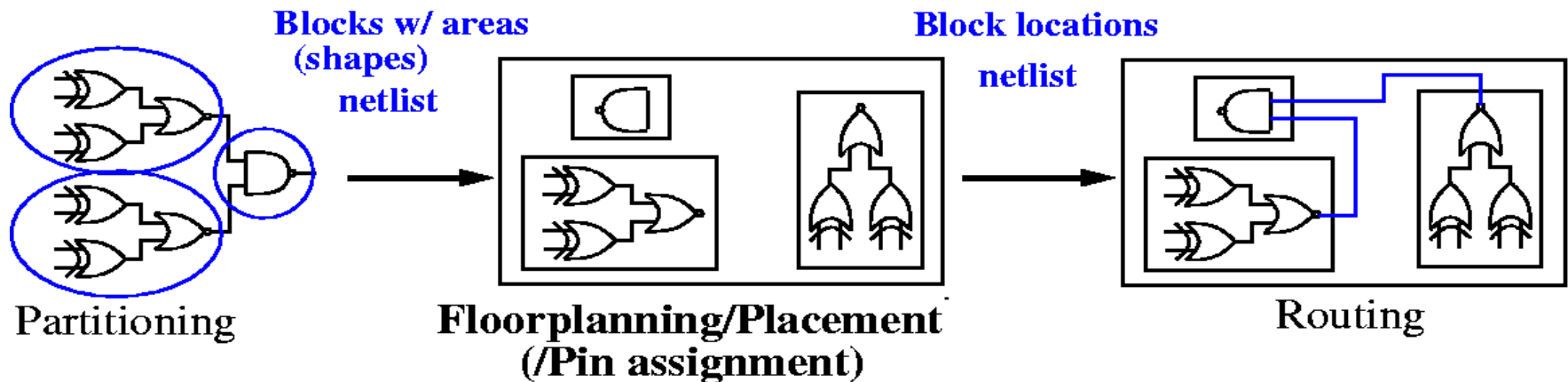


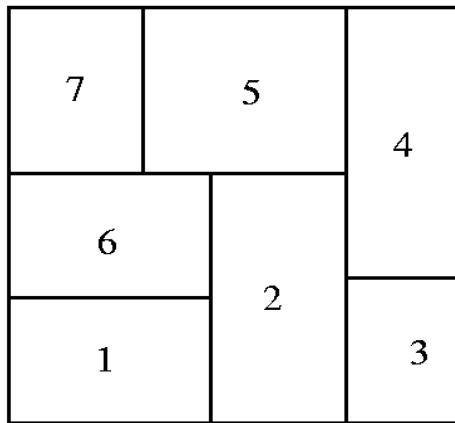
# Floorplanning

- Partitioning leads to
  - Modules (or called blocks) with well-defined areas and shapes (*hard modules*).
  - Modules with approximated areas and no particular shapes (*soft modules*).
  - A netlist specifying connections between the modules.
- Objectives
  - Find **locations** for all modules, as well as **orientations (if allowable)** for hard modules.
  - Find shapes (and pin locations) of the soft modules.

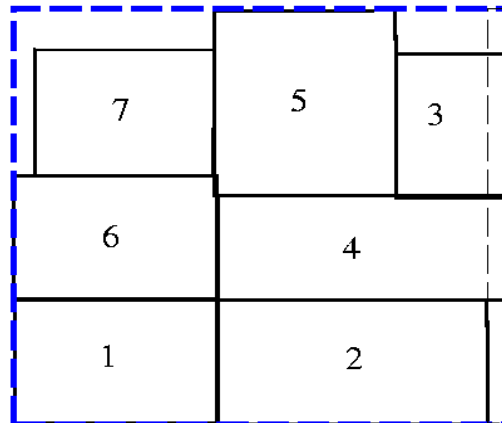


# Floorplanning Problem

- Inputs:
  - A set of modules, hard or soft.
  - Pin locations of hard modules.
  - A netlist.
- Objectives: Minimize area, reduce wirelength for (critical) nets, maximize routability (minimize congestion), determine shapes of soft modules.



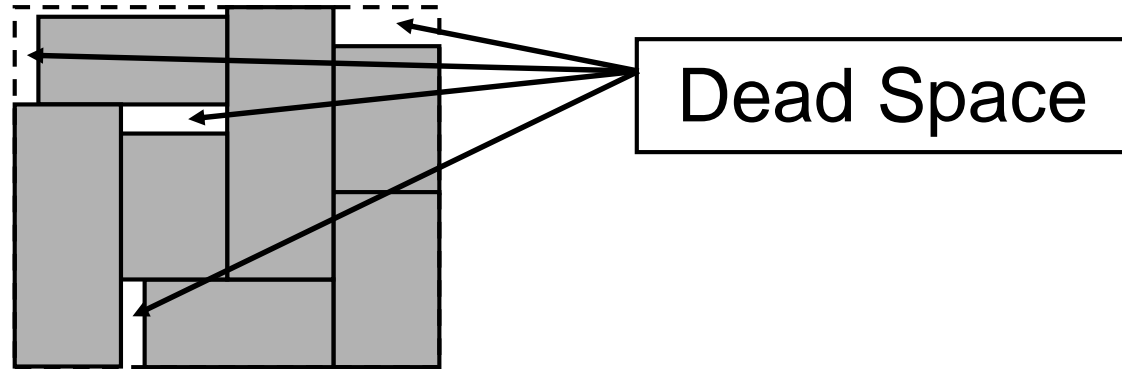
An optimal floorplan,  
in terms of area



A non-optimal floorplan

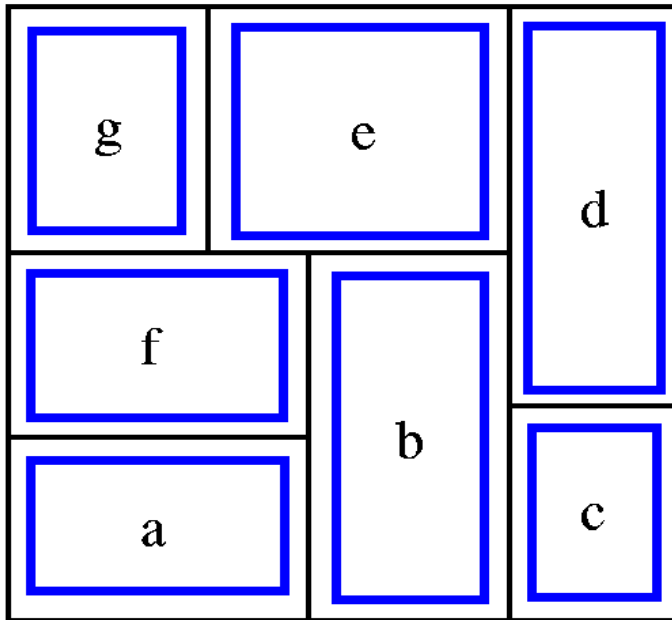
# Dead Space




- The space that is wasted.

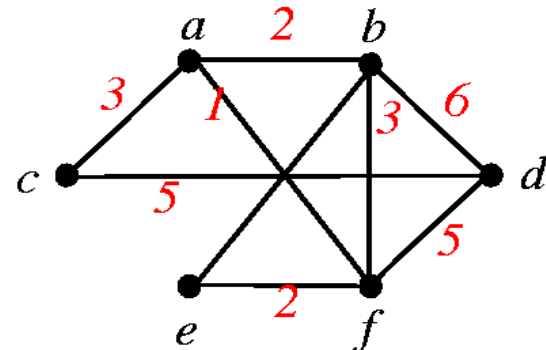


- Minimizing area is the same as minimizing dead space.
- Percentage of dead space  
$$= ((\text{Area of resulting rectangle} / \text{Total area of all modules}) - 1) \times 100\%.$$

# Floorplan Design

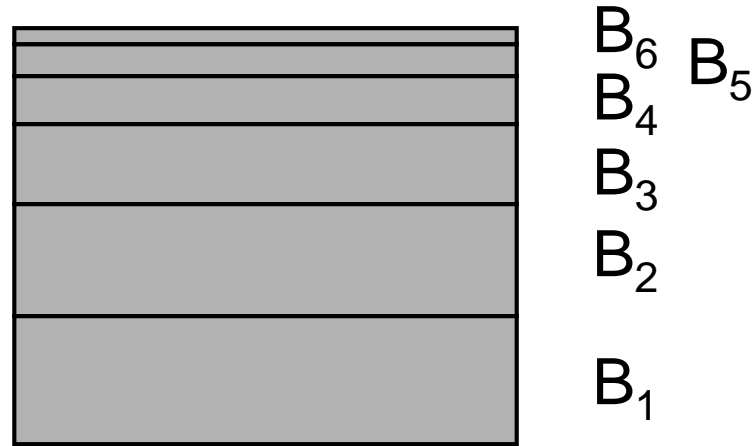


- *Modules:*   $x$   $y$
- *Area:*  $A=xy$
- *Aspect ratio:*  $r \leq y/x \leq s$
- *Rotation:*  
- *Module connectivity*



# Bounds on Aspect Ratios

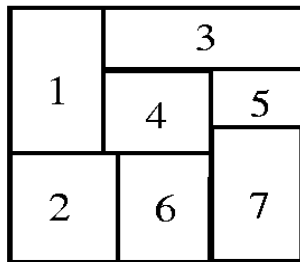
- If there is no bounds on aspect ratios, we can always pack modules completely tight (i.e., no dead space).



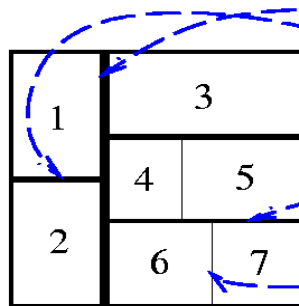
- We do not want to layout a module as a long strip.

# Terminology

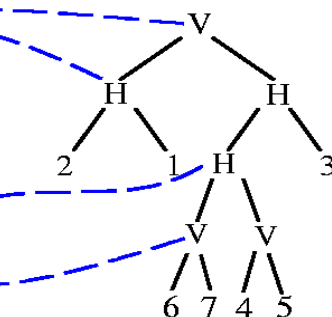
- **Rectangular dissection:** Subdivision of a given rectangle by a finite # of horizontal and vertical line segments into a finite # of non-overlapping rectangles.
- **Slicing structure:** a rectangular dissection that can be obtained by repetitively subdividing rectangles horizontally or vertically.
- **Slicing tree:** A binary tree, where each internal node represents a vertical cut line or horizontal cut line, and each leaf a basic rectangle.
- **Skewed slicing tree:** A slicing tree in which no node and its right child are the same type of cut line.



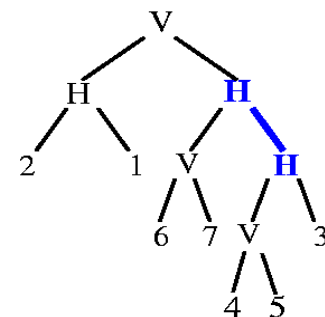
Unit 4 Non-slicing floorplan



Slicing floorplan



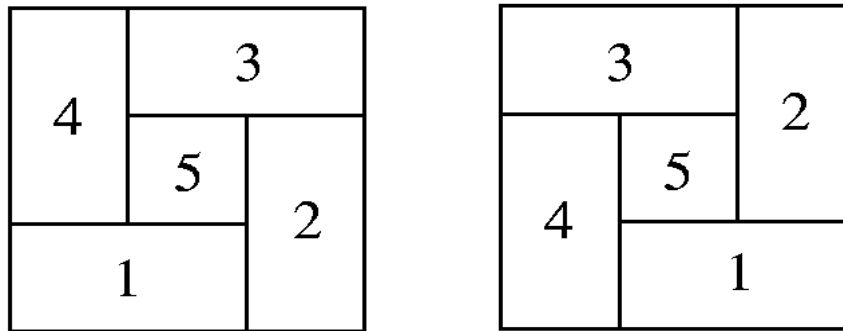
A slicing tree (skewed)



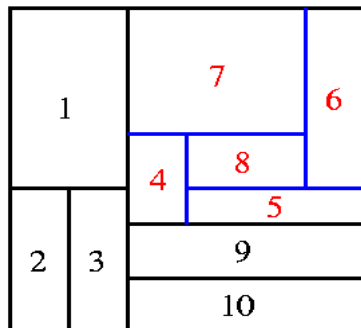
Another slicing tree (non-skewed)

# Terminology (cont'd)

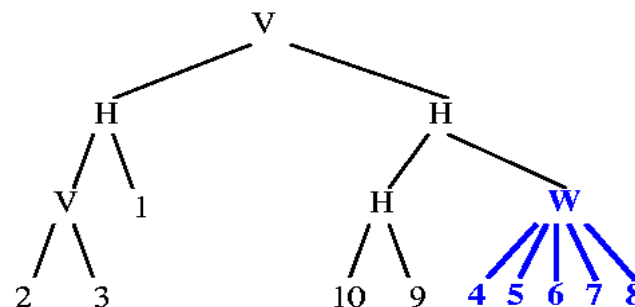
- **Wheel:** The smallest non-slicing floorplans (Wang & Wong, IEEE TCAD'92).
- **Floorplan of order 5.**
- **Floorplan tree:** A tree representing the hierarchy of partitioning.



The two possible wheels.



A floorplan of order 5



Corresponding floorplan tree

# Slicing Floorplan Design by Simulated Annealing

- Related works
  - Wong & Liu, “A new algorithm for floorplan design,” DAC’86.
  - Wong, Leong & Liu, *Simulated Annealing of VLSI Design*, pp. 31-51, Kluwer Academic Publishers, 1988.
- Ingredients: solution space, neighborhood structure, cost function, annealing schedule.



# Solution Representation

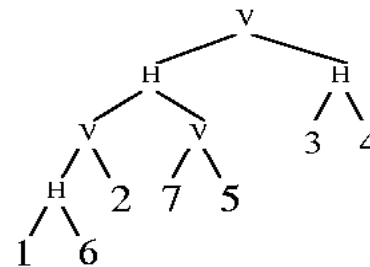
- An expression  $E = e_1 e_2 \dots e_{2n-1}$ , where  $e_i \in \{1, 2, \dots, n, H, V\}$ ,  $1 \leq i \leq 2n-1$ , is a **Polish expression** of length  $2n-1$  iff
  - every operand  $j$ ,  $1 \leq j \leq n$ , appears exactly once in  $E$ ;
  - (**balloting property**) for every sub-expression  $E_i = e_1 \dots e_i$ ,  $1 \leq i \leq 2n-1$ ,  $\#operands > \#operators$ .

1 6 H 3 5 V 2 H V 7 4 H V

# of operands = 4 ..... = 7  
 # of operators = 2 ..... = 5

- Polish expression  $\leftrightarrow$  Postorder traversal.
- $ijH$ :  $i$  below  $j$ ;  $ijV$ :  $i$  on the left of  $j$ .

7	5	4
6	2	
1		3



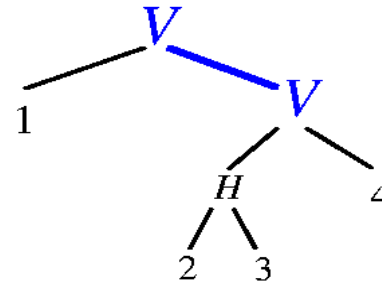
$E = 16H2V75VH34HV$

$E = 16 + 2 * 75 * + 34 + *$

*Postorder traversal of a tree!*

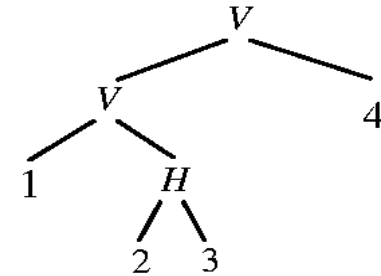
# Solution Representation (cont'd)

1	3	4
	2	



$E = 123H4VV$

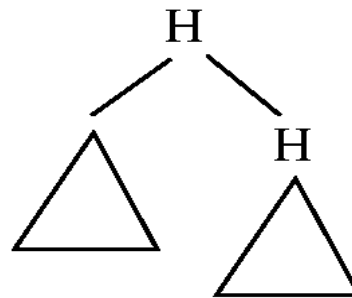
*non-skewed!*



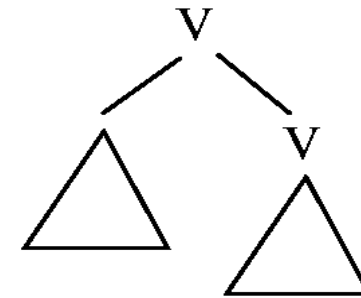
$E = 123HV4V$

*skewed!*

**Non-skewed cases**



..... HH .....

