 with 8GB memory
- Use seven benchmark circuits provided by the foundry UMC and the design service company Faraday.
- Compare the following algorithms
  - B*-tree alone (DAC-00)
  - TCG alone (DAC-01)
  - Our method

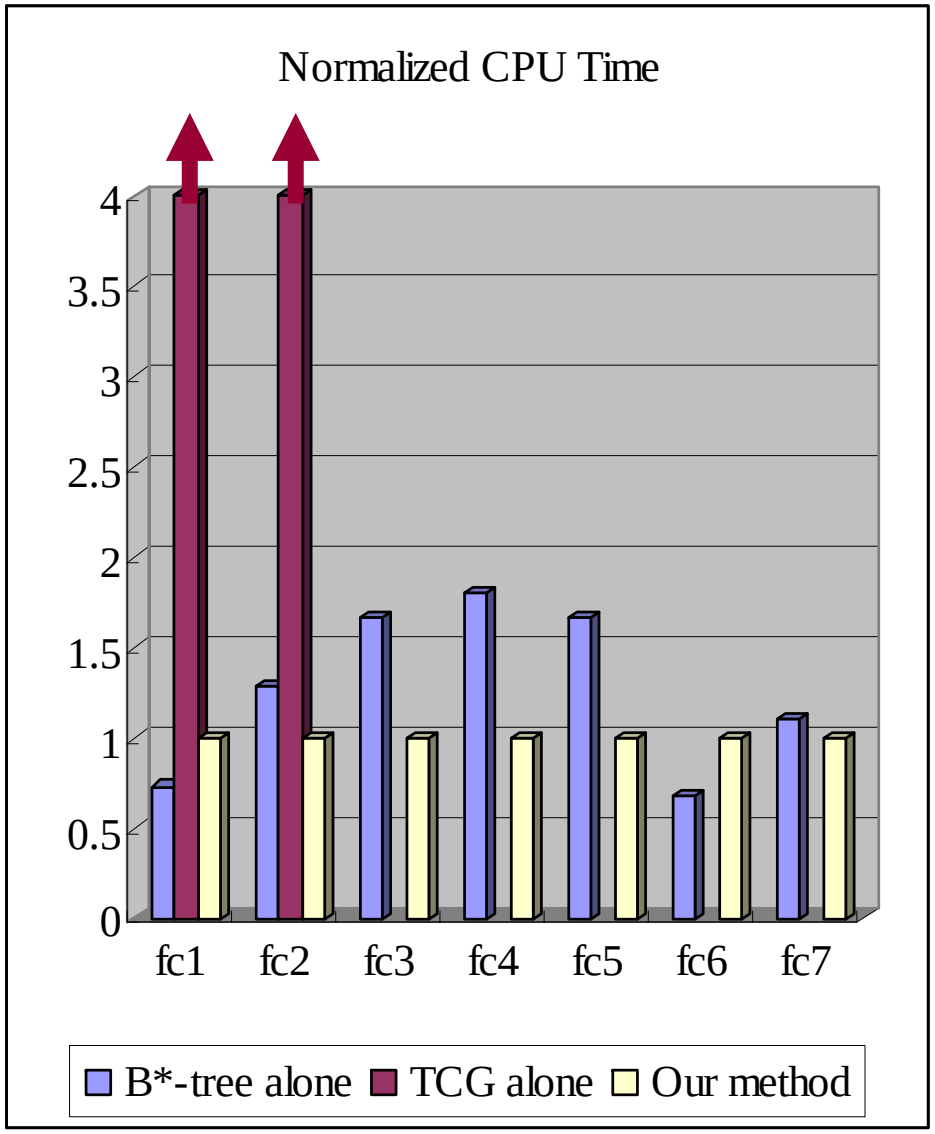
Test Cases

- # blocks + # buffers ranges from 31 to 412.
- Chip density is from 42% to 88%.

