

Unit 4: Floorplanning

- Course contents:

slicing

- Normalized polish expression for slicing floorplans

non-slicing

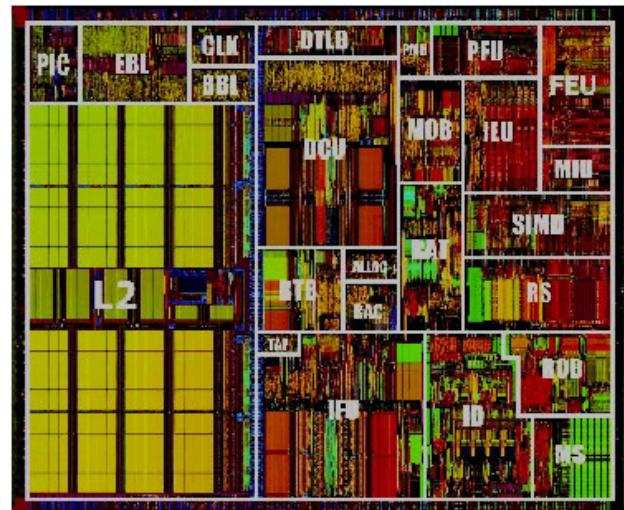
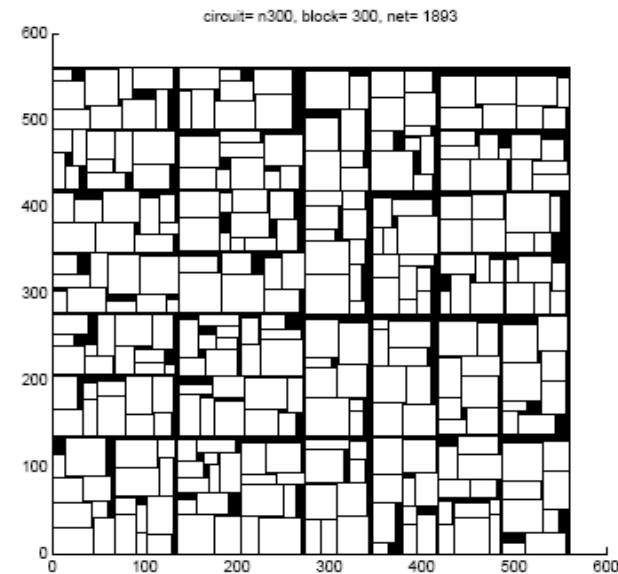
- B*-trees for compacted and large-scale floorplans, floorplanning with various constraints

non-slicing

- P*-admissible floorplan representations (Sequence pair, TCG, etc)
- Comparisons on recently developed floorplan representations
- ILP for general floorplans

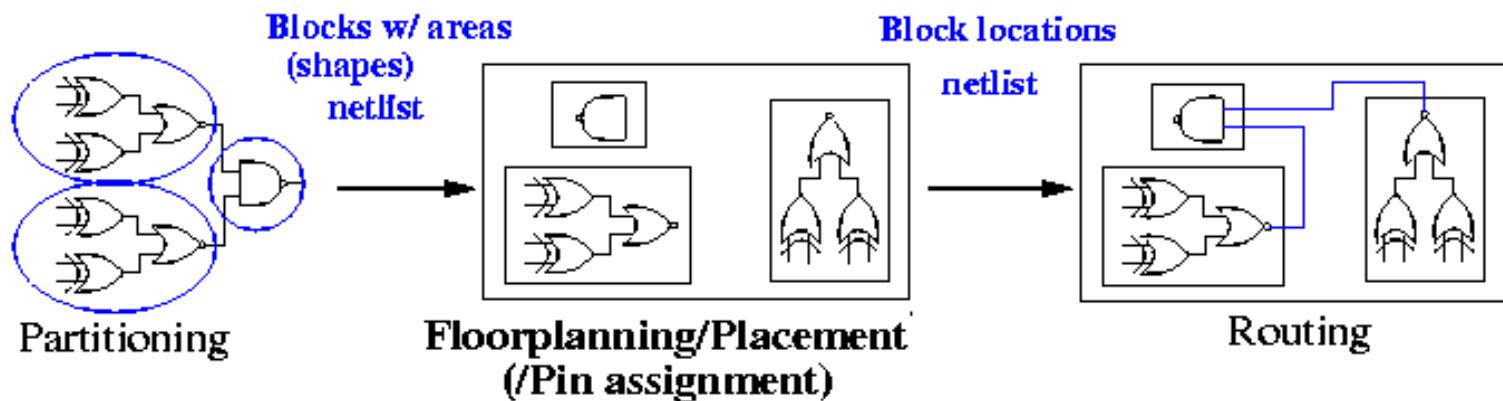
- Readings

- W&C&C: Chapter 10
- S&Y: Chapter 3



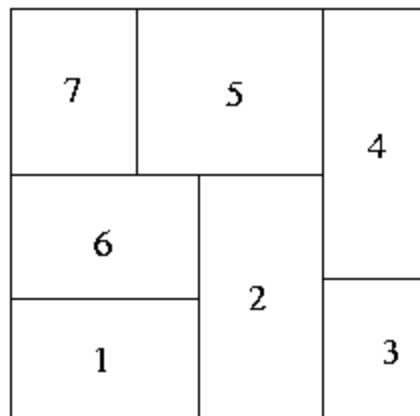
Floorplanning/Placement

- . Partitioning leads to
 - Blocks with well-defined **areas and shapes** (rigid/hard blocks).
 - Blocks with approximated areas and no particular shapes (flexible/soft blocks).
 - A **netlist** specifying connections between the blocks.
- . Objectives 利用partition後的結果進行floorplanning
 - Find **locations** for all blocks.
 - Consider shapes of soft block and pin locations of all the blocks.