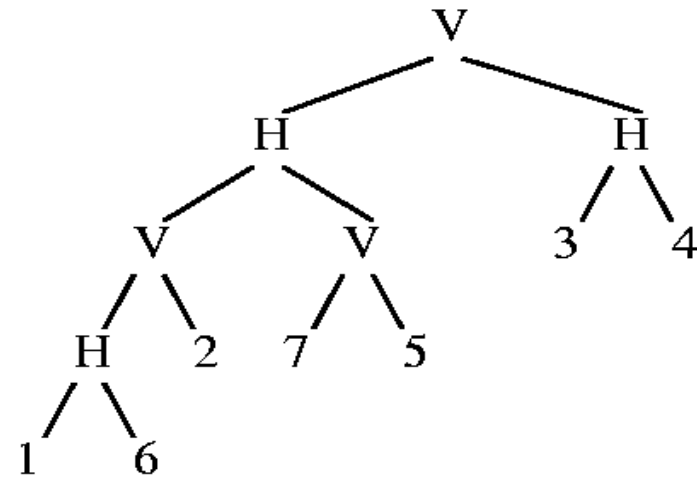
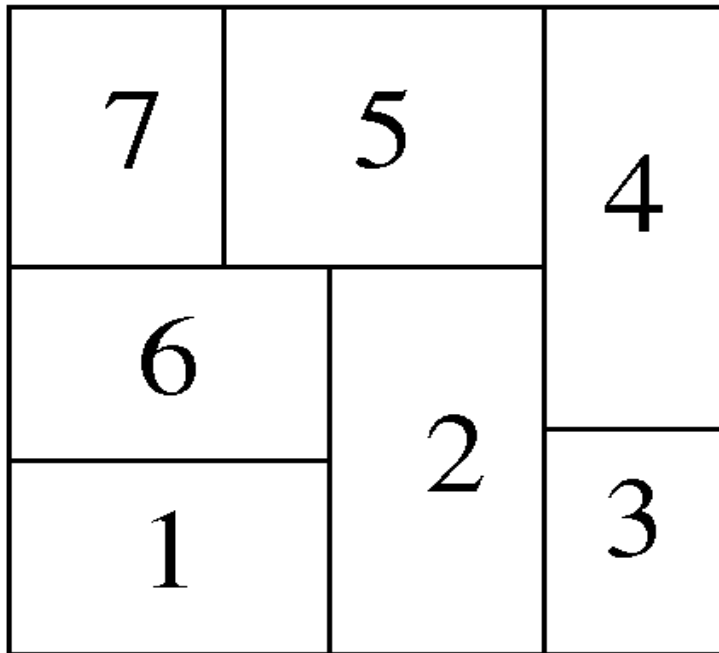


..... VV .....

- **Question:** how to eliminate redundant representations?

# Normalized Polish Expression

- A Polish expression  $E = e_1e_2\dots e_{2n-1}$  is called **normalized** iff  $E$  has no consecutive operators of the same type ( $H$  or  $V$ ).
- Given a **normalized** Polish expression, we can construct a **unique** rectangular slicing structure.

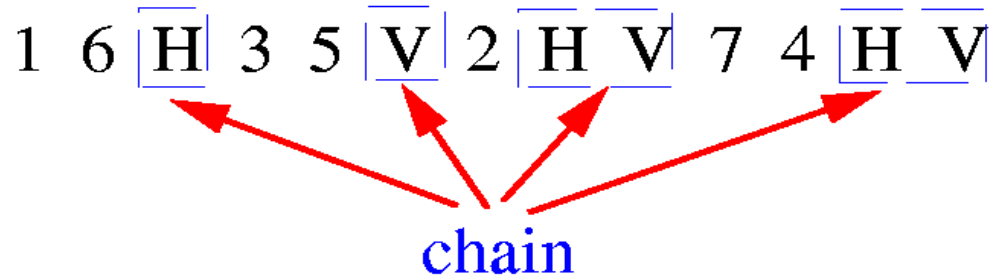


$E = 16H2V75VH34HV$

A normalized Polish expression

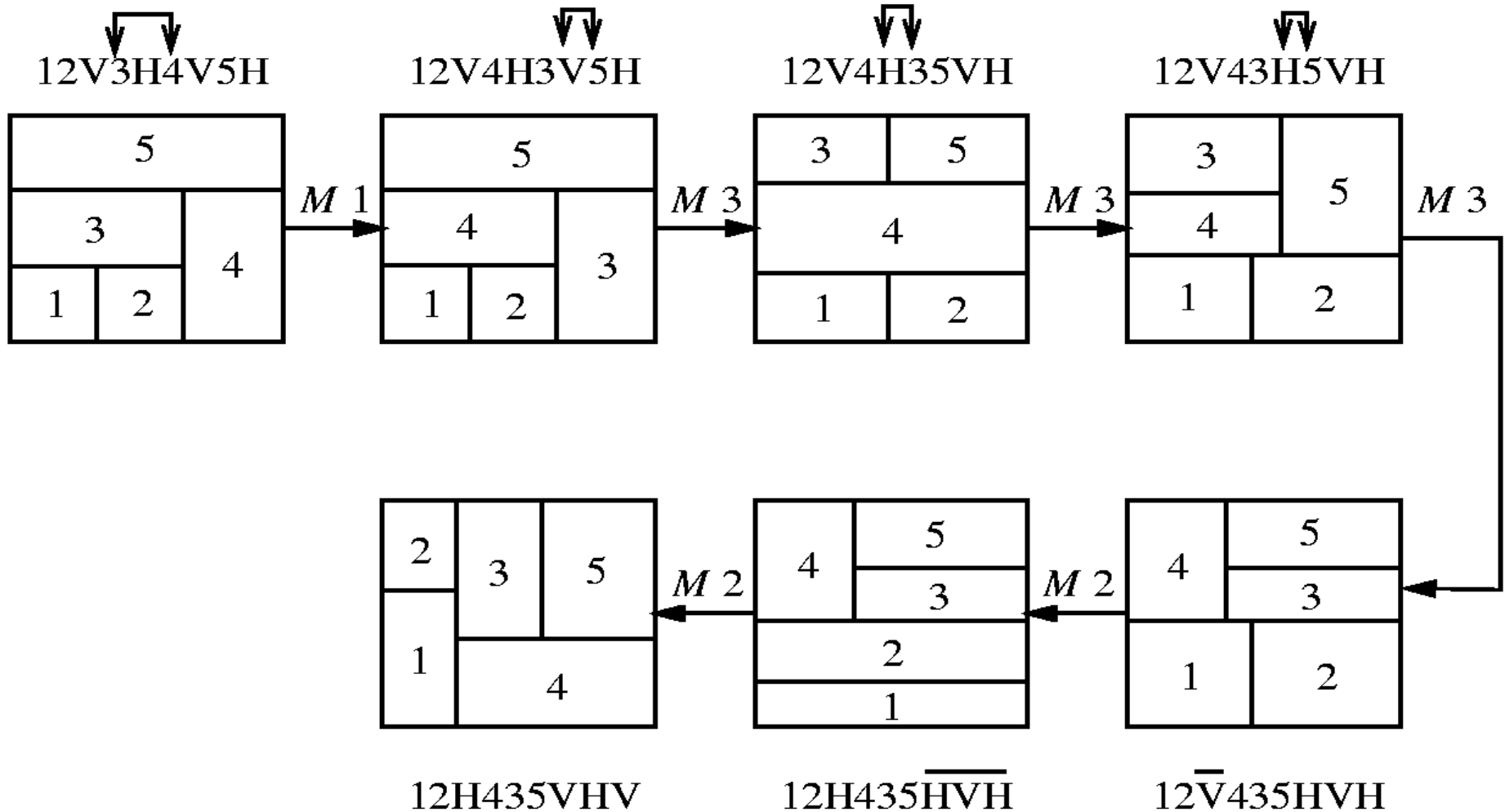
# Neighborhood Structure

- **Chain:**  $HVHVH...$  or  $VHVHV...$

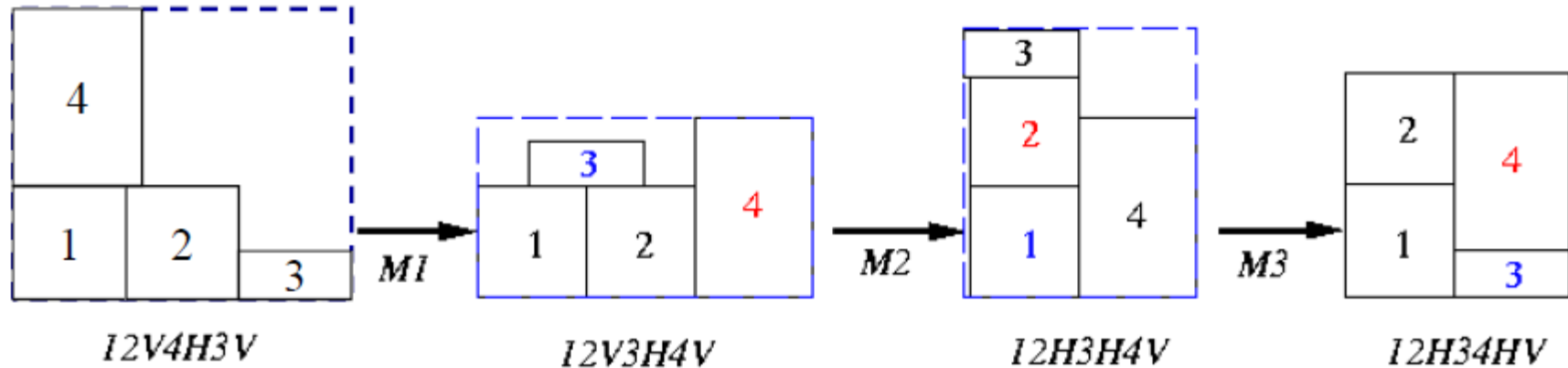


- **Adjacent:** 1 and 6 are adjacent operands; 2 and 7 are adjacent operands; 5 and  $V$  are adjacent operand and operator.
- 3 types of moves:
  - $M_1$  (**Operand Swap**): swap two adjacent operands.
  - $M_2$  (**Chain Invert**): Complement a chain. ( $\overline{V} = H, \overline{H} = V$ )
  - $M_3$  (**Operator/Operand Swap**): Swap two adjacent operand and operator.
- It can be proved that each normalized Polish expression can be obtained from any other one through a finite set of moves of the above three types.

# Solution Perturbation



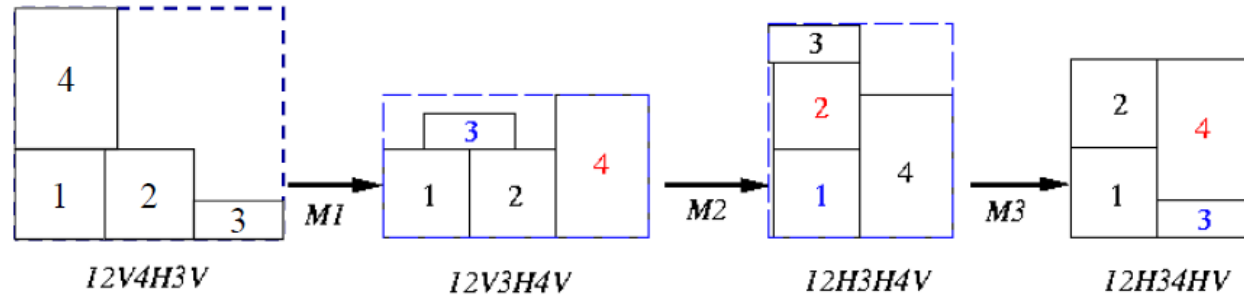
# Effects of Perturbation



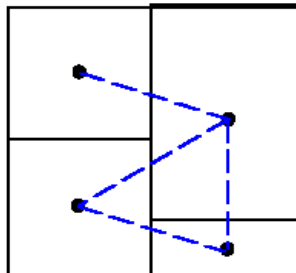
- **Question:** Does the balloting property hold during the moves?
  - $M_1$  and  $M_2$  moves are OK.
  - Check the  $M_3$  moves! Reject “illegal”  $M_3$  moves.
- **Check  $M_3$  moves:** Assume that the  $M_3$  move swaps the operand  $e_i$  with the operator  $e_{i+1}$ ,  $1 \leq i \leq 2n-2$ . Then, the swap will not violate the balloting property iff  $2N_{i+1} < i$ .
  - $N_k$ : # of operators in the Polish expression  $E = e_1 e_2 \dots e_k$ ,  $1 \leq k \leq 2n-1$ .

# Cost Function

- $\Phi = A + \lambda W$ 
  - $A$ : area of the smallest rectangle
  - $W$ : overall wiring length
  - $\lambda$ : user-specified parameter

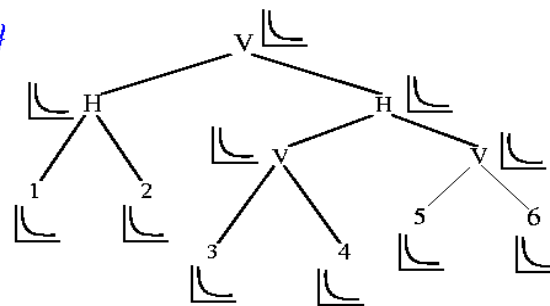
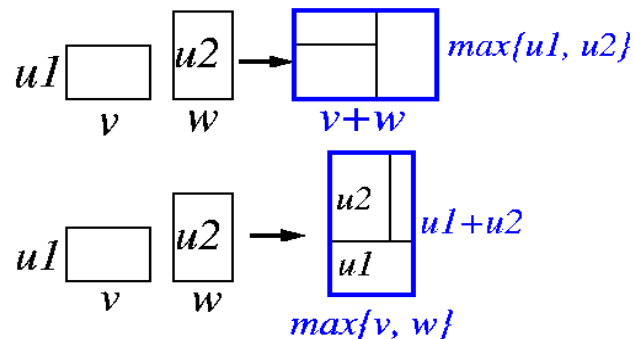
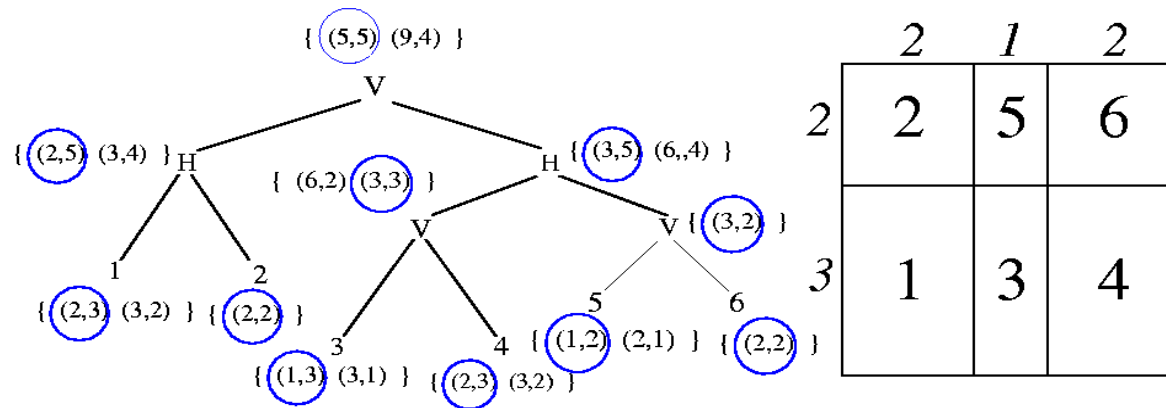


- $W = \sum_{ij} c_{ij} d_{ij}$ 
  - $c_{ij}$ : # of connections between blocks  $i$  and  $j$ .
  - $d_{ij}$ : center-to-center distance between basic rectangles  $i$  and  $j$ .



# Area Computation for Hard Blocks

- Stockmeyer, “Optimal orientations of cells in slicing floorplan designs,” *Information and Control*, 1983.
- Time complexity:  $O(knd)$ , where  $n$  is # modules,  $d$  is the depth of tree, and each modules has  $O(k)$  possible shapes.



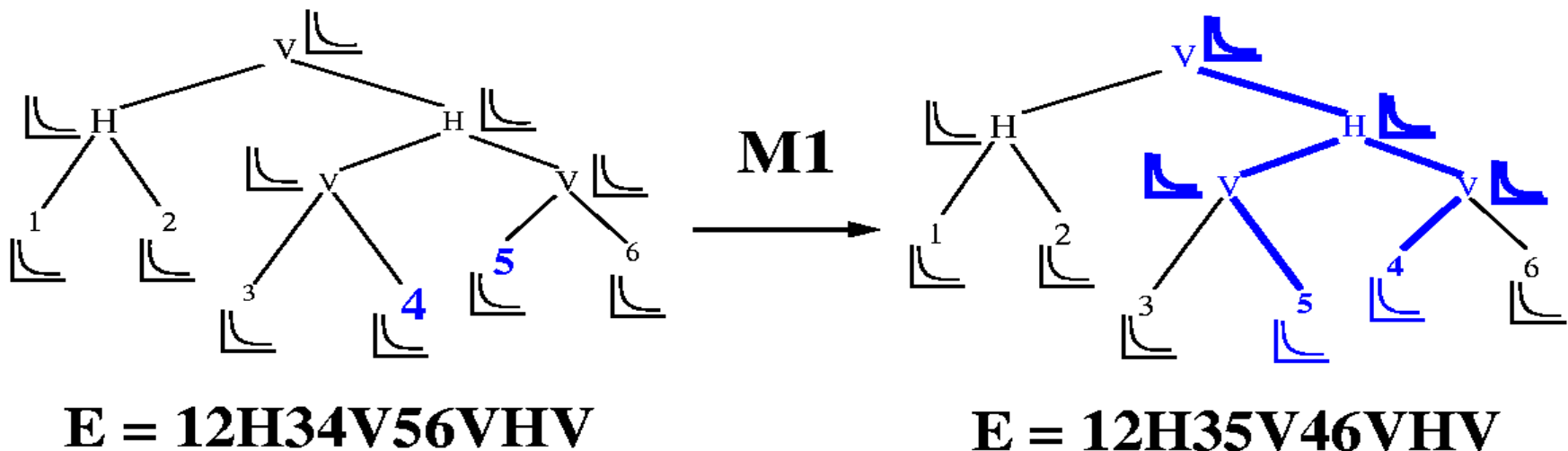


# Slicing Floorplan Sizing

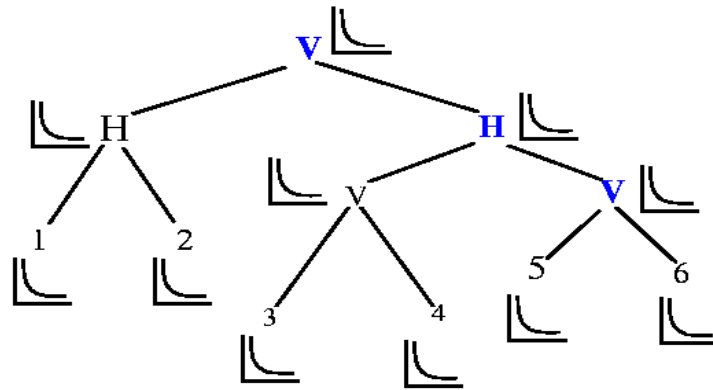
- The shape function of each leaf block is given as a staircase (or piecewise linear) function.
- Traverse the slicing tree to compute the shape functions of all composite blocks (bottom-up composition).
- Choose the desired shape of the top-level block
  - Only the corner points of the function need to be evaluated for area minimization.
- Propagate the consequences of the choice down to the leaf blocks (top-down propagation).