Circuit	# blocks	# buffers	chip area	Blocks area/chip area	α	β
fc1	6	25	1040x1040	0.4216	0.5	0.5
fc2	12	168	3440x3440	0.5598	0.5	0.5
fc3	23	320	4240x4240	0.6584	0.7	0.3
fc4	28	384	4440x4440	0.7276	0.7	0.3
fc5	28	384	4440x4440	0.7276	0.7	0.3
fc6	28	384	4040x4040	0.8788	0.7	0.3
fc7	28	384	4040x4040	0.8788	0.7	0.3

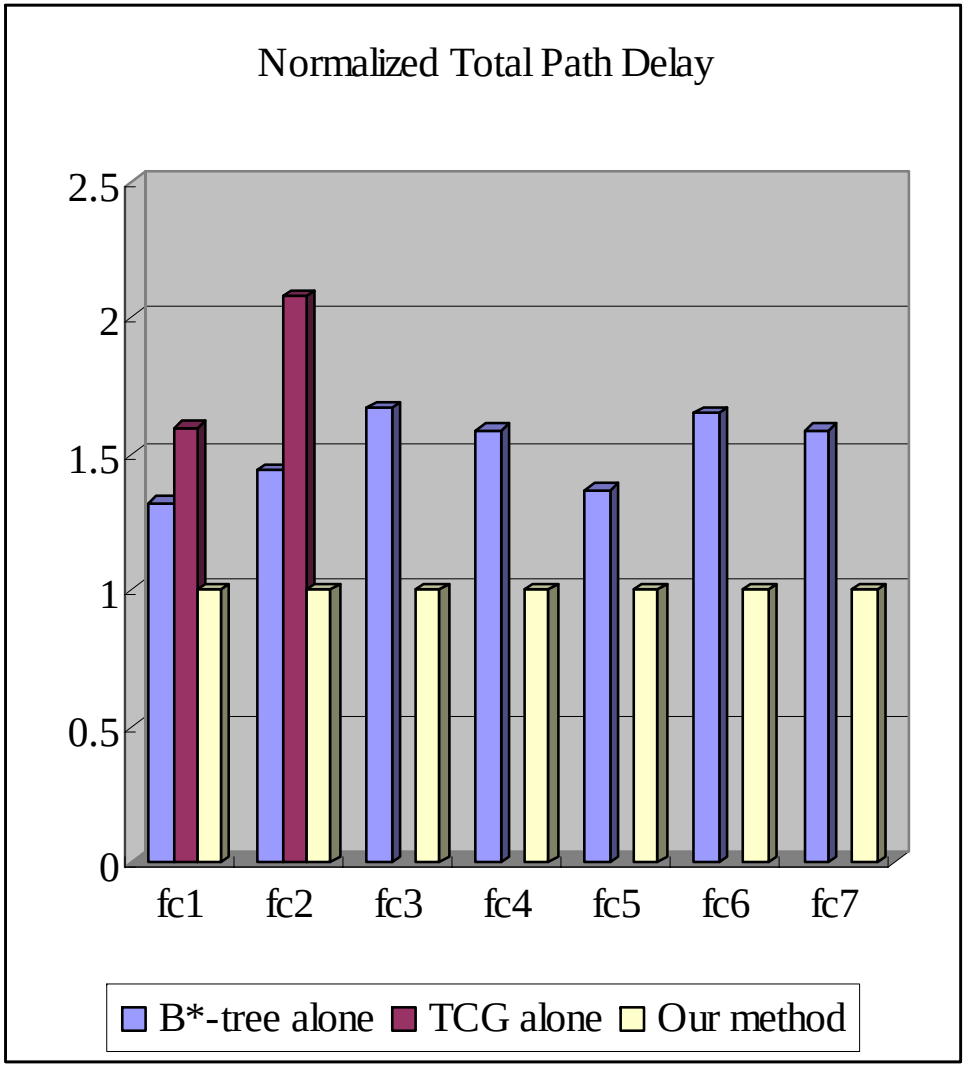
CPU Time Comparison

- Because of the large problem size, the TCG alone algorithm is only feasible for the first two cases.
- On average, B*-tree alone algorithm needs 28% more CPU time than our method does.