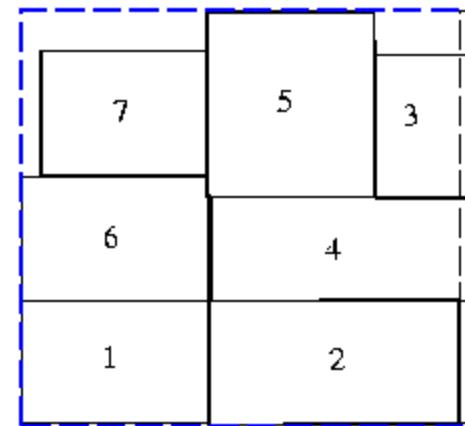


Floorplanning Problem

- . Inputs to the floorplanning problem:
 - A set of blocks, hard or soft.
 - Pin locations of hard blocks. 因為hard block不能隨便亂動，因此會有固定的pin腳位置
 - A netlist.
- . Objectives: minimize area, reduce wirelength for (critical) nets, maximize routability (minimize congestion), determine shapes of soft blocks

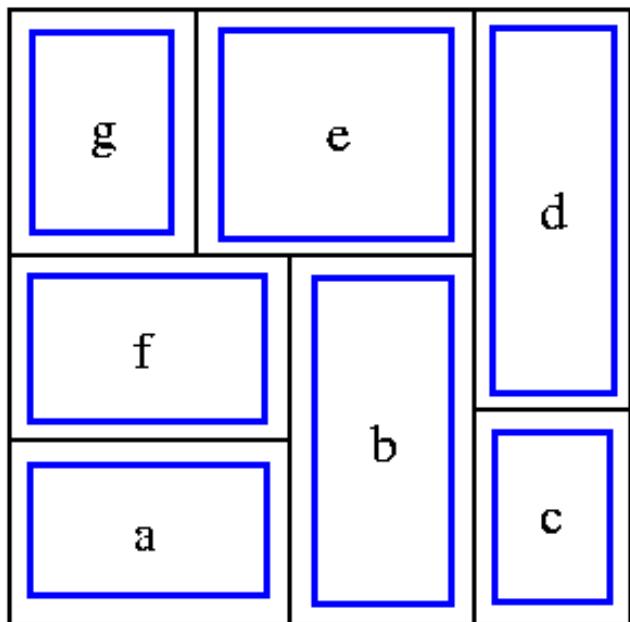


An optimal floorplan,
in terms of area

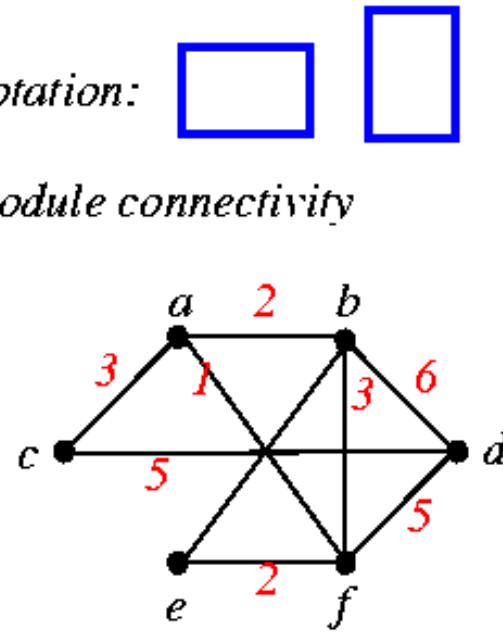


A non-optimal floorplan

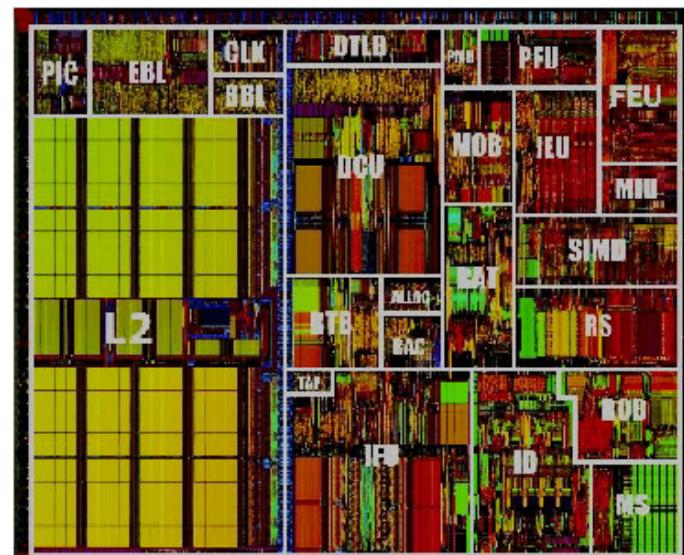
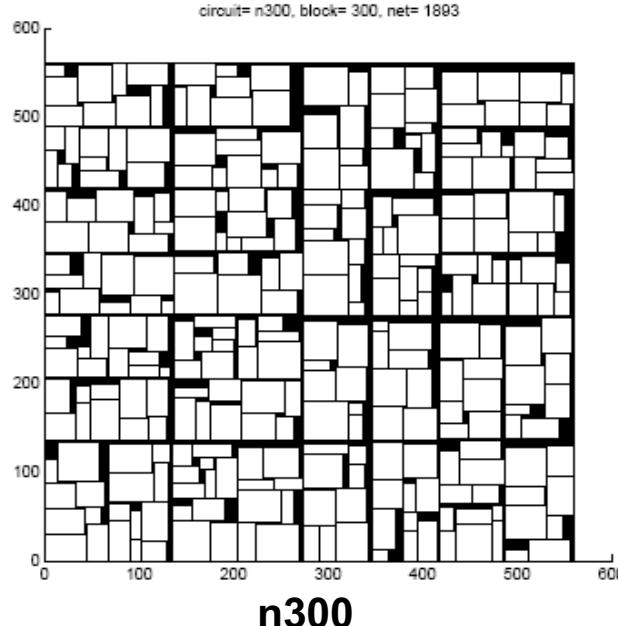
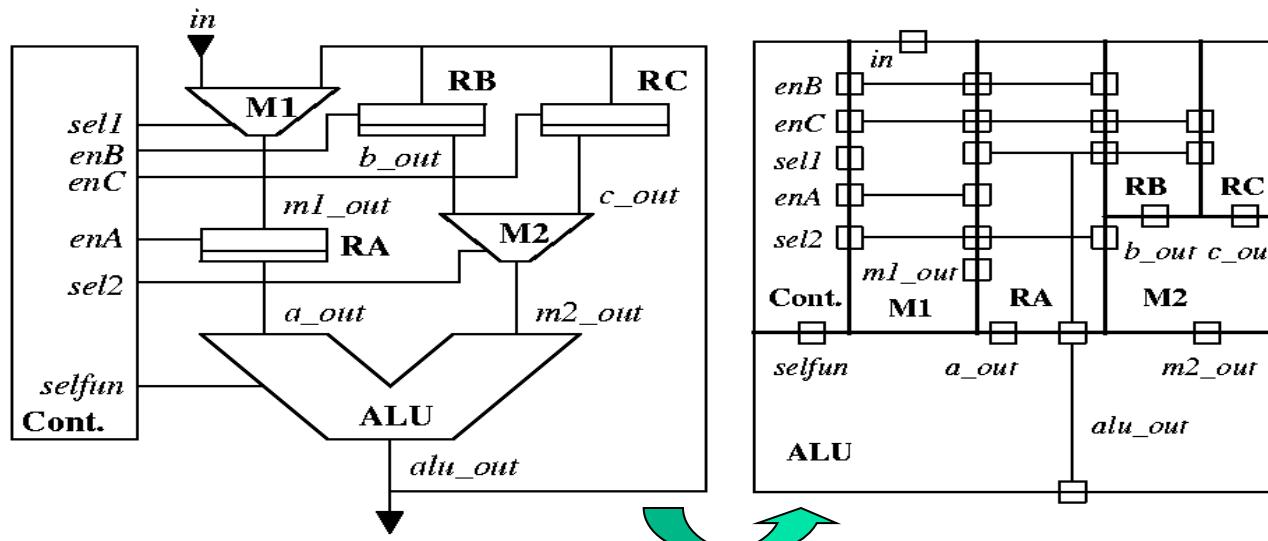
Floorplan Design



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



Floorplan Examples