# Incremental Area Computation

- Each move leads to only a minor modification of the Polish expression.
- At most **two paths** of the slicing tree need to be updated for each move.

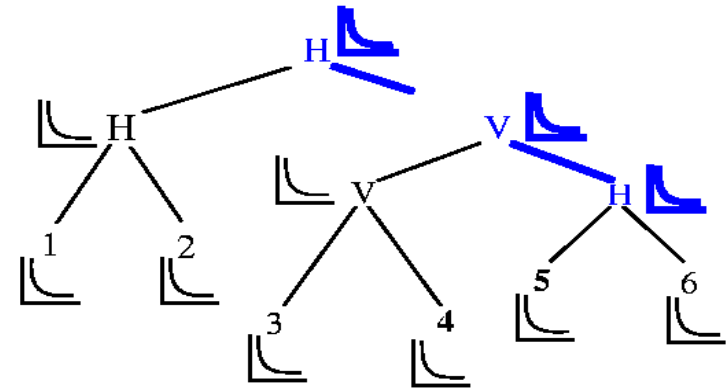


# Incremental Area Computation (cont'd)

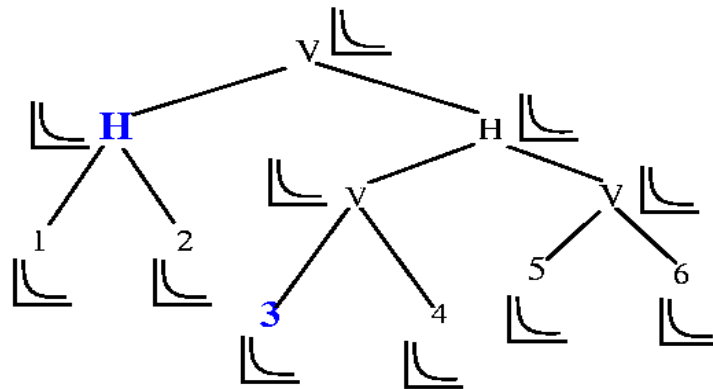


**E = 12H34V56VHV**

**M2**

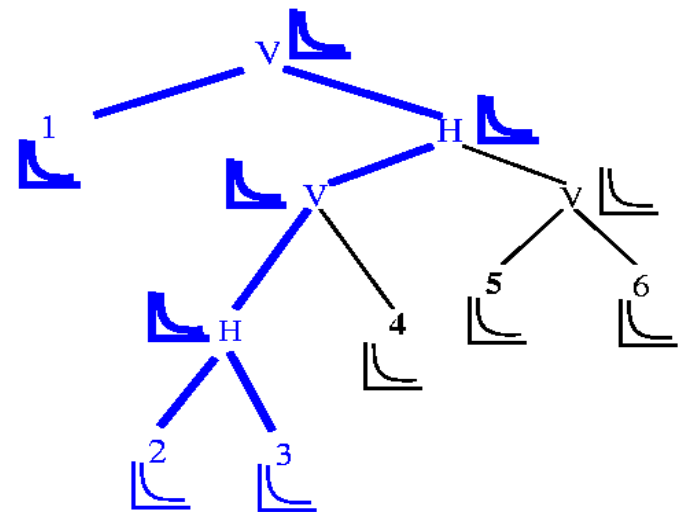


**E = 12H34V56HVVH**



**E = 12H34V56VHV**

**M3**



**E = 123H4V56VHV**

# Annealing Schedule

- Initial solution:  $12V3V...nV$

<b>1</b>	<b>2</b>	<b>3</b>		<b>n</b>
----------	----------	----------	--	----------

- $T_i = r^i T_0$ ,  $i = 1, 2, 3, \dots$ ;  $r = 0.85$
- At each temperature, try  $kn$  moves ( $k = 5-10$ )
- Terminate the annealing process if
  - # of accepted moves  $< 5\%$
  - Temperature is low enough, or
  - Run out of time.

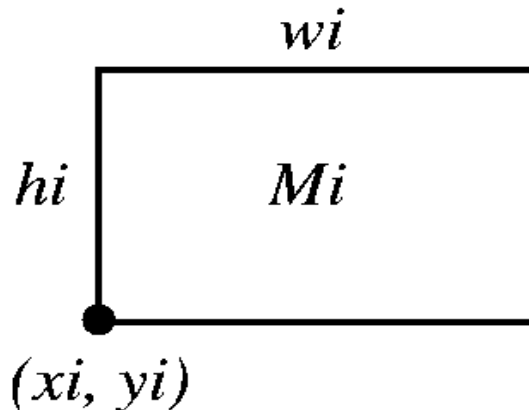
```

Algorithm: Simulated_Annealing_Floorplanning( $P, \epsilon, r, k$ )
1 begin
2  $E \leftarrow 12V3V4V \dots nV$ ; /* initial solution */
3  $Best \leftarrow E$ ;  $T_0 \leftarrow \frac{\Delta_{avg}}{\ln(P)}$ ;  $M \leftarrow MT \leftarrow uphill \leftarrow 0$ ;  $N = kn$ ;
4 repeat
5    $MT \leftarrow uphill \leftarrow reject \leftarrow 0$ ;
6   repeat
7     SelectMove( $M$ );
8     Case  $M$  of
9        $M_1$ : Select two adjacent operands  $e_i$  and  $e_j$ ;  $NE \leftarrow Swap(E, e_i, e_j)$ ;
10       $M_2$ : Select a nonzero length chain  $C$ ;  $NE \leftarrow Complement(E, C)$ ;
11       $M_3$ :  $done \leftarrow FALSE$ ;
12      while not ( $done$ ) do
13        Select two adjacent operand  $e_i$  and operator  $e_{i+1}$ ;
14        if ( $e_{i-1} \neq e_{i+1}$ ) and ( $2N_{i+1} < i$ ) then  $done \leftarrow TRUE$ ;
15         $NE \leftarrow Swap(E, e_i, e_{i+1})$ ;
16         $MT \leftarrow MT + 1$ ;  $\Delta cost \leftarrow cost(NE) - cost(E)$ ;
17        if ( $\Delta cost \leq 0$ ) or ( $Random < e^{\frac{-\Delta cost}{T}}$ )
18        then
19          if ( $\Delta cost > 0$ ) then  $uphill \leftarrow uphill + 1$ ;
20           $E \leftarrow NE$ ;
21          if  $cost(E) < cost(best)$  then  $best \leftarrow E$ ;
22        else  $reject \leftarrow reject + 1$ ;
23      until ( $uphill > N$ ) or ( $MT > 2N$ );
24     $T = rT$ ; /* reduce temperature */
25  until ( $\frac{reject}{MT} > 0.95$ ) or ( $T < \epsilon$ ) or OutOfTime;
26 end

```

# Non-slicing Floorplan Design by Mathematical Programming

- Sutanthavibul, Shragowitz, and Rosen, “An analytical approach to floorplan design and optimization,” DAC’90.
- Notation:
  - $w_i, h_i$ : width and height of module  $M_i$ .
  - $(x_i, y_i)$ : coordinate of the lower left corner of module  $M_i$ .
  - $a_i \leq w_i/h_i \leq b_i$ : aspect ratio  $w_i/h_i$  of module  $M_i$ . (Note: We defined aspect ratio as  $h_i/w_i$  before.)
- Goal: Find a **mixed integer linear programming (MILP)** formulation for the floorplan design.
  - **Linear** constraints? Objective function?



$$\begin{aligned} \text{Area} &= h_i * w_i \\ \text{Aspect ratio} &= w_i / h_i \end{aligned}$$

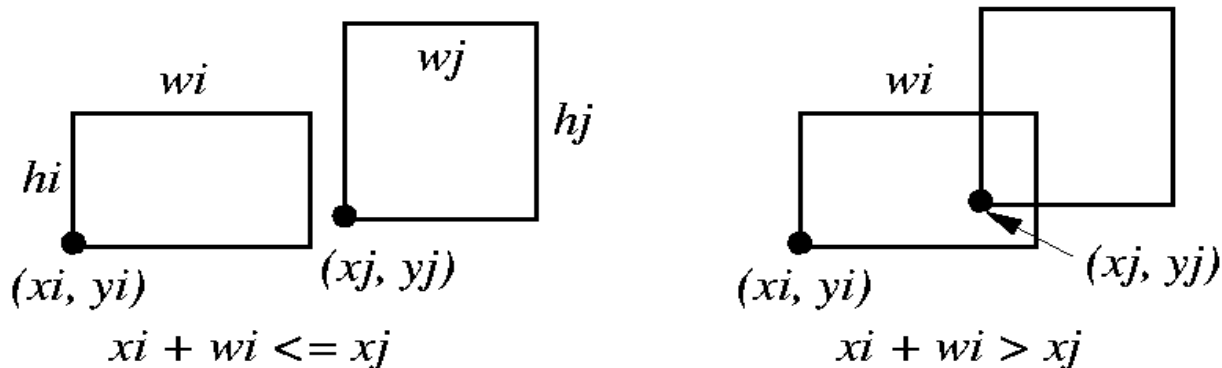
# Non-overlap Constraints

- Two modules  $M_i$  and  $M_j$  do not overlap, if at least one of the following linear constraints is satisfied (cases encoded by  $p_{ij}$  and  $q_{ij}$ ):

		$p_{ij}$	$q_{ij}$
$M_i$ to the left of $M_j$ :	$x_i + w_i \leq x_j$	0	0
$M_i$ below $M_j$ :	$y_i + h_i \leq y_j$	0	1
$M_i$ to the right of $M_j$ :	$x_i - w_j \geq x_j$	1	0
$M_i$ above $M_j$ :	$y_i - h_j \geq y_j$	1	1

- Let  $W, H$  be upper bounds on the floorplan width and height, respectively.
- Introduce two 0, 1 variables  $p_{ij}$  and  $q_{ij}$  to denote that one of the above inequalities is enforced; e. g.,  $p_{ij} = 0, q_{ij} = 1 \Rightarrow y_i + h_i \leq y_j$  is satisfied.

$$\begin{aligned}
 x_i + w_i &\leq x_j + W(p_{ij} + q_{ij}) \\
 y_i + h_i &\leq y_j + H(1 + p_{ij} - q_{ij}) \\
 x_i - w_j &\geq x_j - W(1 - p_{ij} + q_{ij}) \\
 y_i - h_j &\geq y_j - H(2 - p_{ij} - q_{ij})
 \end{aligned}$$



# Cost Function & Constraints

- Minimize Area =  $xy$ , nonlinear! ( $x, y$ : width and height of the resulting floorplan)
- How to fix?
  - Fix the width  $W$  and minimize the height  $y$ !
- Four types of constraints:
  1. No two modules overlap ( $\forall i, j: 1 \leq i < j \leq n$ );
  2. Each module is enclosed within a rectangle of width  $W$  and height  $H$  ( $x_i + w_i \leq W, y_i + h_i \leq H, 1 \leq i \leq n$ );
  3.  $x_i \geq 0, y_i \geq 0, 1 \leq i \leq n$ ;
  4.  $p_{ij}, q_{ij} \in \{0, 1\}$ .
- $w_i, h_i$  are known.



# Mixed ILP for Floorplanning

Mixed ILP for the floorplanning problem with rigid, fixed modules.

$$\begin{array}{ll}
 \min & y \\
 \text{subject to} & \\
 & x_i + w_i \leq W, \quad 1 \leq i \leq n \quad (1) \\
 & y_i + h_i \leq y, \quad 1 \leq i \leq n \quad (2) \\
 & x_i + w_i \leq x_j + W(p_{ij} + q_{ij}), \quad 1 \leq i < j \leq n \quad (3) \\
 & y_i + h_i \leq y_j + H(1 + p_{ij} - q_{ij}), \quad 1 \leq i < j \leq n \quad (4) \\
 & x_i - w_j \geq x_j - W(1 - p_{ij} + q_{ij}), \quad 1 \leq i < j \leq n \quad (5) \\
 & y_i - h_j \geq y_j - H(2 - p_{ij} - q_{ij}), \quad 1 \leq i < j \leq n \quad (6) \\
 & x_i, y_i \geq 0, \quad 1 \leq i \leq n \quad (7) \\
 & p_{ij}, q_{ij} \in \{0, 1\}, \quad 1 \leq i < j \leq n \quad (8)
 \end{array}$$

- Size of the mixed ILP: for  $n$  modules,
  - # continuous variables:  $O(n)$ ; # integer variable:  $O(n^2)$ ; # linear constraints:  $O(n^2)$ .
  - Unacceptably huge program for a large  $n$ !
- Popular LP software: LINDO, lp\_solve, CPLEX, GUROBI, etc.

# Mixed ILP for Floorplanning (cont'd)

$$\begin{array}{ll}
 \min & y \\
 \text{subject to} & \\
 & x_i + r_i h_i + (1 - r_i) w_i \leq W, \quad 1 \leq i \leq n \quad (9) \\
 & y_i + r_i w_i + (1 - r_i) h_i \leq y, \quad 1 \leq i \leq n \quad (10) \\
 & x_i + r_i h_i + (1 - r_i) w_i \leq x_j + M(p_{ij} + q_{ij}), \quad 1 \leq i < j \leq n \quad (11) \\
 & y_i + r_i w_i + (1 - r_i) h_i \leq y_j + M(1 + p_{ij} - q_{ij}), \quad 1 \leq i < j \leq n \quad (12) \\
 & x_i - r_j h_j - (1 - r_j) w_j \geq x_j - M(1 - p_{ij} + q_{ij}), \quad 1 \leq i < j \leq n \quad (13) \\
 & y_i - r_j w_j - (1 - r_j) h_j \geq y_j - M(2 - p_{ij} - q_{ij}), \quad 1 \leq i < j \leq n \quad (14) \\
 & x_i, y_i \geq 0, \quad 1 \leq i \leq n \quad (15) \\
 & p_{ij}, q_{ij} \in \{0, 1\}, \quad 1 \leq i < j \leq n \quad (16)
 \end{array}$$

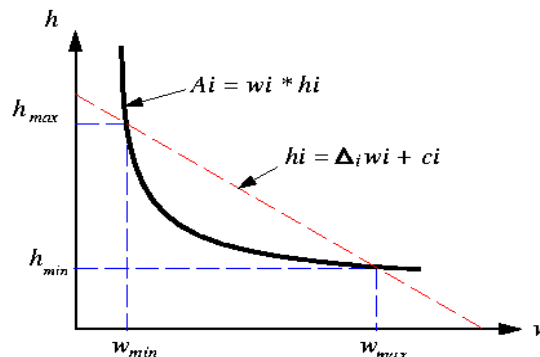
- For each module  $i$  with free orientation, associate a 0-1 variable  $r_i$ ;
- $r_i = 0$ :  $0^\circ$  rotation for module  $i$ .
- $r_i = 1$ :  $90^\circ$  rotation for module  $i$ .
- $M = \max\{W, H\}$ .

# Soft Modules

- Assumptions:  $w_i, h_i$  are unknown; area lower bound:  $A_i$ .
- Module size constraints:  $w_i h_i \geq A_i$ ;  $a_i \leq \frac{w_i}{h_i} \leq b_i$ .
- Hence,  $w_{min} = \sqrt{A_i a_i}$ ,  $w_{max} = \sqrt{A_i b_i}$ ,  $h_{min} = \sqrt{\frac{A_i}{b_i}}$ ,  $h_{max} = \sqrt{\frac{A_i}{a_i}}$ .
- $w_i h_i \geq A_i$  nonlinear! How to fix? (The following fixing scheme is different from the one given in the paper.)
  - Can apply a first-order approximation of the equation: a line passing through  $(w_{min}, h_{max})$  and  $(w_{max}, h_{min})$ .
 
$$h_i = \Delta_i w_i + c_i \quad /* y = mx + c */$$

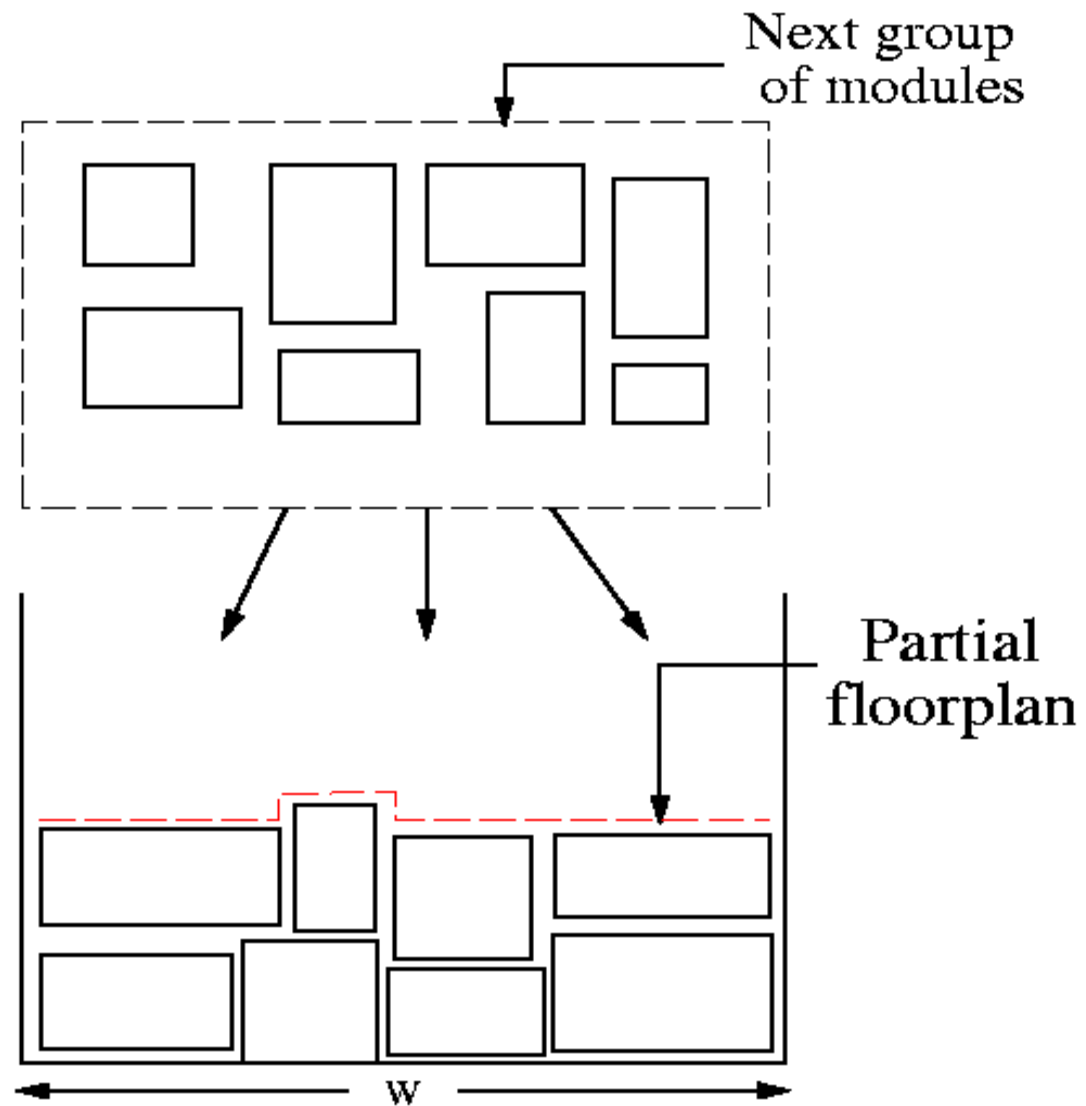
$$\Delta_i = \frac{h_{max} - h_{min}}{w_{min} - w_{max}} \quad /* slope */$$

$$c_i = h_{max} - \Delta_i w_{min} \quad /* c = y_0 - mx_0 */$$
  - Substitute  $\Delta_i w_i + c_i$  for  $h_i$  to form linear constraints ( $x_i, y_i, w_i$  are unknown;  $\Delta_i, c_i, h_i$  can be computed as above).