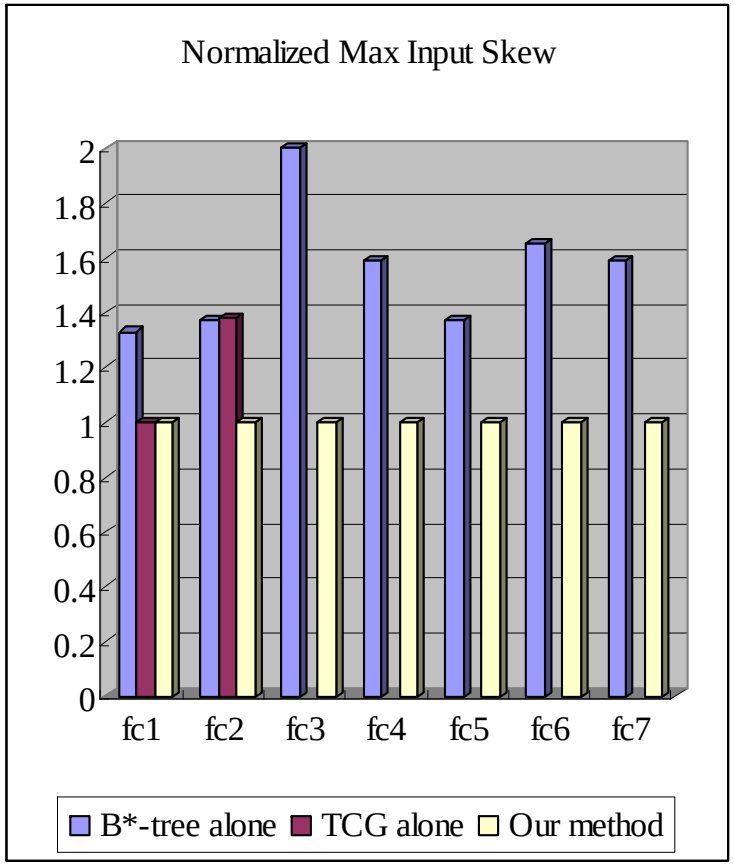
Total Path Delay Comparison

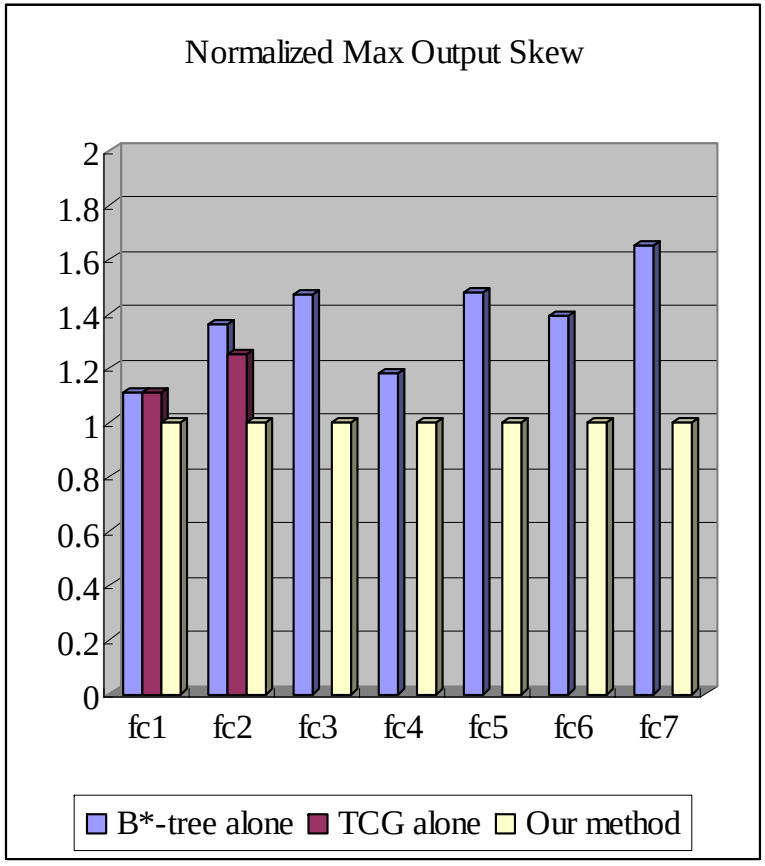
- On average, the path delay obtained by B*-tree alone (TCG alone) is 1.52 (1.84) times than ours.



Max Input/Output Skew

- On average, the max skew obtained by B*-tree alone (TCG alone) is 1.47 (1.19) times than ours.