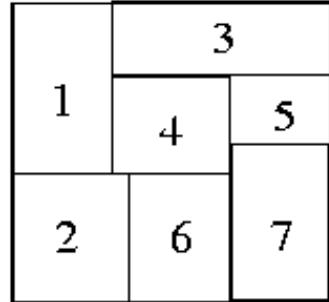
Courtesy of Prof. Y.-W. Chang and H.-M. Chen

Floorplanning in Design Flow

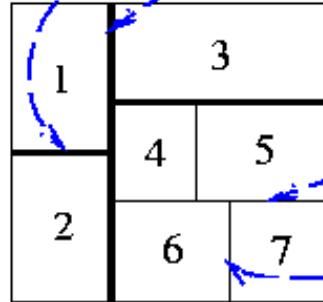
- . An IC is a 2-D medium; consider the dimensions of blocks in early stages of the design helps to improve the quality.
- . Floorplanning fits very well in a *top-down* design strategy.
- . Floorplanning gives early feedback
 - Suggests valuable architectural modifications
 - Estimates the whole chip area
 - Estimates delay and congestion due to wiring
 - Many more, e.g., buffer location, IR drop, etc.
- . Floorplanning considers the *flexibility* in the shapes and terminal locations of blocks.
- . The flooplanning problem is NP-hard (cf. 2-D bin packing).