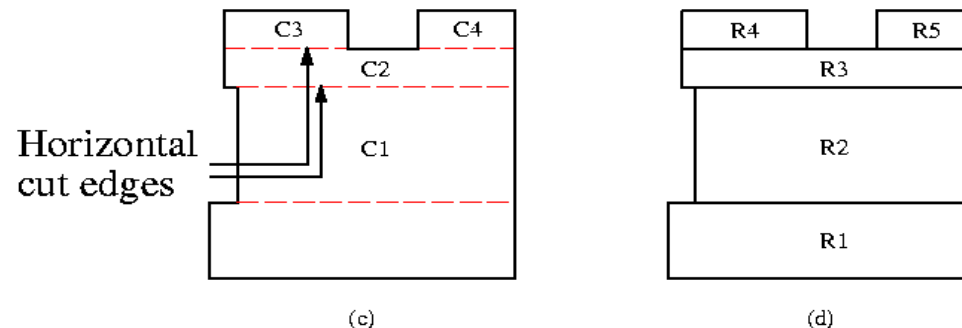
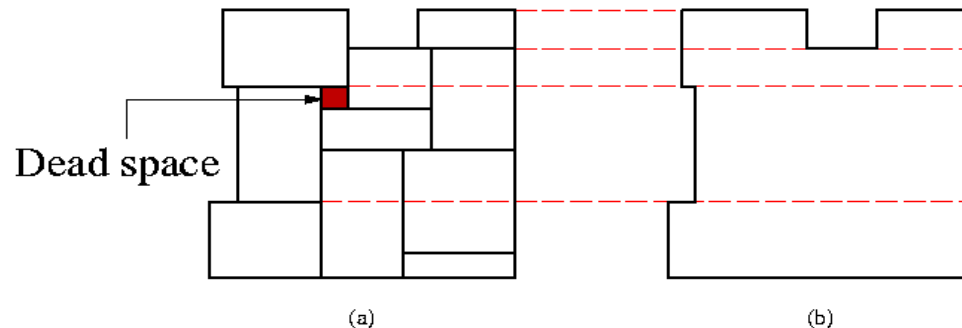
# Reducing the Size of the Mixed ILP

- Time complexity of a mixed ILP: exponential!
- Recall the large size of the mixed ILP: # variables, # constraints:  $O(n^2)$ .
  - How to fix it?
- Key: solve a partial problem at each step (successive augmentation)
- Questions:
  - How to select next subgroup of modules?  $\Rightarrow$  linear ordering based on connectivity.
  - How to minimize # of required variables?



# Reducing the Size of the Mixed ILP (cont'd)

- Size of each successive mixed ILP depends on (1) # of modules in the next group; (2) “size” of the partially constructed floorplan.
- Keys to deal with (2):
  - Minimize the problem size of the partial floorplan.
  - Replace the already placed modules by a set of covering rectangles.
  - # rectangles is usually much smaller than # placed modules.

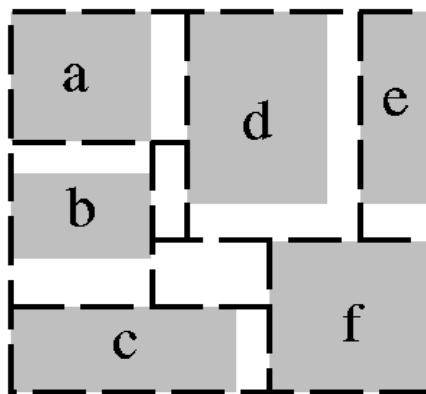
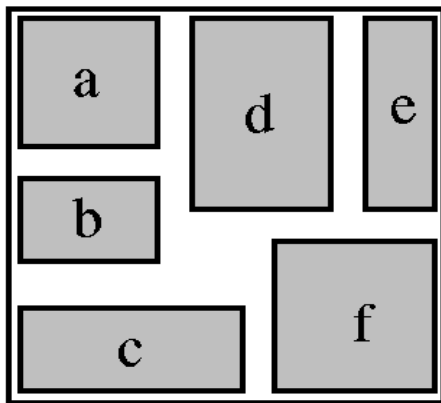


# P-admissible Solution Space

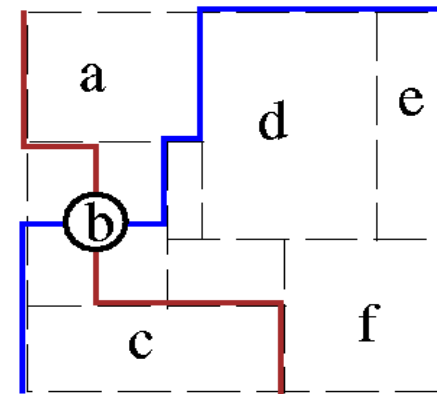
- **P-admissible** solution space for Problem  $P$ :
  1. the solution space is finite,
  2. every solution is feasible,
  3. evaluation for each configuration is possible in polynomial time and so is the implementation of the corresponding configuration, **and**
  4. the configuration corresponding to the best evaluated solution in the space coincides with an optimal solution of  $P$ .
- Slicing floorplan is **not** P-admissible. Why?
- A P-admissible floorplan representation:  
**Sequence-Pair.**

# Sequence Pair (SP)

- Murata, Fujiyoshi, Nakatake and Kajitani, “Rectangle-packing-based module placement,” ICCAD’95.
- Represent a *packing* by a pair of module permutations called sequence-pair (e.g.,  $(abdecf, cbfade)$ ).
- The set of all sequence-pairs is a P-admissible solution space whose size is  $(n!)^2$ .
- Search in the P-admissible solution space by simulated annealing.
  - Swap two modules in the first sequence.
  - Swap two modules in both sequences.
  - Rotate a module.



A floorplan

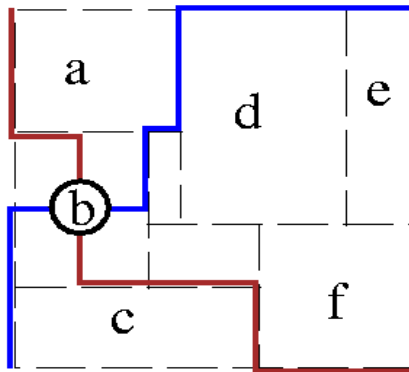
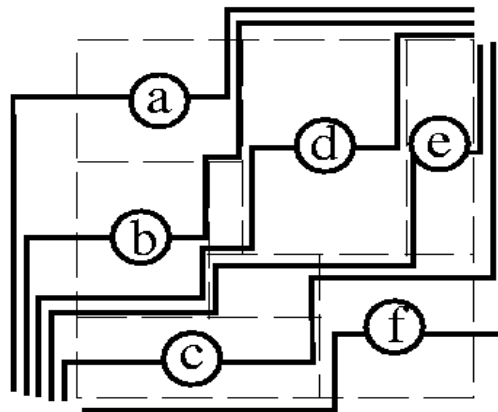


Loci of module b

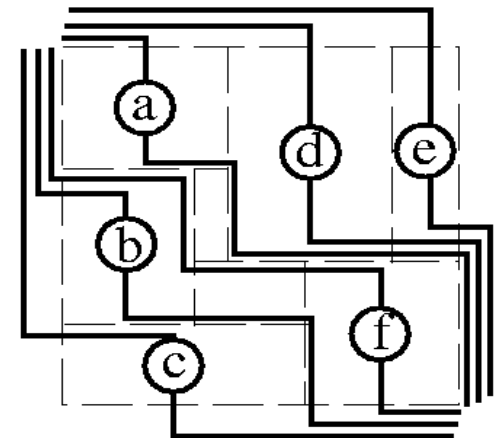


# Relative Module Positions

- A floorplan is a partition of a chip into rooms, each containing at most one module.
- **Locus** (right-up, left-down, up-left, down-right)
  1. Take a non-empty room.
  2. Start at the center of the room, walk in two alternating directions to hit the sides of rooms.
  3. Continue until to reach a corner of the chip.
- **Positive locus:** Union of right-up locus and left-down locus.
- **Negative locus:** union of up-left locus and down-right locus.

Unit 4 *Loci of module  $b$* 

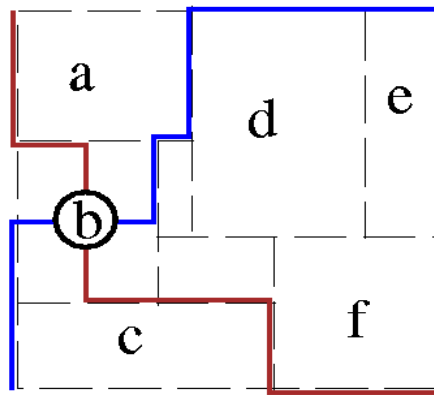
*Positive loci: abdecf*



*Negative loci: cbfade* 33

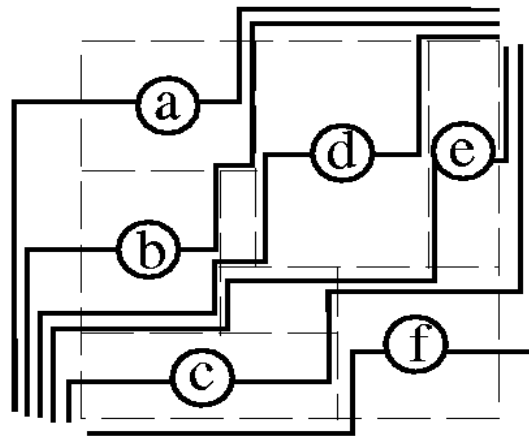
# Geometrical Information

- No pair of positive (negative) loci cross each other, i.e., **loci are linearly ordered**.
- **Sequence-Pair**  $(\Gamma_+, \Gamma_-)$ :  $\Gamma_+$  ( $\Gamma_-$ ) is the module permutation representing the order of positive (negative) loci.  
E.g.,  $(\Gamma_+, \Gamma_-) = (abdecf, cbfade)$ .
- $x'$  is **after** (**before**)  $x$  in both  $\Gamma_+$  and  $\Gamma_- \Rightarrow x'$  is **right** (**left**) to  $x$ .
- $x'$  is **after** (**before**)  $x$  in  $\Gamma_+$  and before (**after**)  $x$  in  $\Gamma_- \Rightarrow x'$  is **below** (**above**)  $x$ .

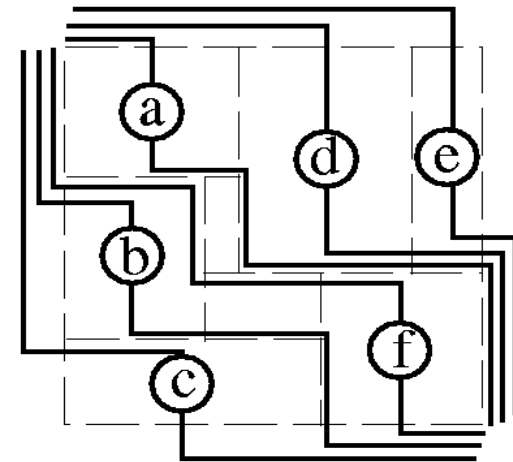


### *Loci of module $b$*

## Unit 4



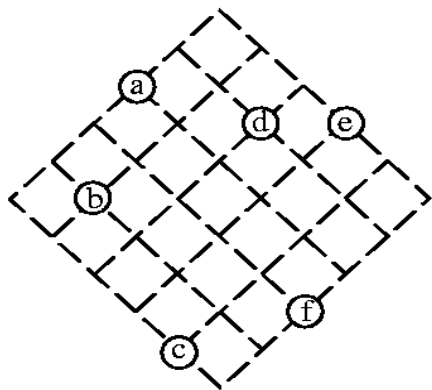
*Positive loci: abdecf*



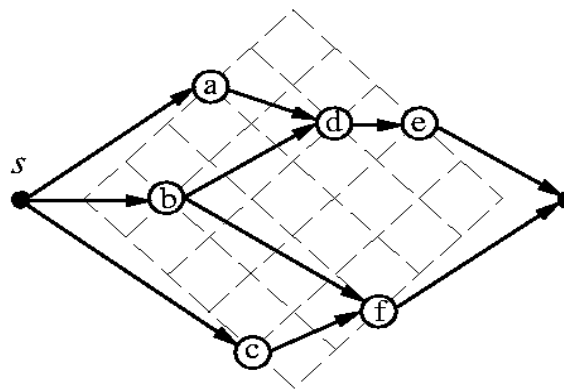
*Negative loci: cbfade*

# Optimal $(\Gamma_+, \Gamma_-)$ -Packing

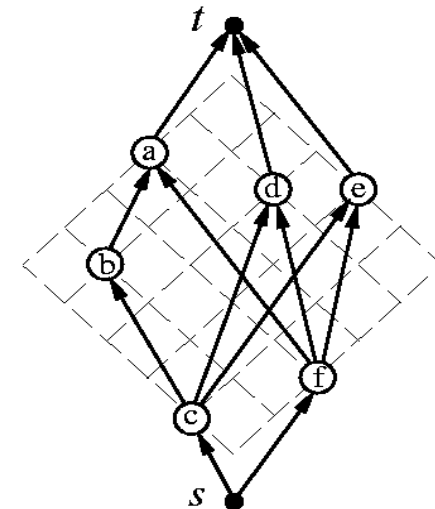
- For every sequence-pair  $(\Gamma_+, \Gamma_-)$ , there is an optimal  $(\Gamma_+, \Gamma_-)$ -packing.
- **Horizontal constraint graph**  $G_H(V, E)$  (similarly for  $G_V(V, E)$ ):
  - $V$ : source  $s$ , sink  $t$ ,  $n$  vertices for modules.
  - $E$ :  $(s, x)$  and  $(x, t)$  for each module  $x$ , and  $(x, x')$  iff  $x$  must be left-to  $x'$ .
  - **Vertex weight**: 0 for  $s$  and  $t$ , **width** of module  $x$  for the other vertices.



Packing for sequence pair:  
Unit 4 (abdecf, cbfade)



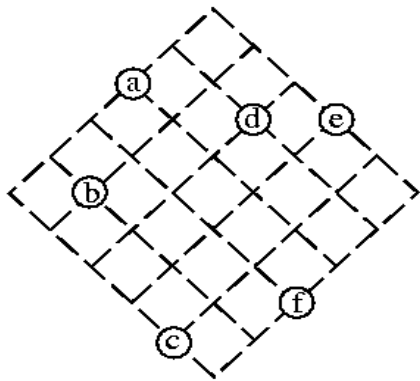
Horizontal constraint graph  
(Transitive edges are not shown)



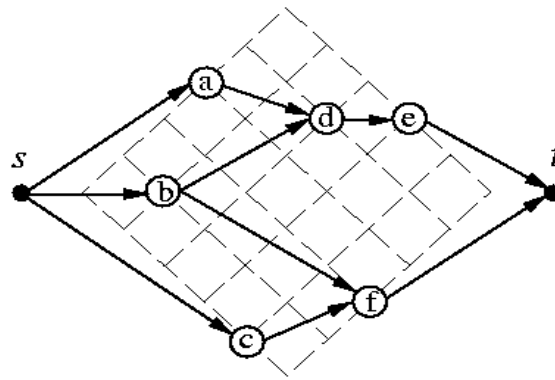
Vertical constraint graph  
(Transitive edges are not shown)

# Optimal $(\Gamma_+, \Gamma_-)$ -Packing (cont'd)

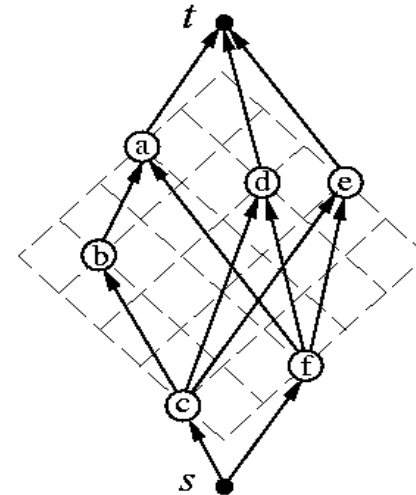
- **Optimal  $(\Gamma_+, \Gamma_-)$ -packing** can be obtained in  $O(n^2)$  time by applying a longest path algorithm on a vertex-weighted directed acyclic graph.
  - $G_H$  and  $G_V$  are independent.
  - The  $x$  and  $y$  coordinates of each module are determined by assigning the longest path length between  $s$  and the vertex of the module in  $G_H$  and  $G_V$ , respectively.
- More efficient algorithms for obtaining optimal  $(\Gamma_+, \Gamma_-)$ -packing:  $O(n \log n)$  by Takahashi, IEICE'96;  $O(n \log n)$  by Tang, Tian & Wong, DATE'00;  $O(n \log \log n)$  by Tang & Wong, ASP-DAC'01.