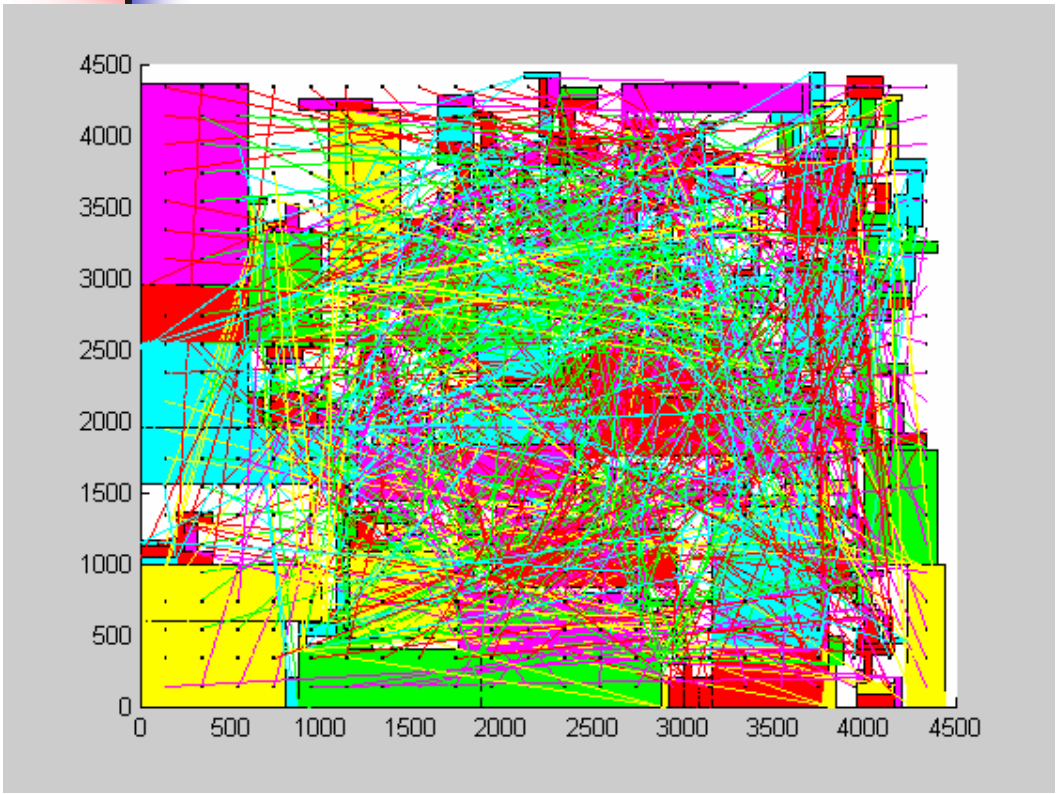


Result Summary

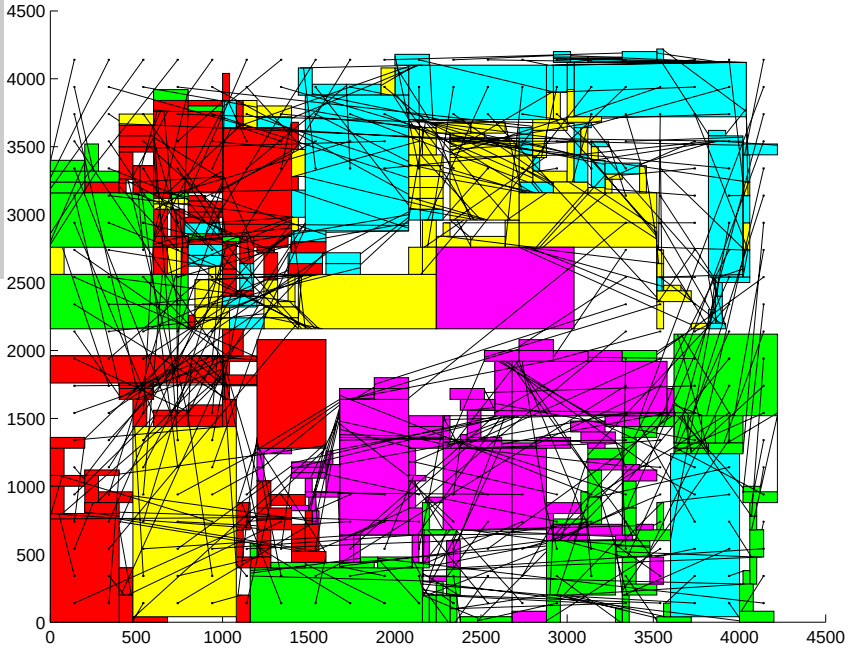
- On average, the total cost obtained by B*-tree alone (TCG alone) is 2.04 (1.52) times than ours.