Slicing Floorplan Structure

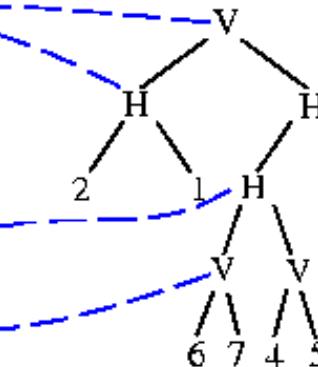
- **Rectangular dissection:** Subdivision of a given rectangle by a finite # of horizontal and vertical line segments into a finite # of non-overlapping rectangles.
- **Slicing structure:** a rectangular dissection that can be obtained by repetitively subdividing rectangles horizontally or vertically.
- **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:** One in which no node and its **right** child are the same.



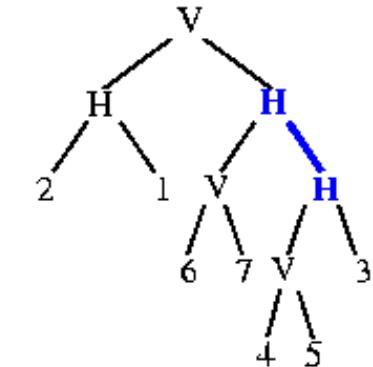
Non-slicing floorplan



Slicing floorplan



A slicing tree (skewed)



Another slicing tree (non-skewed)

可能會有多個slicing tree對應到同一個floorplan

但是 skewed slicing tree與其對應的floorplan是 1-to-1

可以藉由這個特性來讓每次測試的floorplan都是不一樣的，避免浪費時間在檢查同一組floorplan

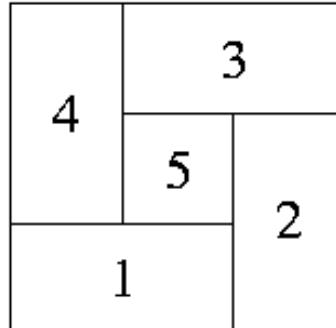
Floorplan Order

- **Wheel:** The smallest non-slicing floorplans (Wang and Wong, TCAD, Aug. 92).
- **Order of a floorplan:** a slicing floorplan is of order 2.
- **Floorplan tree:** A tree representing the hierarchy of partitioning.

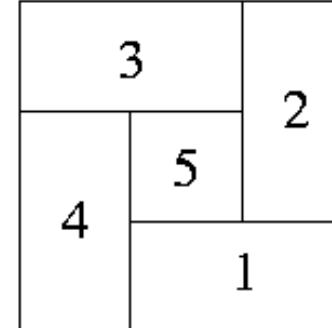
只要是non-slicing floorplan · 其
內部必定隱含wheel的結構
通常這種結構的演算法效率不
會很高

non-slicing floorplan
可以用slicing + wheel
的tree來表示

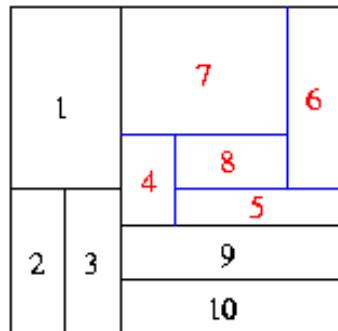
逆時針



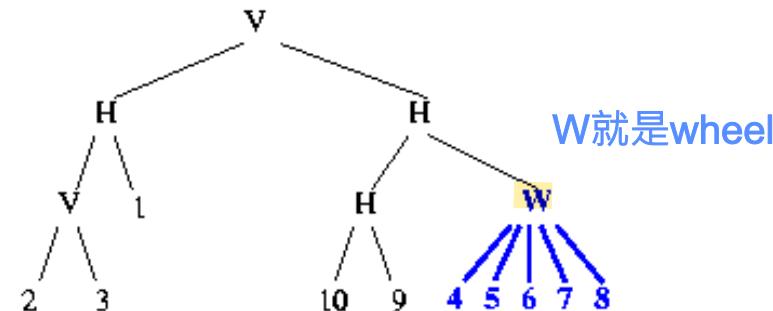
順時針



The two possible wheels.



A floorplan of order 5



Corresponding floorplan tree

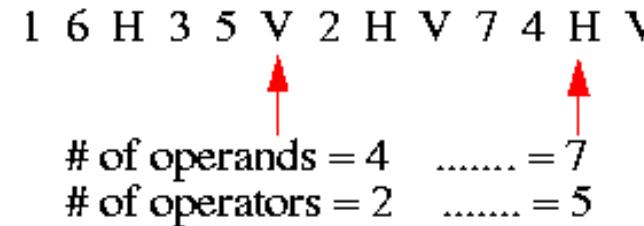
Slicing Floorplan Design by Simulated Annealing

- Related work
 - Wong & Liu, “A new algorithm for floorplan design,” DAC-86.
 - Considers slicing floorplans.
 - Wong & Liu, “Floorplan design for rectangular and L-shaped modules,” ICCAD'87.
 - Also considers L-shaped modules.
 - Wong, Leong, Liu, *Simulated Annealing for VLSI Design*, pp. 31--71, Kluwer Academic Publishers, 1988.
- Ingredients
 - solution space search space要是discrete的，因此我們要用NPE來套用在SA中
 - neighborhood structure 經過"一次變動"所得到的新slicing tree
 - cost function 評估當前slicing tree好壞的指標
 - annealing schedule 容許SA溫度往上的程度