Packing for sequence pair:  
Unit 4 (abdecf, cbfade)



Horizontal constraint graph  
(Transitive edges are not shown)



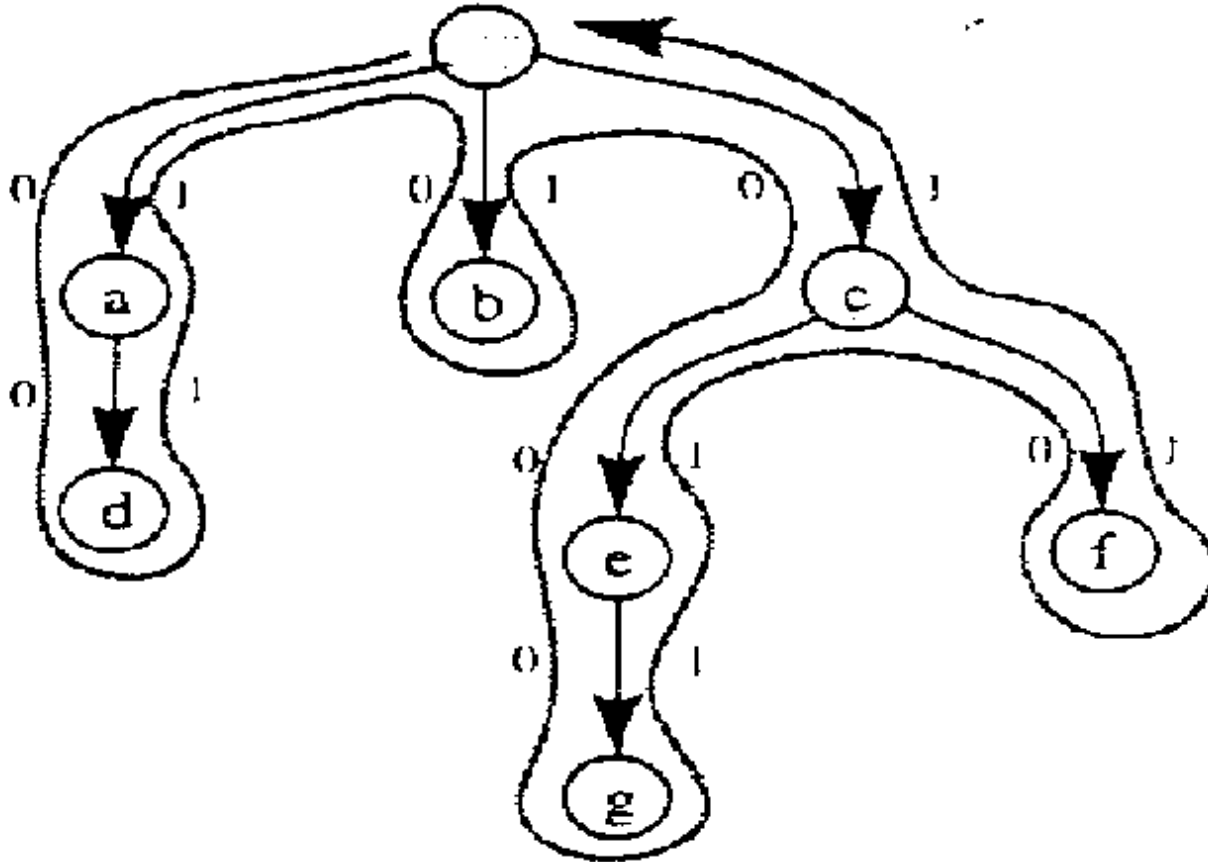
Vertical constraint graph  
(Transitive edges are not shown)

# O-Tree

- Guo, Cheng, and Yoshimura, “An o-tree representation of non-slicing floorplan and its applications,” DAC’99.
- Definitions:
  - A placement is **L-compact** (**B-compact**) if and only if no module can be moved left (down) from its original position with other modules’ positions fixed.
  - A placement is **LB-compact** (or **admissible**) if and only if it is both L-compact and B-compact.
- Given any placement  $P_1$ , a corresponding admissible placement  $P_2$  can be obtained by a sequence of  $x$ -direction and  $y$ -direction compactions. The overall area of  $P_2$  is no larger than the overall area of  $P_1$ .

# O-Tree Encoding

(00110100011011, *adbcegf*) with Depth-First-Search



# O-Tree Encoding (cont'd)

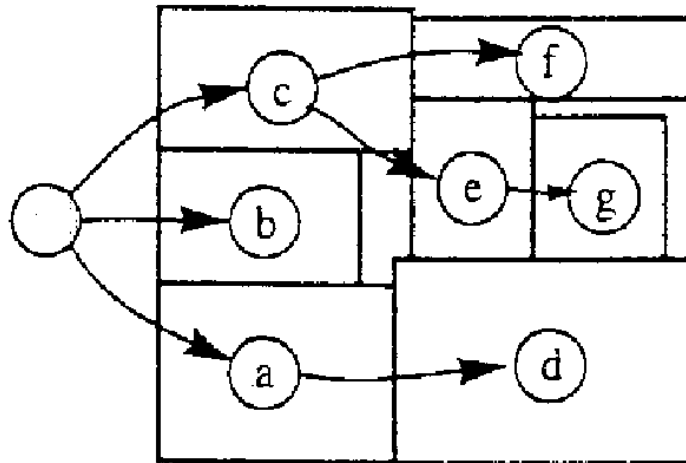
- **Space needed to store  $(T, \pi)$ :** Given a tree with  $n$  nodes in addition to its root, the label of each node can be encoded into a  $\lceil \lg n \rceil$  bit string, and hence  $n(2 + \lceil \lg n \rceil)$  bits are needed to store  $(T, \pi)$  where  $2n$  bits for  $T$ , and  $n\lceil \lg n \rceil$  bits for  $\pi$ .
- **Number of possible  $(T, \pi)$ 's:**  $O(n! 2^{2n-2} / n^{1.5})$

# O-Tree and Placement

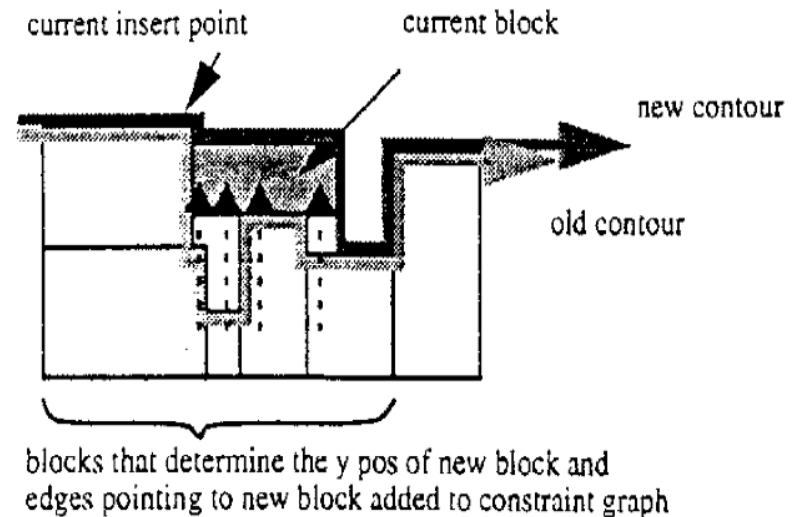
- **Horizontal O-Tree:**

- Suppose  $i$  is the parent of  $j$ ,  $\Psi(j)$  is the set of blocks each of which appears before  $j$  in  $\pi$  and overlaps with  $j$  in the  $x$ -coordinate projections.

(00110100011011, *adbcegf*)



$$\begin{cases} x_j = x_i + w_i \\ y_j = 0 \mid (\max_{k \in \Psi(j)} y_k + h_k) \end{cases}$$

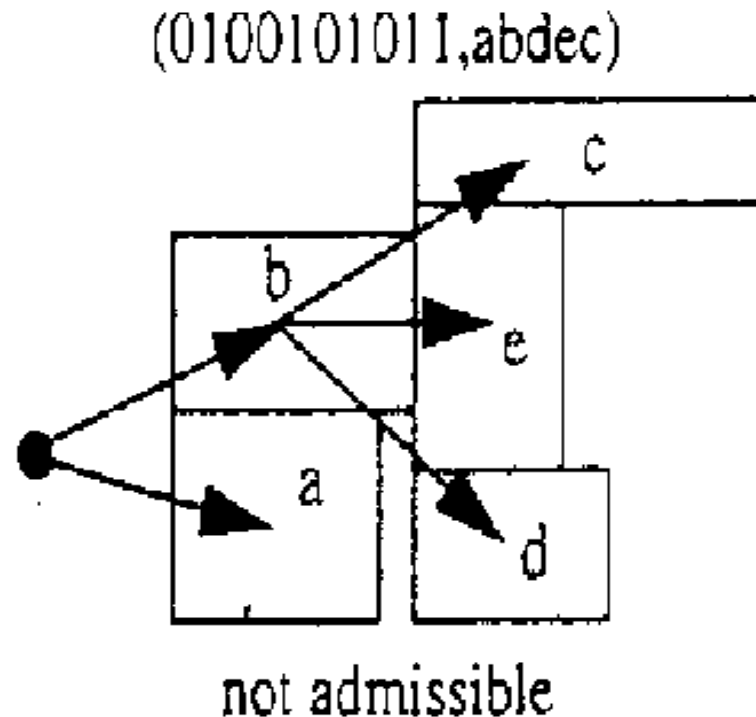
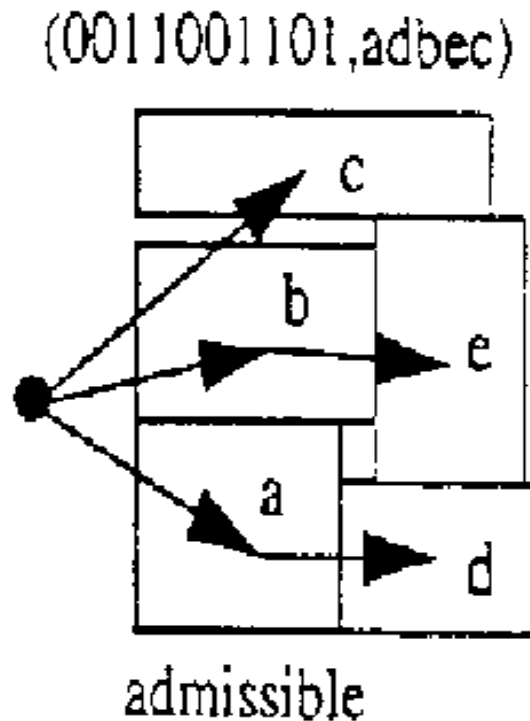


- **Vertical O-Tree:** can be defined similarly



# O-Tree and Placement (cont'd)

An O-tree is **admissible** if its corresponding placement is admissible.



# Admissible O-Tree Transformation (AOT)

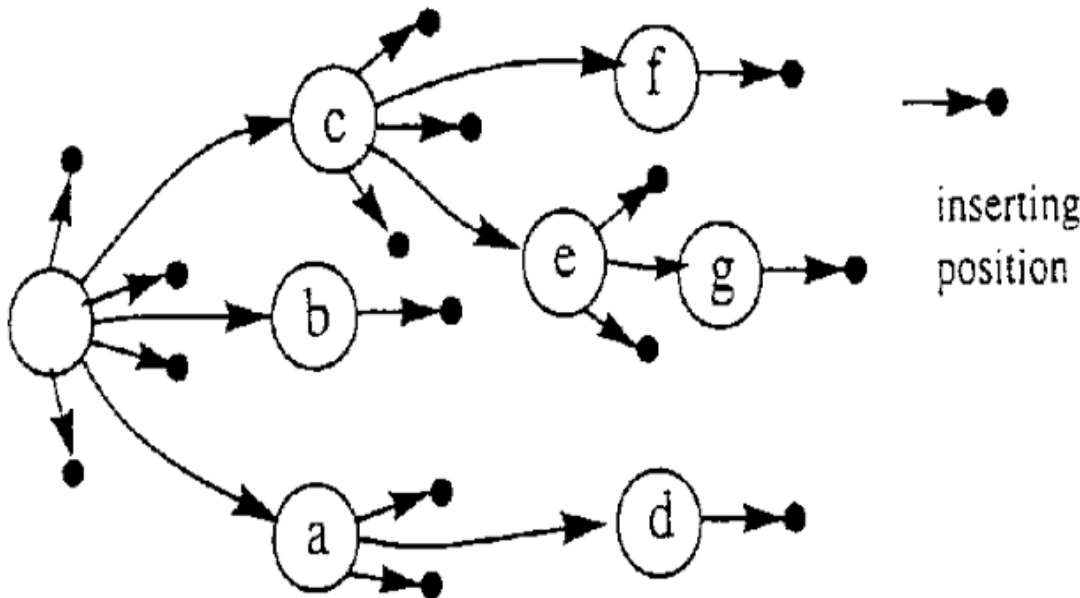
- Given a horizontal O-tree  $T$ , we can first get a vertical constraint graph  $G_y$  by applying “OT2OCG” to  $T$  in linear time, and then get a vertical O-tree  $T_y$  by applying “CG2OT” to  $G_y$  in linear time. After applying the same procedures OT2OCG and CG2OT again, we can get another horizontal O-tree. The OT2OCG and CG2OT are iterated until an admissible O-tree is found.

$$\text{H-O-tree} \xrightarrow{\text{OT2OCG}} G_v(\text{B-compact}) \xrightarrow{\text{CG2OT}} \text{V-O-tree} \xrightarrow{\text{OT2OCG}} G_h(\text{L-compact}) \xrightarrow{\text{CG2OT}} \text{H-O-tree} \dots$$

- All compactions are **monotone** because modules are either moved down or left. Therefore, convergence of the above iteration is assured and we can get an admissible O-tree.

# Solution Perturbation

- Select a module  $B_i$  in the O-tree  $(T, \pi)$ .
- Delete  $B_i$  from the O-tree  $(T, \pi)$ .
- Insert  $B_i$  in the position with the best cost value among all possible inserting positions in  $(T, \pi)$  as an external node.
- Perform a-c on it orthogonal O-tree.



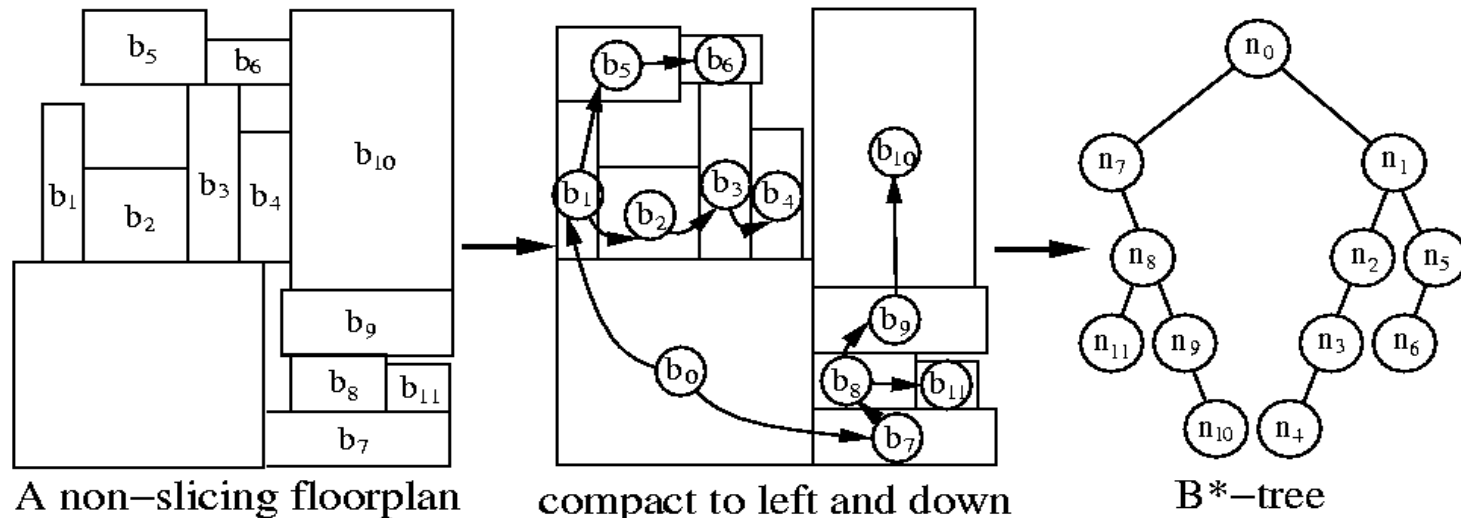
• Given any O-tree with  $n$  nodes, the number of possible inserting position as external nodes is  $2n-1$ .