	CPU time	Total path delay	Max input skew	Max output skew	Cost Γ
Our Method	1.00	1.00	1.00	1.00	**1.00**
B*-tree alone	1.28	1.52	1.49	1.38	**2.04**
TCG alone	332.80	1.84	1.19	1.18	**1.52**

Placement Results

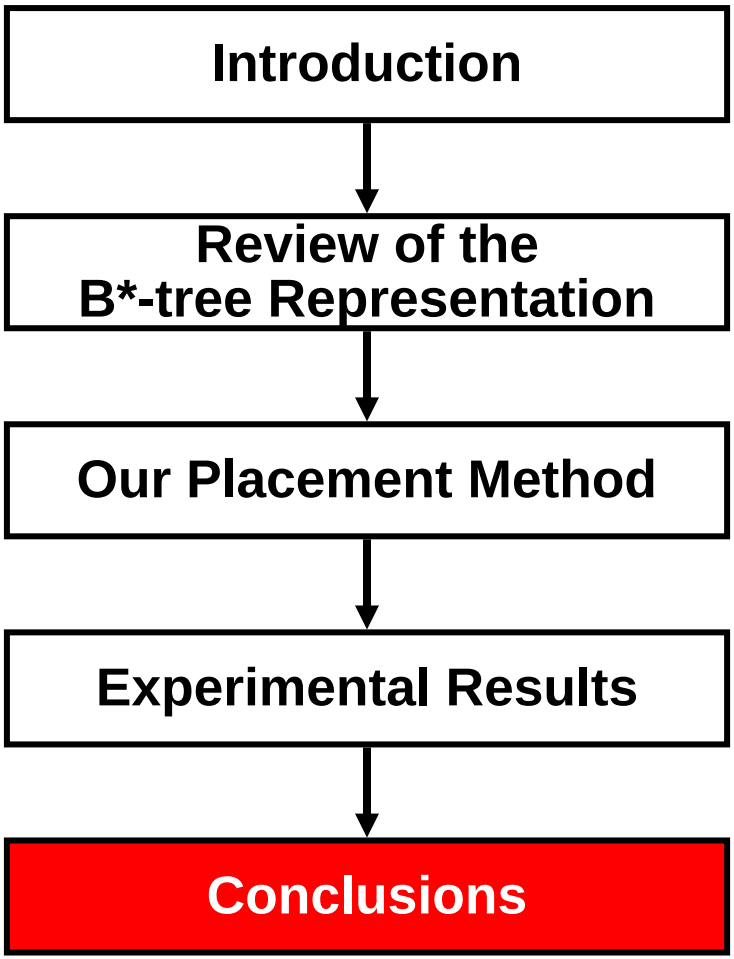


- **fc5**
- # blocks +
  # buffers = 412



- **fc3**
- # blocks +
  # buffers = 343

Outline


Introduction
Review of the
B*-tree Representation
Our Placement Method
Experimental Results
Conclusions

Conclusions

- We presented a B*-tree based hierarchical top-down method for the block and input/output buffer floorplanning for flip-chip designs.
- Experimental results based on real industrial flip-chip designs have shown the effectiveness and efficiency of our algorithm.
- Future work lies in developing other heuristics to slice the chip to further improve the results.

Thank you for your attention.

Our Algorithm (cont'd)

- Clustering step:
  - Cluster a block and its corresponding I/O buffers to a clustered block.
  - In the following global optimization and partitioning step, a clustered block is represented by a *block node*.
- Global optimization and Partitioning step:
  - After the clustering steps, we go into the main steps of alternating and interacting global optimization and partitioning step.
  - In global optimization step, we place blocks based on simulated annealing using the B*-tree to minimize the objective function.
  - In partitioning step, the chip is dissected into two subregions. The blocks are divided into two groups according to their coordinates and are placed in respective subregions.