用array比用tree更有效率
因此可以利用一個string來表示slicing tree
--> polish expression

Solution Representation

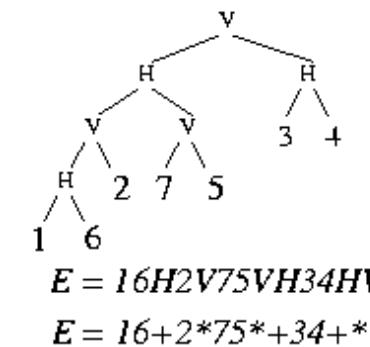
- An expression $E = e_1 e_2 \dots e_{2n-1}$, where $e_i \in \{1, 2, \dots, n, H, V\}$, $1 \leq i \leq 2n-1$, is a **Polish expression** of length $2n-1$ iff
 - every operand j , $1 \leq j \leq n$, appears exactly once in E ;
 - (the balloting property) for every subexpression $E_i = e_1 \dots e_i$,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



- Polish expression \leftrightarrow Postorder traversal. 利用post order traversal來建立slicing tree的polish expression
- ijH : rectangle i on bottom of j ; ijV : rectangle i on the left of j .

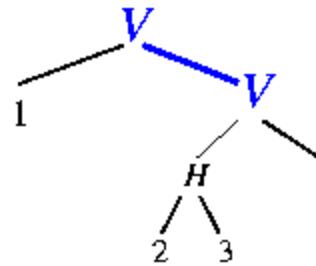
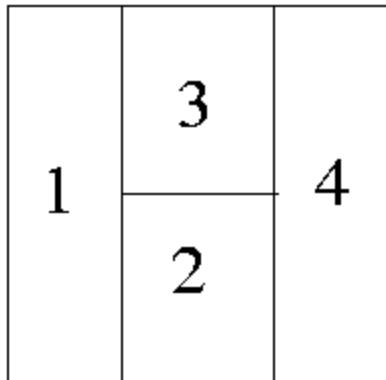
1~7: operand
H, V: operator

7	5	4
6		2
1	2	3

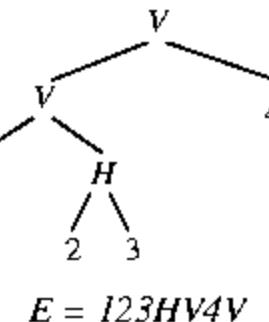


Postorder traversal of a tree!

Redundant Representation

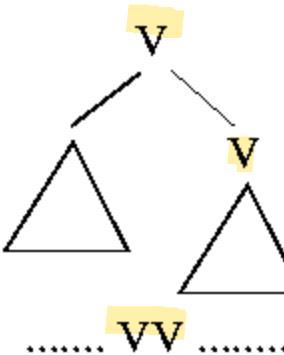
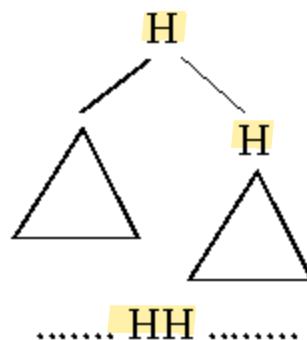


non-skewed!



skewed!

Non-skewed cases



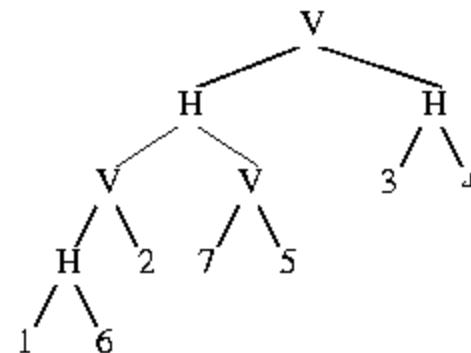
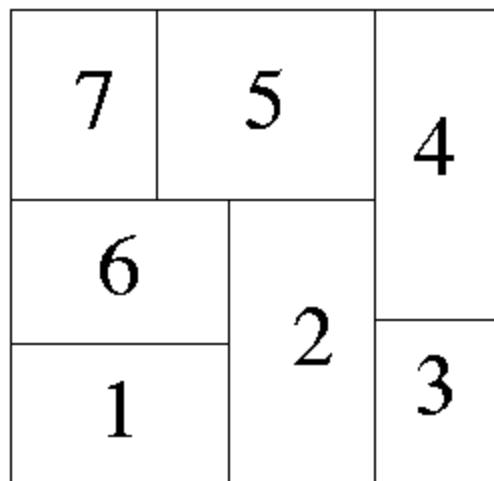
只要polish expression中沒有連續相同的operator，就可以確保不會檢查到重覆的slicing tree
稱此種polish expression為 "normalize polish expression(NPE)"

. **Question:** How to eliminate ambiguous representation?

NPE會對應到唯一的一個skewed slicing tree (?)