# A Deterministic Placement Algorithm

- Perturb O-trees in sequence.
  - Select nodes in sequence and find the best perturb position for each of them.
  - A perturbed O-tree can be made admissible using AOT.
- Implementation is straightforward.

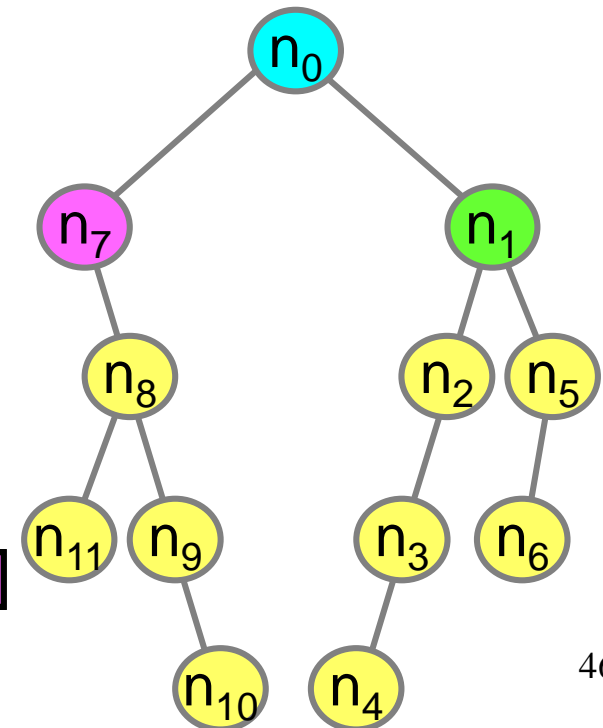
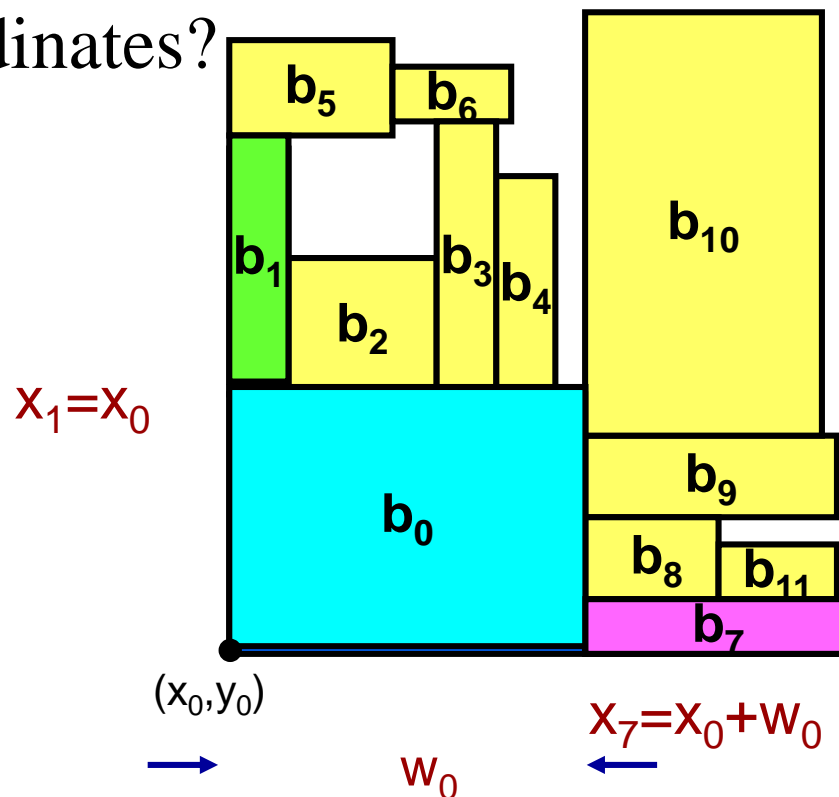
# B\*-Tree

- Chang, Chang, Wu, and Wu, “B\*-tree: a new representation for non-slicing floorplans,” DAC’00.
- Ideas:
  - From an admissible placement to a B\*-tree:
    - Left child: the lowest module on the right.
    - Right child: the module above, with the same left-side coordinate.



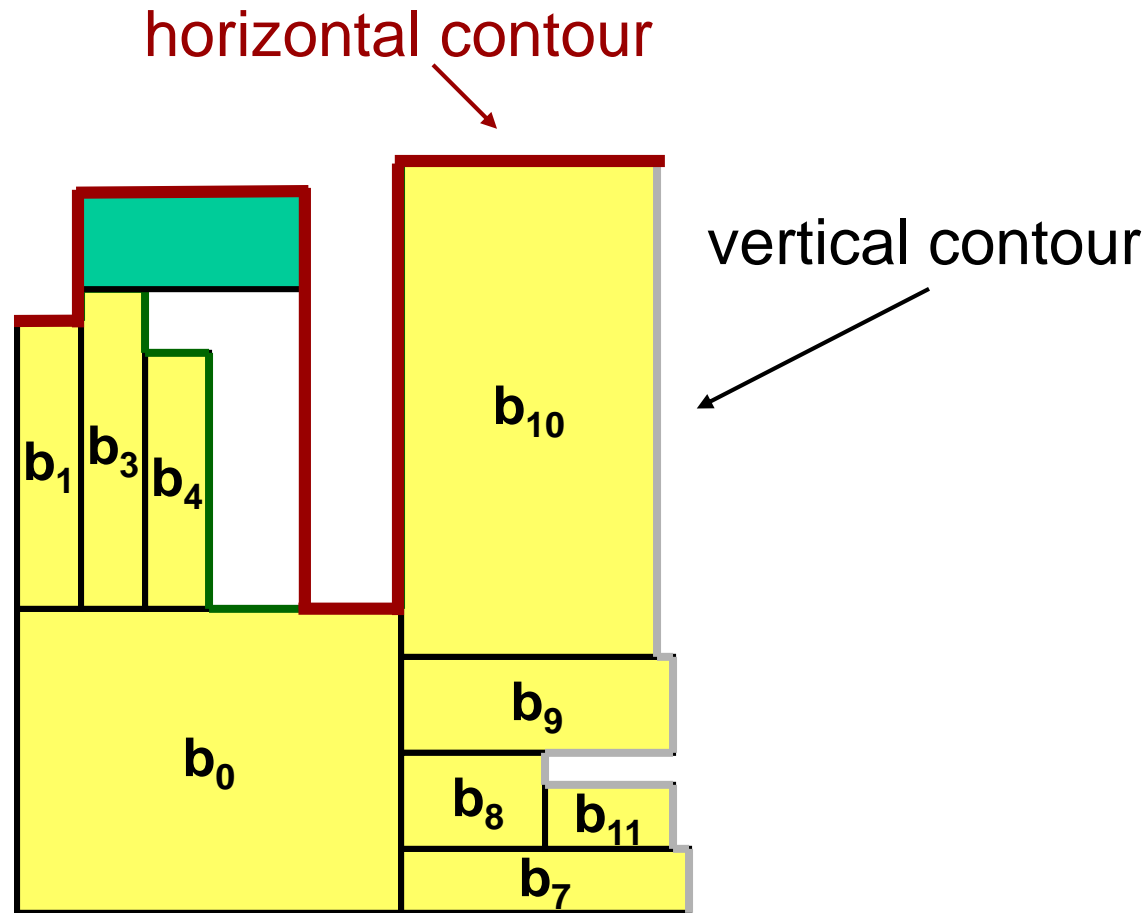
# B\*-tree Packing

- x-coordinates can 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?



# Computing y-coordinates

- Reduce the complexity of computing a y-coordinate to amortized  $O(1)$  time.



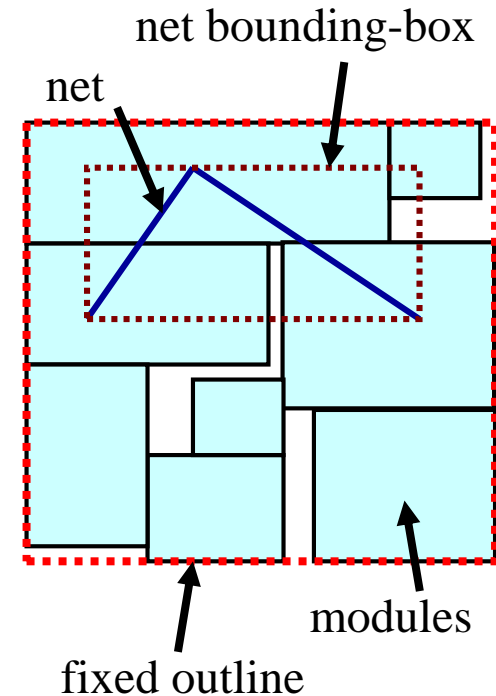
# Perturbations & Solutions

- Perturbing B\*-trees in simulated annealing
  - Op1: Rotate a module.
  - Op2: Flip a module.
  - Op3: Move a module to another place.
  - Op4: Swap two modules



# Fixed-Outline Floorplanning

- Chen and Chang, “Modern floorplanning based on fast simulated annealing,” ISPD’05 & TCAD’06
- Input: modules, netlist, fixed outline
- Output: module positions, orientations
- Objectives
  - Minimize the half-perimeter wirelength (HPWL)
  - All modules are within the fixed die (fixed-outline constraint) and no overlaps occur between modules



# Fixed-Outline Constraint

- Given the total area  $A$  of modules, the percentage  $\Gamma$  of dead space, and the desired aspect ratio  $R^*$ , the outline is defined by

$$H^* = \sqrt{(1+\Gamma)AR^*} \quad W^* = \sqrt{(1+\Gamma)A/R^*}$$

$$- R^* = H^*/W^*, H^*W^* = (1+\Gamma)A$$

- Cost for floorplan  $F$

$$\Phi(F) = \alpha A + \beta L + (1 - \alpha - \beta)(R^* - R)^2$$

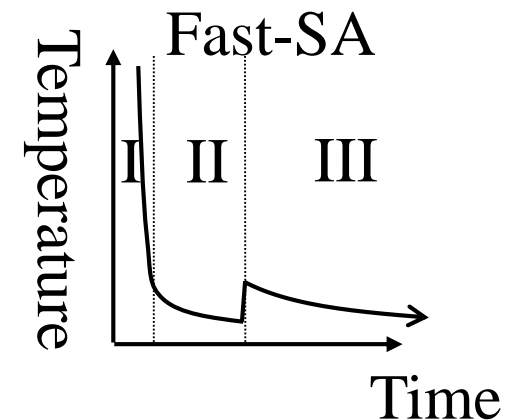
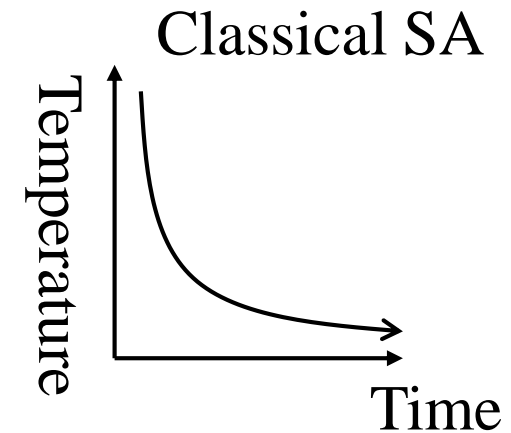
- $A$ : floorplan area
- $L$ : wirelength
- $R^*$ : fixed-outline aspect ratio
- $R$ : floorplan aspect ratio

# Simulated Annealing Schedules

- Classical simulated annealing (SA)
  - Non-zero probability for up-hill move:
$$p = e^{-\Delta C/T}$$
  - Initial temperature:  $T = |\Delta_{\text{avg}} / \ln p|$ ,  $p$  is the initial acceptance rate (typically, close to 1.0),  $\Delta_{\text{avg}}$  is the average cost of the up-hill moves
  - Classical temperature updating function:  $\lambda$  is set to a fixed value (e.g., 0.85)
$$T_{\text{new}} = \lambda T_{\text{old}}, \quad 0 < \lambda < 1$$
- TimberWolf annealing schedule (Sechen and Sangiovanni-Vincentelli, DAC'86)
  - Increase  $\lambda$  gradually from its lowest value (0.8) to its highest value (approximately 0.95) and then gradually decreases  $\lambda$  back to its lowest value.

# Fast Simulated Annealing

- Fast Simulated Annealing (Fast-SA) consists of 3 stages
  - High-temperature random search (temperature  $T \rightarrow$  a very large value)
  - Pseudo-greedy local search ( $T \rightarrow 0$ )
  - Hill-climbing search (increase  $T$  to simulate regular SA)



# Fast Simulated Annealing (cont'd)

- Temperature update ( $T_1$ : initial temperature)

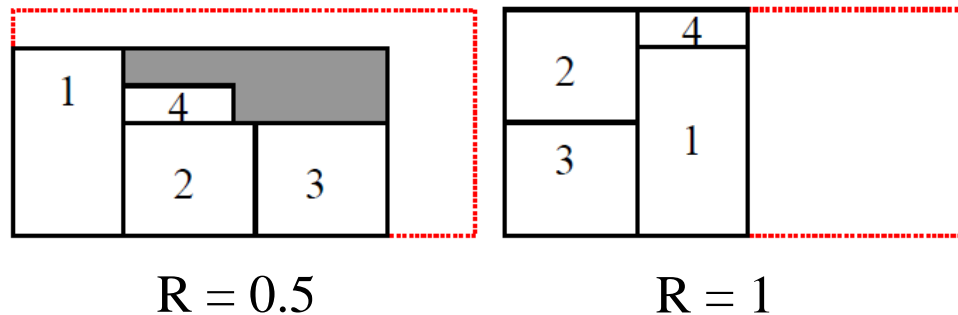
$$T_r = \begin{cases} \frac{\Delta avg}{\ln P} & r = 1 \\ \frac{T_1 \langle \Delta_{cost} \rangle}{rc} & 2 \leq r \leq k \\ \frac{T_1 \langle \Delta_{cost} \rangle}{r} & r > k \end{cases}$$

$\Delta avg$	Average uphill cost
$P$	Initial acceptance rate
$\langle \Delta_{cost} \rangle$	Average cost change for current temperature
$r$	Number of iterations
$c, k$	User-specified parameters (e.g., $c = 100$ , $k = 7$ )

- If  $\langle \Delta_{cost} \rangle$  is larger, temperature decreases slowly.
- If  $\langle \Delta_{cost} \rangle$  is smaller, temperature decreases quickly.

# Adaptive Fast-SA

- The aspect ratio of the best floorplan area in the fixed outline is not the same as that of the outline.
- Decrease the weight of aspect ratio penalty ( $1 - \alpha - \beta$ ) to concentrate more on the floorplan wirelength/area optimization (i.e., increase  $\alpha$  and  $\beta$ ).
  - Adopt an adaptive method to control the weights in the cost function based on  $n$  most recent floorplans.
  - The more feasible floorplans, the less aspect ratio penalty.

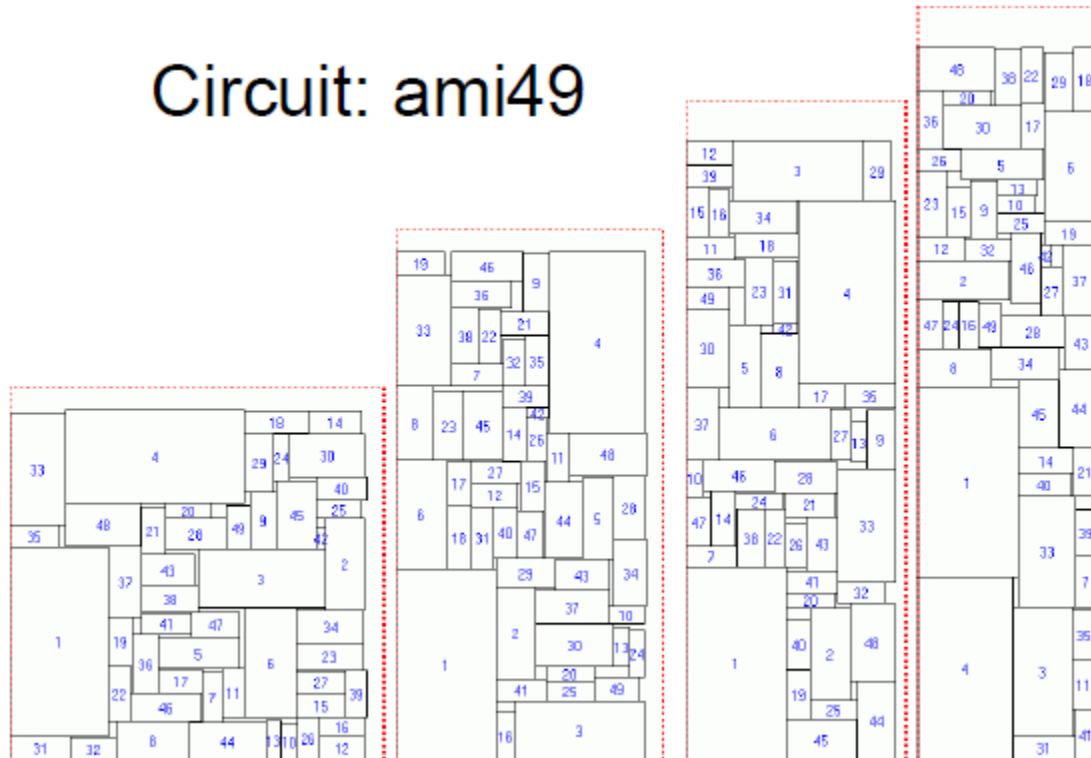


fixed-outline aspect  
ratio  $R^* = 0.5$

# B\*-tree Fixed-Outline Floorplanning Results

- B\*-tree representation
- Fixed-outline floorplans with 10% dead space and aspect ratios 1, 2,3, and 4.