Our Algorithm (cont'd)

- Declustering step:
  - Until each region contains at most *q* clustered blocks, we decluster these clustered blocks.
  - A clustered block represented by a *block node* is expanded into a subtree representing the clustered block's component which are constructed at the clustered step.
- Final Floorplanning
  - After the declustering step, the global optimization and partitioning steps repeat until each region contains at most *k* blocks.
  - In the final floorplanning step, we refine the floorplan by purturbing blocks inside a subregion as well as in different regions

The parameters *q* and *k*

- Define the *q* and *k* by a heuristic method

```
q = # clustered_blocks;
k = # blocks;
r = total_blocks_area/chip_area  /*utilization ratio*/
 If ( r < 0.75 );
     q = 10 x r;
     k = 10 x r x (# blocks / # clustered_blocks)
else
     k = 10 x r x (# blocks / # clustered_blocks)
q = max{q,3};   /* the smallest q = 3*/
K = max{k,20};    /*the smallest k = 20*/
```

Ckt		B*-tree alone		TCG alone		Our Method	
fc1	Tot. path delay	23390	1.32	28430	1.60	17760	1.0
	Max. input skew	160	1.33	120	1.00	120	1.0
	Max. output skew	100	1.11	100	1.11	90	1.0
	Cost Γ	2.95e+06	1.46	2.64e+06	1.31	2.01e+06	1.0
	CPU Time	1 s	0.73	32 s	33.83	1 s	1.0
fc2	Tot. path delay	521030	1.44	750450	2.08	361650	1.0
	Max. input skew	1360	1.37	1390	1.38	1010	1.0
	Max. output skew	1890	1.36	1740	1.25	1390	1.0
	Cost Γ	2.97e+08	1.79	2.86e+08	1.72	1.66e+08	1.0
	CPU Time	20 s	1.29	9944 s	631.7	16 s	1.0
fc3	Tot. path delay	1033800	1.67	NR	-	619200	1.0
	Max. input skew	3320	2.00	NR	-	1660	1.0
	Max. output skew	2500	1.47	NR	-	1700	1.0
	Cost Γ	1.24e+09	3.00	NR	-	4.14e+08	1.0
	CPU Time	85 s	1.66	> 10 hr	-	51 s	1.0

Ckt		B*-tree alone		TCG alone		Our Method	
fc4	Tot. path delay	1153560	1.59	NR	-	726040	1.0
	Max. input skew	3380	1.54	NR	-	2190	1.0
	Max. output skew	2820	1.18	NR	-	2380	1.0
	Cost Γ	1.39e+09	1.84	NR	-	7.54e+08	1.0
	CPU Time	130 s	1.80	> 10 hr	-	72 s	1.0
fc5	Tot. path delay	969140	1.37	NR	-	707430	1.0
	Max. input skew	3300	1.91	NR	-	1730	1.0
	Max. output skew	3200	1.48	NR	-	2160	1.0
	Cost Γ	1.51e+09	2.71	NR	-	5.57e+08	1.0
	CPU Time	130 s	1.66	> 10 hr	-	78 s	1.0
fc6	Tot. path delay	1233720	1.65	NR	-	745880	1.0
	Max. input skew	3580	1.19	NR	-	3000	1.0
	Max. output skew	4360	1.39	NR	-	3140	1.0
	Cost Γ	2.26e+09	1.69	NR	-	1.34e+09	1.0
	CPU Time	108 s	0.68	> 10 hr	-	160 s	1.0

NR: No Result

Ckt		B*-tree alone		TCG alone		Our Method	
fc7	Tot. path delay	1159560	1.59	NR	-	729180	1.0
	Max. input skew	3880	1.11	NR	-	3500	1.0
	Max. output skew	4720	1.65	NR	-	2860	1.0
	Cost Γ	2.65e+09	1.82	NR	-	1.45e+09	1.0
	CPU Time	251 s	1.11	> 10 hr	-	226 s	1.0

- Because of the higher complexity, the TCG based floorplanner alone is only feasible for the first two cases
- The B*-tree based algorithm (the TCG based algorithm) in the overall cost of 2.04 times (1.52 times) of that of our algorithm
- The B*-tree based algorithm (the TCG based algorithm) needs 1.28 times (more than 332 times) of our CPU times