Normalized Polish Expression

- A Polish expression $E = e_1 e_2 \dots e_{2n-1}$ 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.

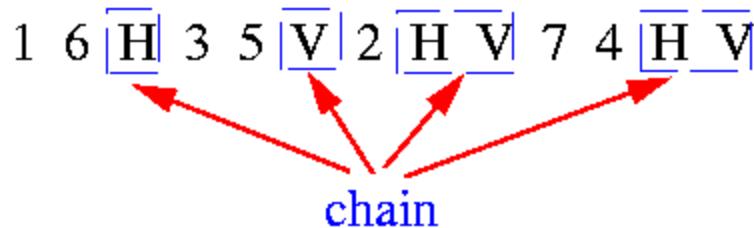


$$E = 16H2V75VH34HV$$

A normalized Polish expression

Neighborhood Structure

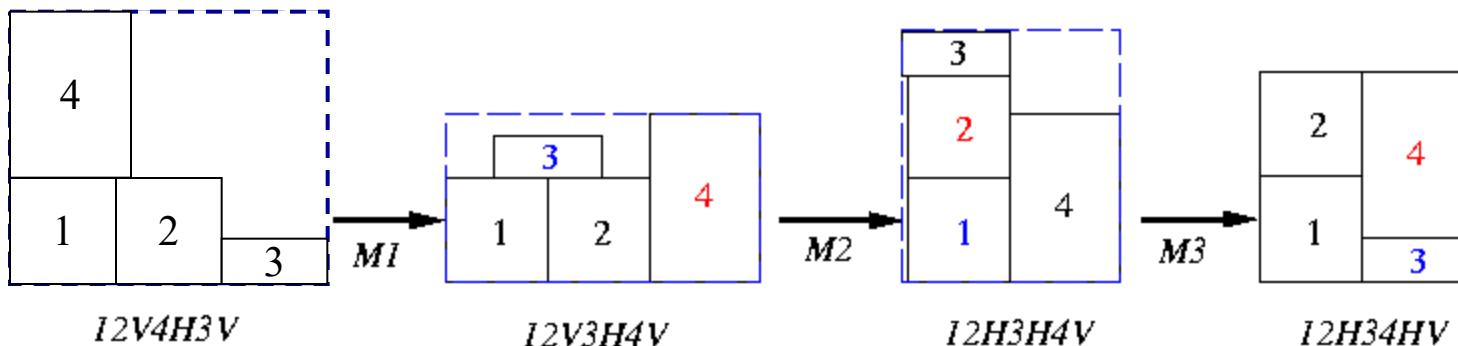
- **Chain:** $HVHVH \dots$ or $VHVHV \dots$



- **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:
 - **M1 (Operand Swap):** Swap two adjacent operands.
 - **M2 (Chain Invert):** Complement some chain ($V = H$, $H = V$).
 - **M3 (Operator/Operand Swap):** Swap two adjacent operand and operator.

M1, M2不論怎麼做都不會怎樣
但做M3的時候要注意會不會違反NPE的balloting property

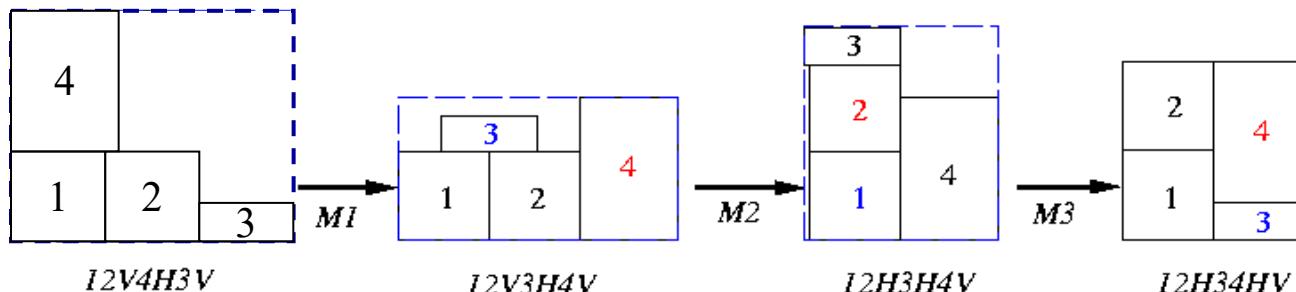
Effects of Perturbation



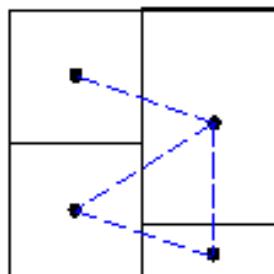
- . **Question:** The balloting property holds during the moves?
 - $M1$ and $M2$ moves are OK.
 - **Check the $M3$ moves!** Reject “illegal” $M3$ moves.
- . **Check $M3$ moves:** Assume that $M3$ swaps the operand e_i with the operator e_{i+1} , $1 \leq i \leq k-1$. Then, the swap will not violate the 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
 - λ : user-specified parameter 如果覺得wiring很重要可以給大一點