Circuit: ami49



# Floorplanning for Large-Scale Circuits

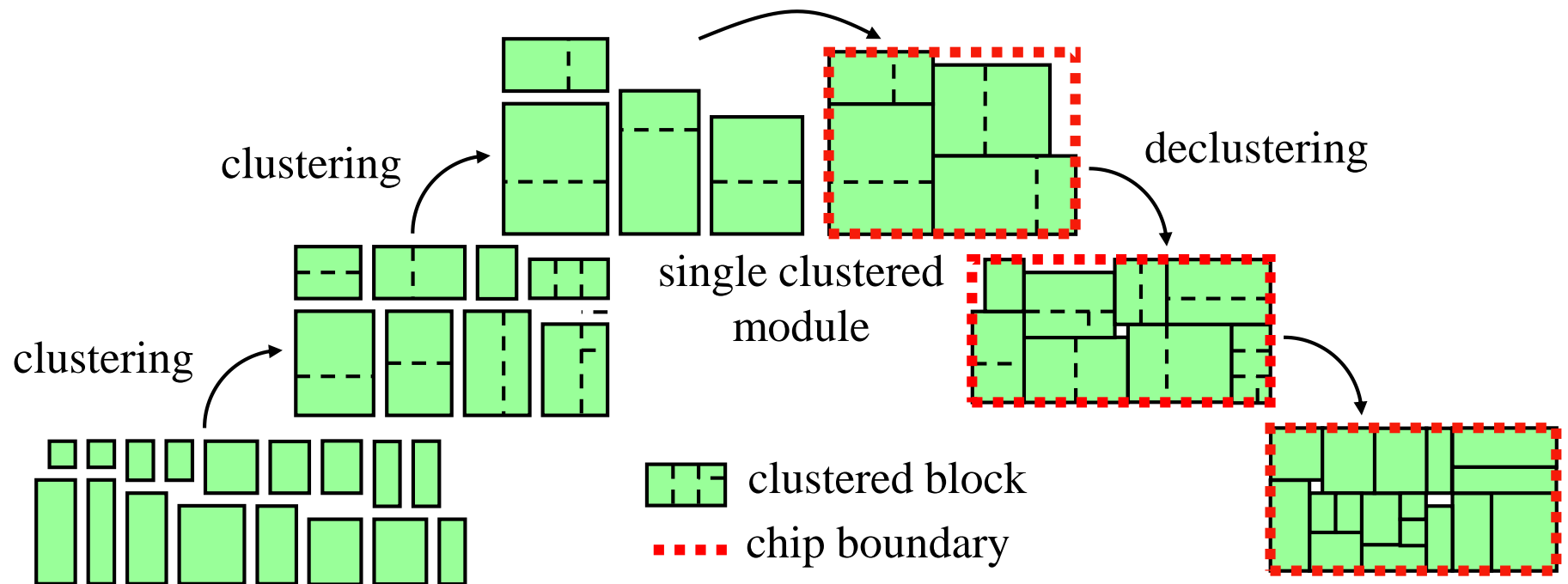
- Lee, Hsu, Chang, Yang, “Multilevel floorplanning/placement for large-scale modules using B\*-trees,” DAC’03 & TCAD’07
- Clustering (bottom-up coarsening ) + declustering (top-down uncoarsening)
  - Clustering
    - Iteratively groups a set of modules based on area utilization and module connectivity
    - Constructs a B\*-tree to keep the geometric relations for the newly clustered modules
  - Declustering
    - Iteratively ungroups a set of the previously clustered modules (i.e., perform tree expansion)
    - Refines the solution using simulated annealing



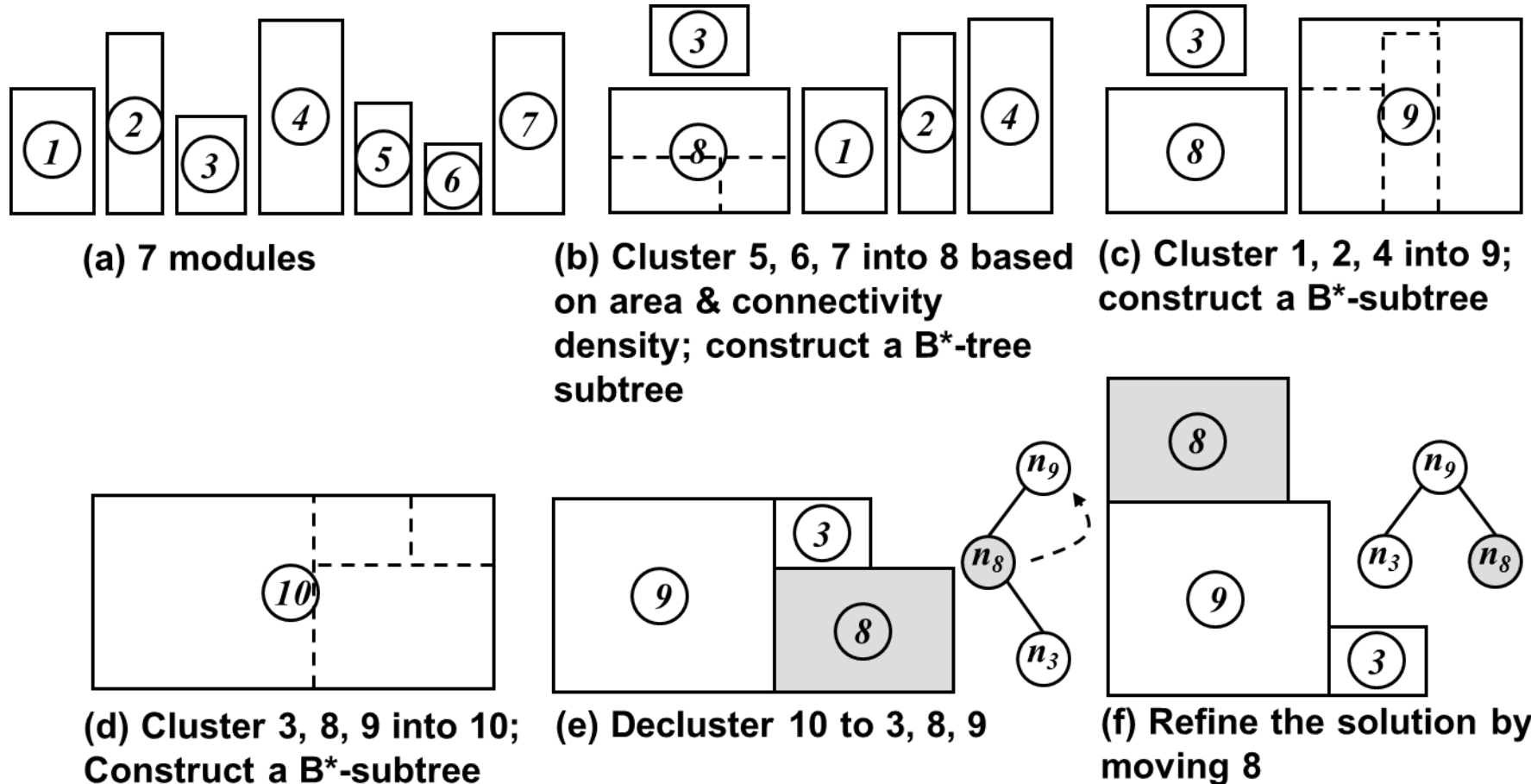
# $\Lambda$ -Shaped Multilevel Floorplanning

Cluster the modules based on area and local connectivity and create clustered modules for the next level.

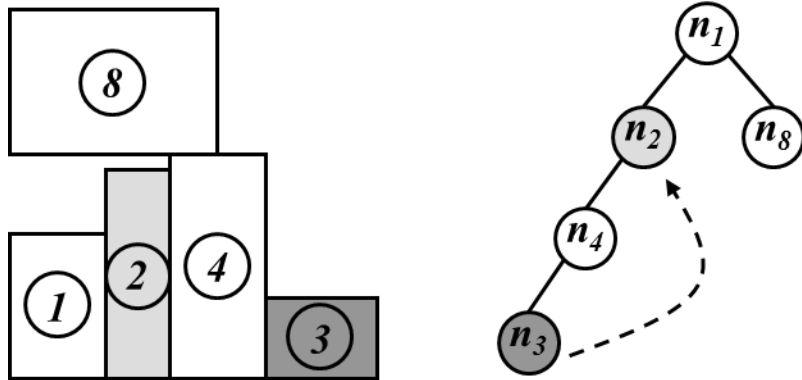
Recursively decluster the clusters and use simulated annealing to refine the floorplan.



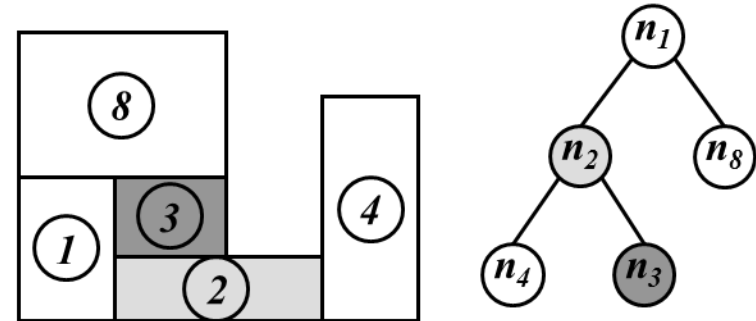
# MB\*-tree: Multilevel B\*-tree



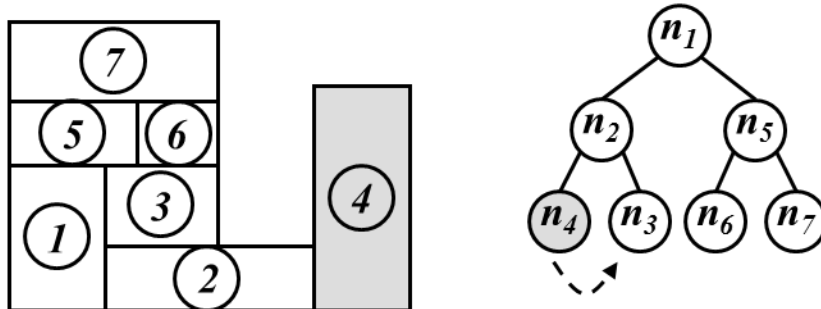
# MB\*-tree: Multilevel B\*-tree (cont'd)



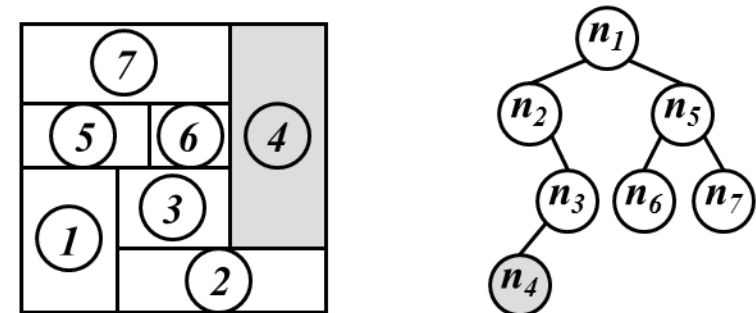
(g) Decluster 9 to 1, 2, 4



(h) Refine the solution by moving 2, 3



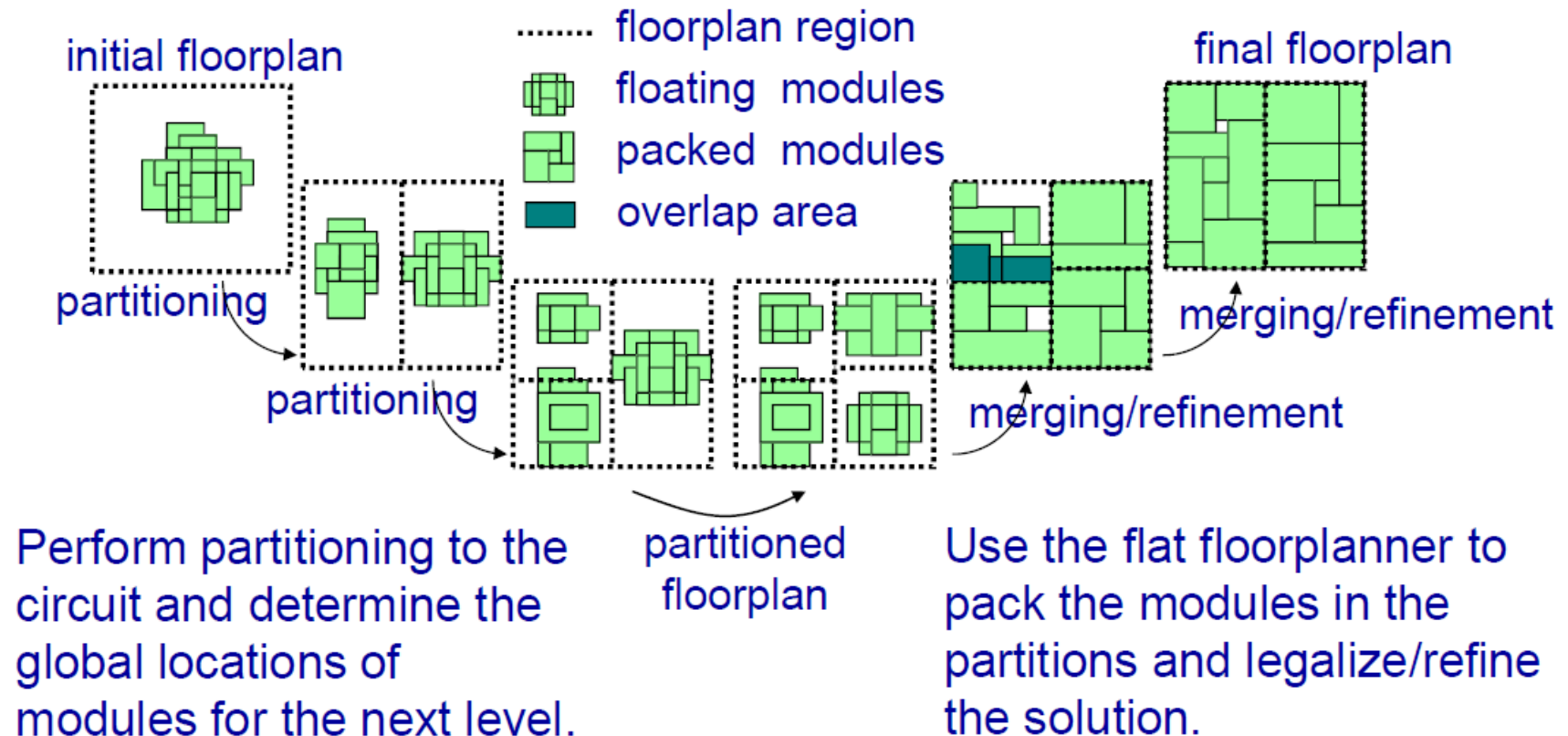
(i) Decluster 8 to 5, 6, 7



(j) Refine the solution by moving 4

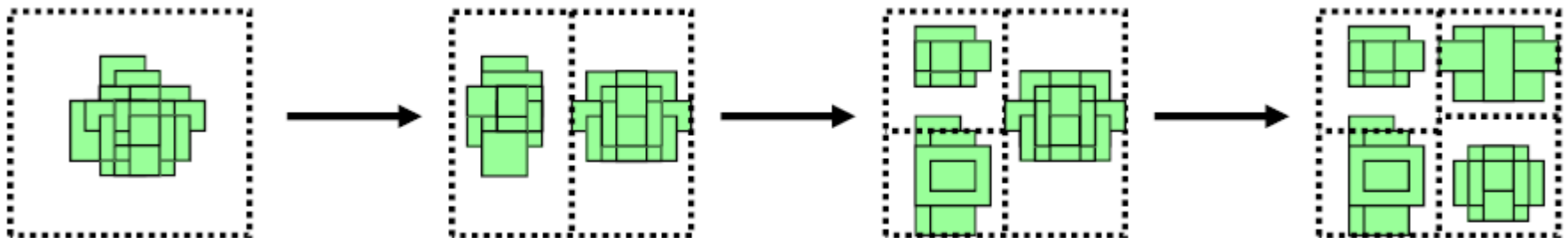
# IMF: V-Shaped Multilevel Floorplanning

- Chen, Chang, Lin, “IMF: interconnect-driven floorplanning for large-scale building-module designs,” ICCAD’05



# Stage 1: Partitioning Stage

- All modules are set to the center of the chip region initially
- Partition the circuit recursively to minimize the interconnect and assign the regions of the modules
- The partitioning stage continues until the number of modules in each partition is smaller than a threshold, and the partitioned floorplan is obtained.

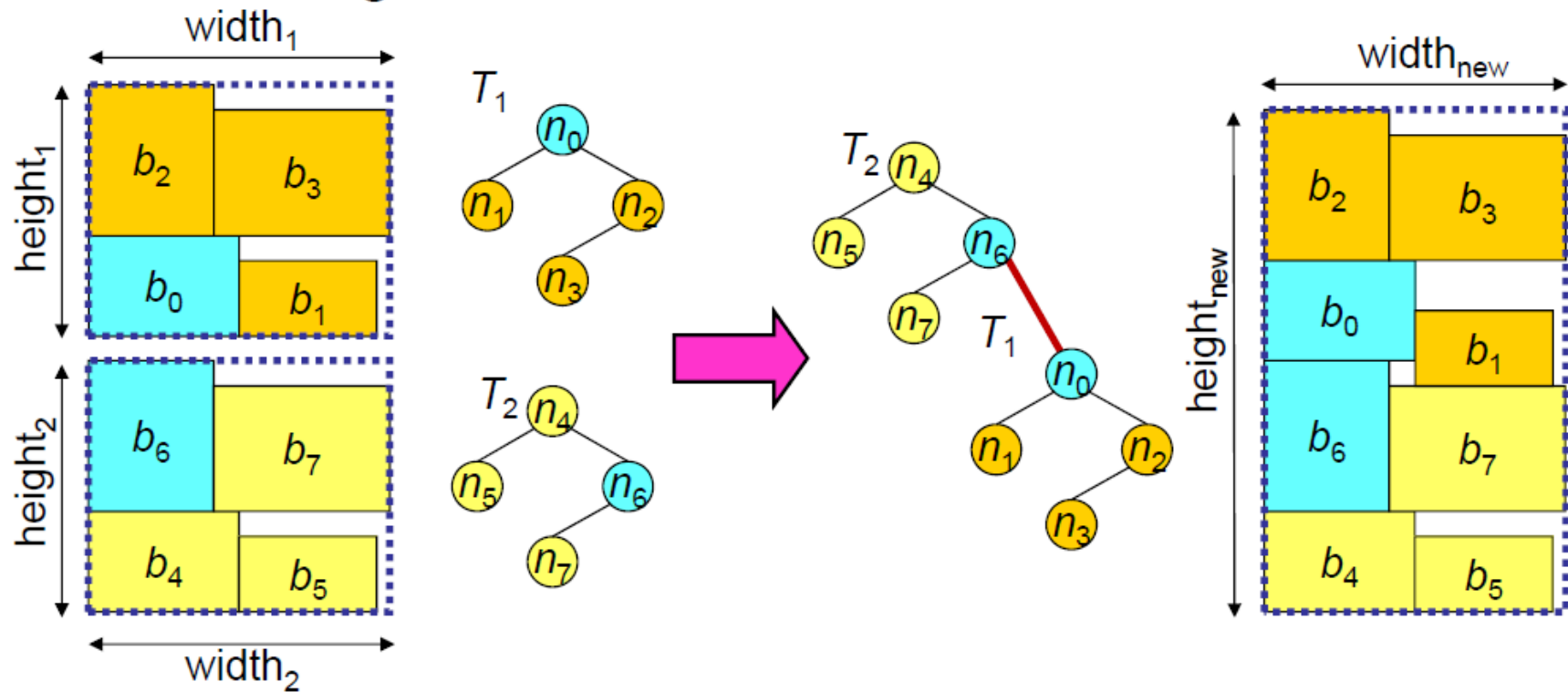


# Stage 2: Merging Stage

- Construct a B\*-tree and find the sub-floorplan for each sub-region (fixed-outline floorplanning)
- Cost function for the simulated annealing: area, wirelength, and aspect ratio penalty
- Merge two B\*-trees (sub-floorplans) to form a new B\*-tree (floorplan) recursively
- Refine the merged sub-floorplan using fixed-outline floorplanning again

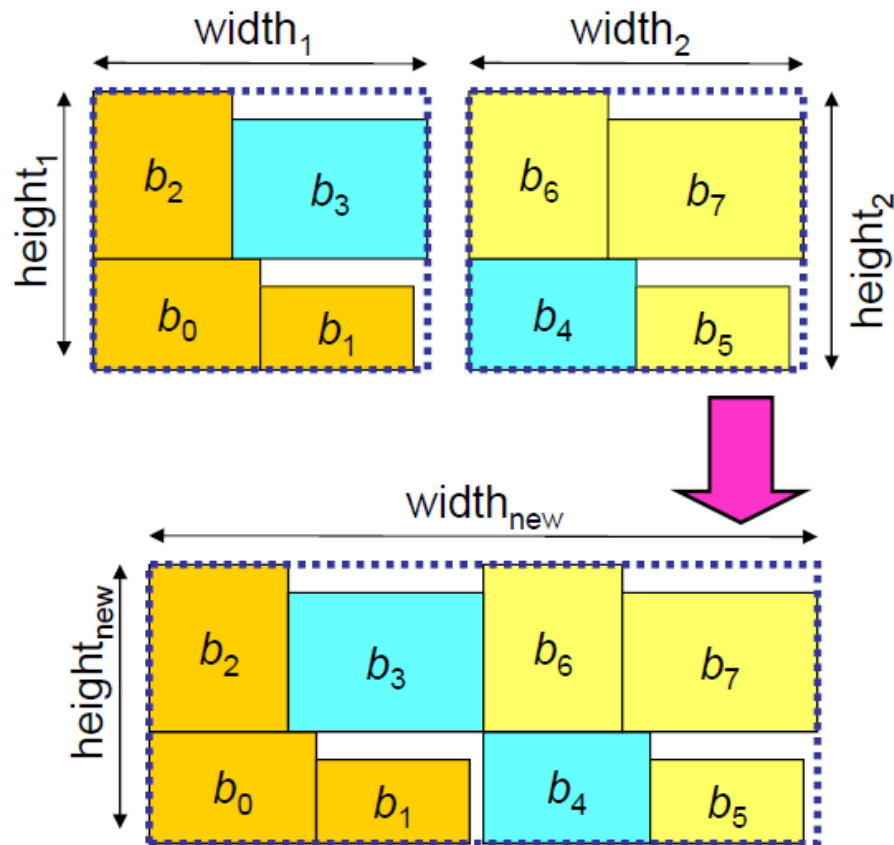
# Vertical Merging

Make the root of the top B\*-tree as the right child of the right-most node of the bottom B\*-tree.



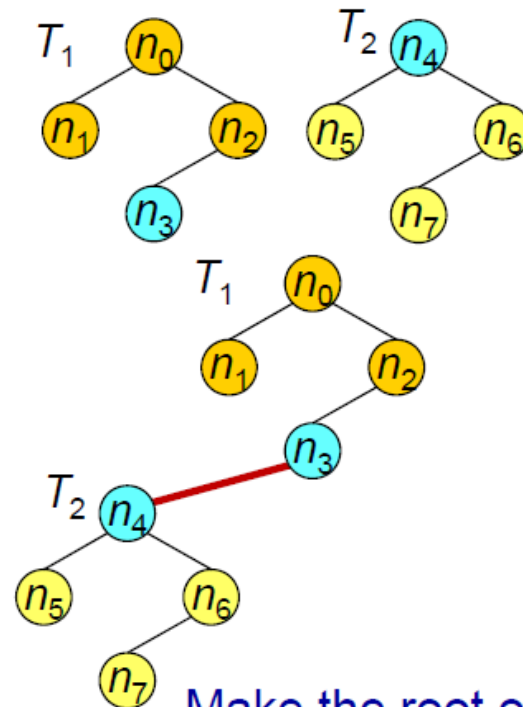
$$\begin{aligned} height_{new} &\leq height_1 + height_2 \\ width_{new} &= \max( width_1, width_2 ) \end{aligned}$$

# Horizontal Merging



$$\text{height}_{\text{new}} = \max(\text{height}_1, \text{height}_2)$$

$$\text{width}_{\text{new}} = \text{width}_1 + \text{width}_2$$

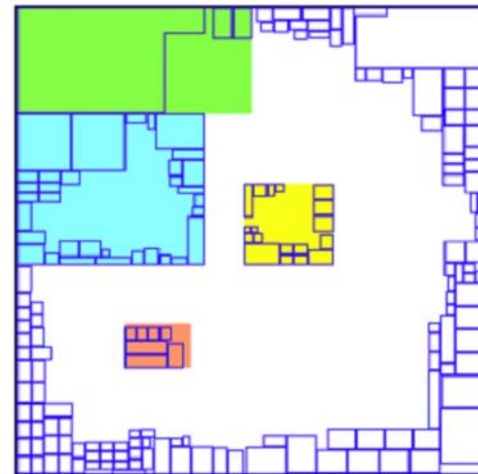
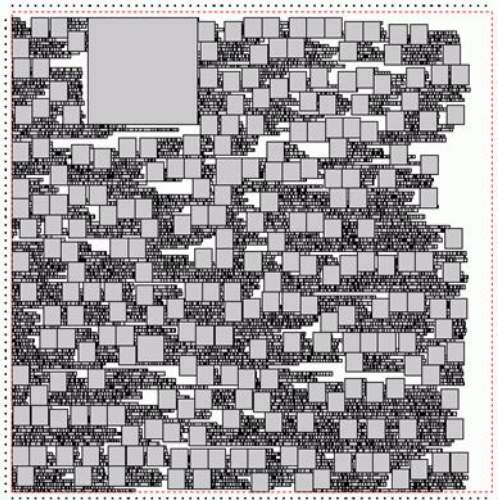


Make the root of the right B\*-tree as the left child of the node corresponding to the right-most module of the left B\*-tree.



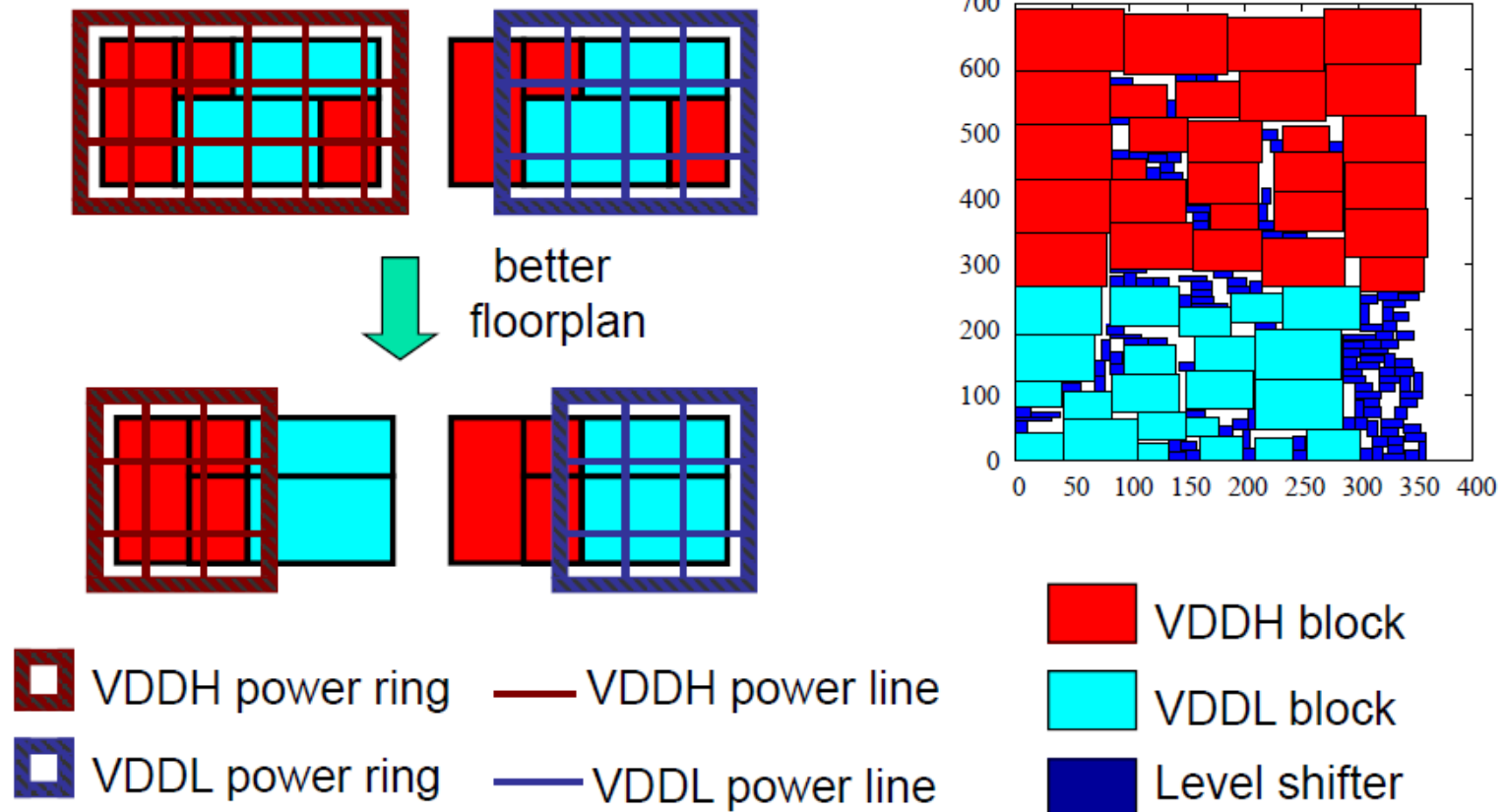
# Macro Placement/Floorplanning

- Mixed sized cell/block placement/floorplanning:  
apply floorplanning techniques for macros to address various design constraints, e.g., range constraints, block rotation, block sizing



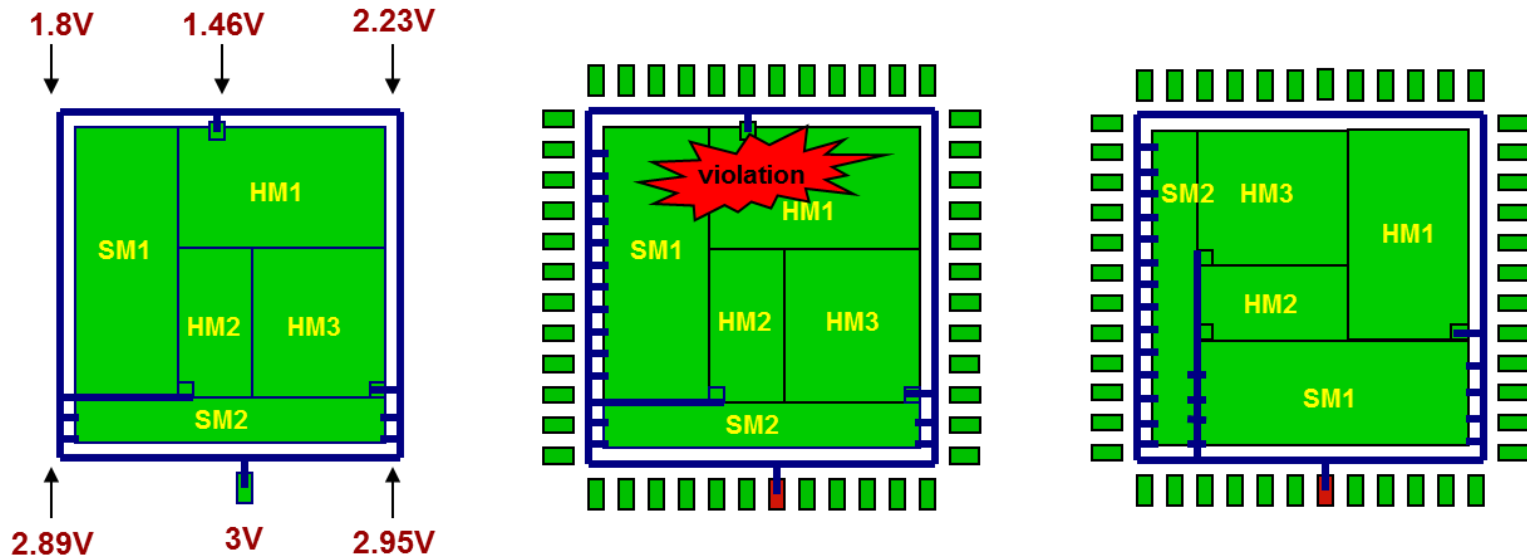
# Voltage Island Aware Floorplanning

- Multiple supply voltages (voltage islands)



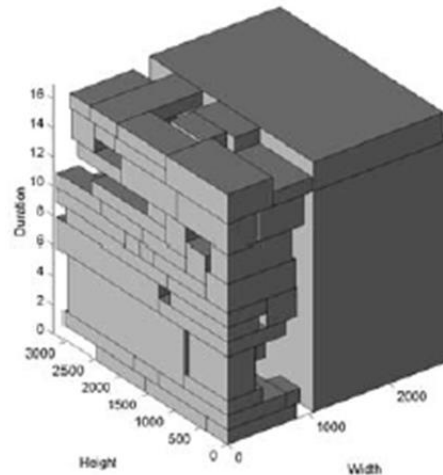
# Voltage Drop Aware Floorplanning

- Power/ground networks for static/dynamic IR drop minimization (voltage drop aware floorplanning)



# Beyond 2D Floorplanning

- Floorplanning for reconfigurable computing



- Floorplanning for digital microfluidic biochips
- SiP/2.5D/3D floorplanning

# Existing Floorplan Representations

- Slicing: **slicing tree**, **normalized Polished expression**
- Mosaic: corner block list (ICCAD'00), twin binary tree (ISPD'01)
- Compacted: **O-tree**, **B\*-tree**, corner sequence (TVLSI'03)
- General: **sequence pair**, bounded-sliceline grid (ICCAD'96), transitive closure graph (DAC'01), TCG-S (DAC'02), adjacent constraint graph (ICCD'04)



# Comparison

Representation	Solution Space	Packing Time	Flexibility
Normalized Polish Expression	$O(n!2^{3n}/n^{1.5})$	$O(n)$	Slicing
Corner Block List	$O(n!2^{3n})$	$O(n)$	Mosaic
Twin Binary Sequence	$O(n!2^{3n}/n^{1.5})$	$O(n)$	Mosaic
O-tree	$O(n!2^{2n}/n^{1.5})$	$O(n)$	Compacted
B*-tree	$O(n!2^{2n}/n^{1.5})$	$O(n)$	Compacted
Corner Sequence	$\leq (n!)^2$	$O(n)$	Compacted
Sequence Pair	$(n!)^2$	$O(n^2)$	General
BSG	$O(n!C(n^2, n))$	$O(n^2)$	General
Transitive Closure Graph	$(n!)^2$	$O(n^2)$	General
TCG-S	$(n!)^2$	$O(n \lg n)$	General
Adjacent Constraint Graph	$O((n!)^2)$	$O(n^2)$	General

# Existing Floorplanning Problems

- Outline free (variable die)
- Fixed outline (fixed die)
- Hard modules only
- Soft (and hard) modules
- Large scale
- Mixed size
- Pre-placed modules
- Range-constrained modules
- Boundary-constrained modules
- Abutment-constrained modules
- Symmetry-constrained modules
- Rectilinear modules
- Analog placement
- Beyond 2D

# Existing Floorplanning Problems (cont'd)

- Co-synthesis with
  - Voltage islands
  - Power supply planning (voltage drop)
  - Interconnect planning
  - Bus planning
  - Buffer planning
  - Noise
  - SoC test scheduling
  - Micro-architecture pipelining
  - Double patterning