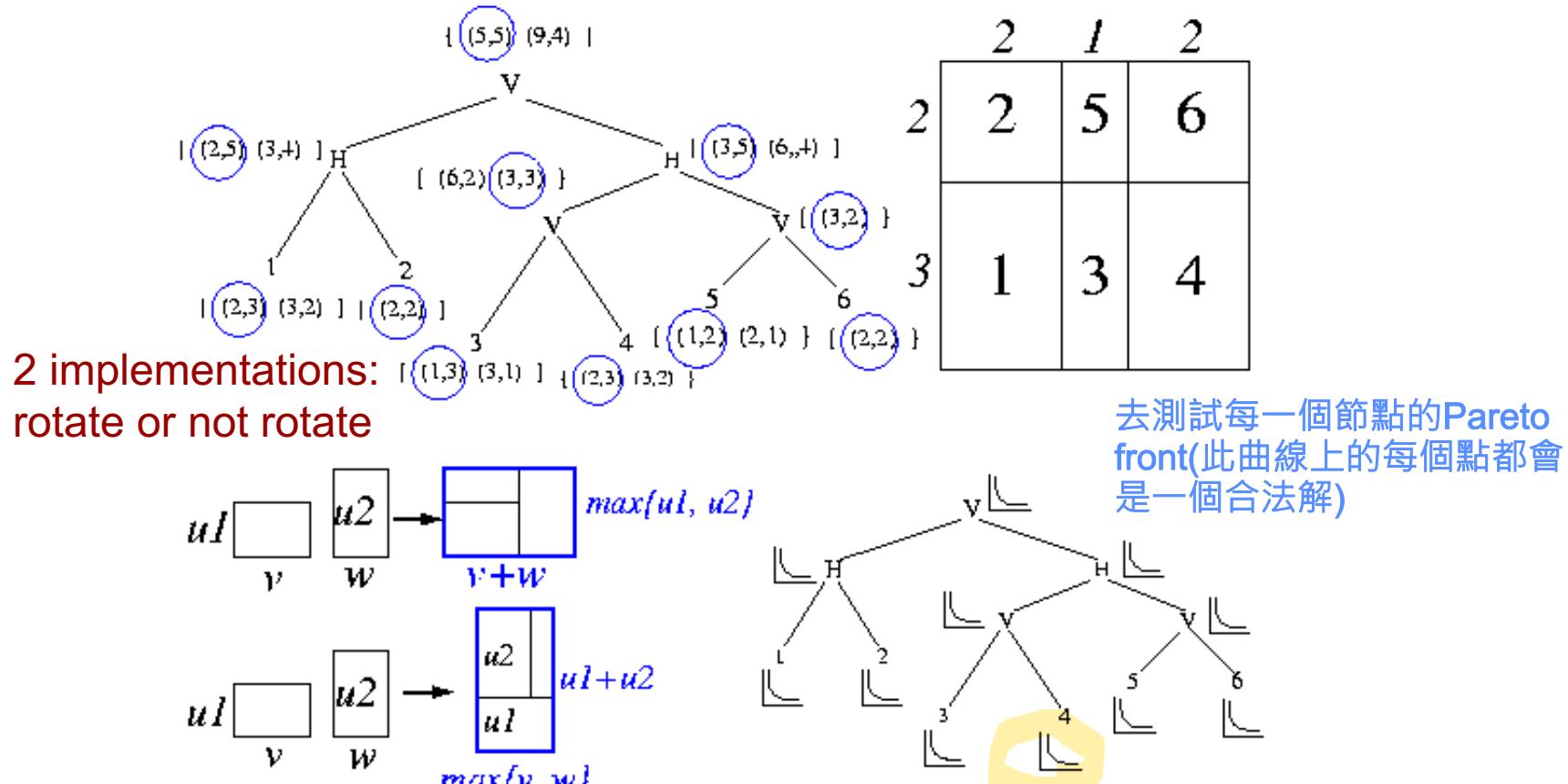


- $W = \sum_{ij} c_{ij} d_{ij}$. 中心點到中心點的距離--> Manhattan Distance · block太大的話可能效果不太好
 - c_{ij} : # of connections between blocks i and j .
 - d_{ij} : center-to-center distance between basic rectangles i and j .

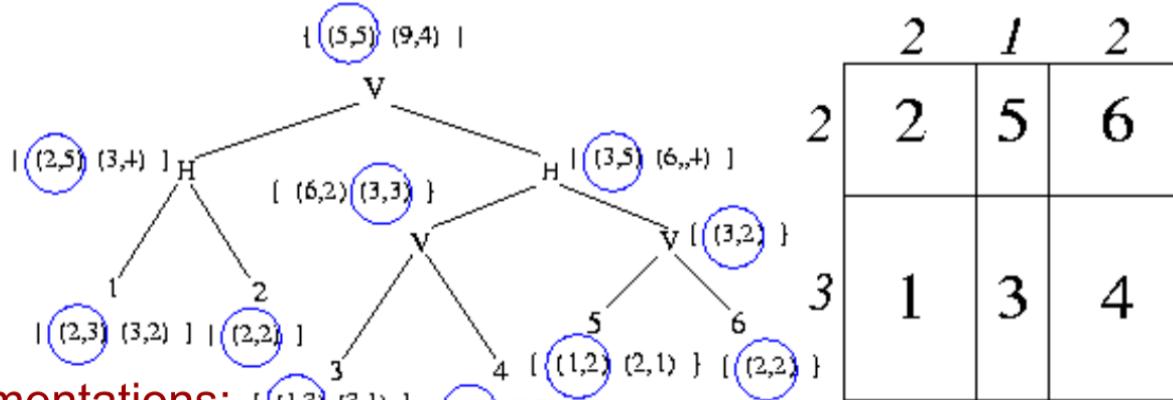


Area Computation for Hard Blocks

- Allow rotation: each block has up to two implementations.



- Wiring cost?
 - Center-to-center interconnection length



2 implementations:
rotate or not rotate

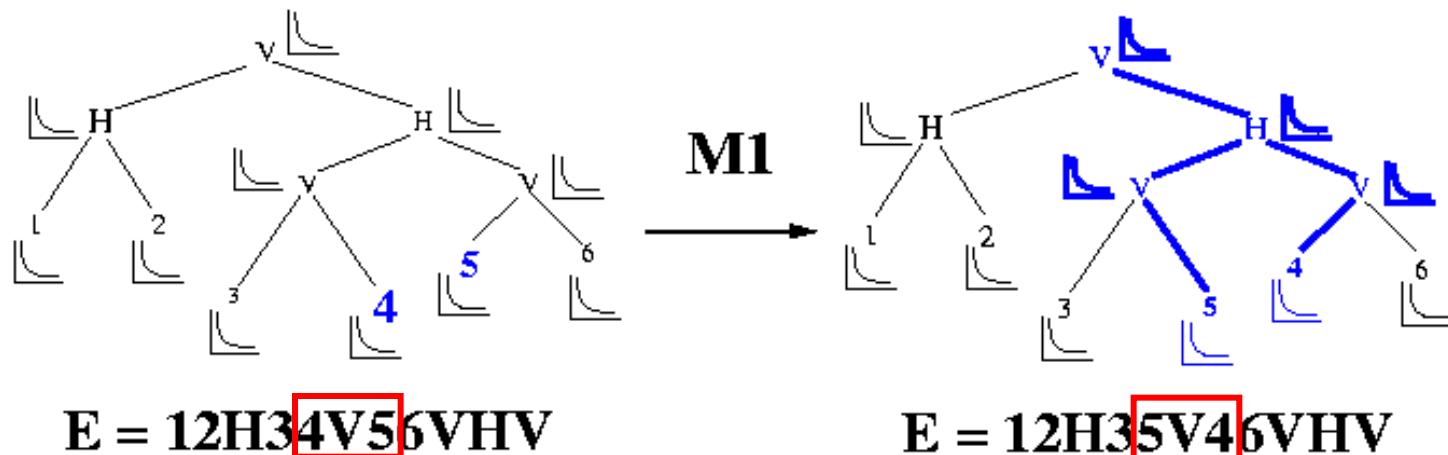
由下往上: 窮舉所有的排列組合，並保留好的組合結果(壞的就丟掉)，一路直行到最上面便會知道最小可能的面積為何

ex: 1, 2算完應該會有4個解，但是最後只保留了兩個，因為另外兩組解完全比不過這兩組解(可能常與寬都大於其他解，因此直接剷除)

由上往下: 做完由下往上後，top down去檢查該解是在何處產生的

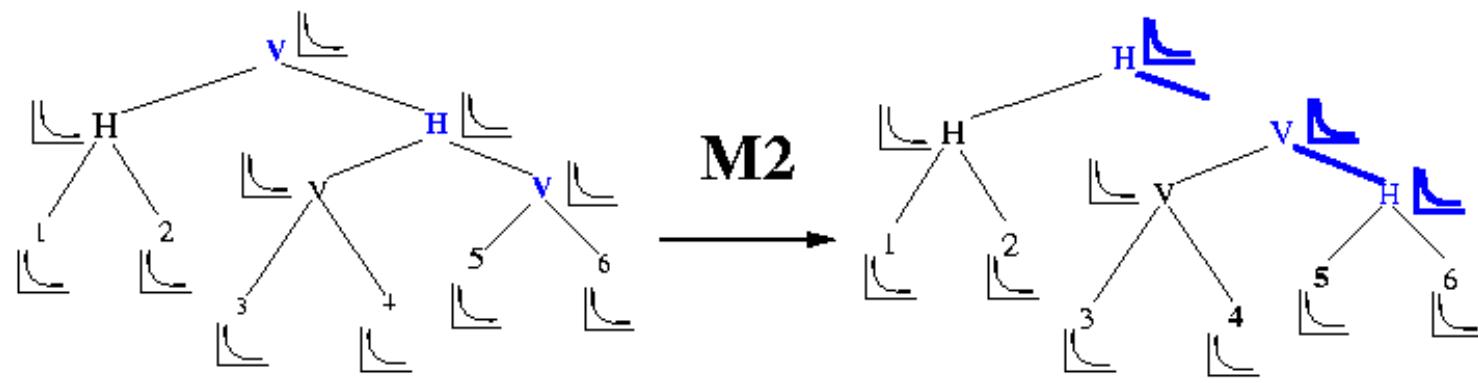
Incremental Computation of Cost Function

- 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.



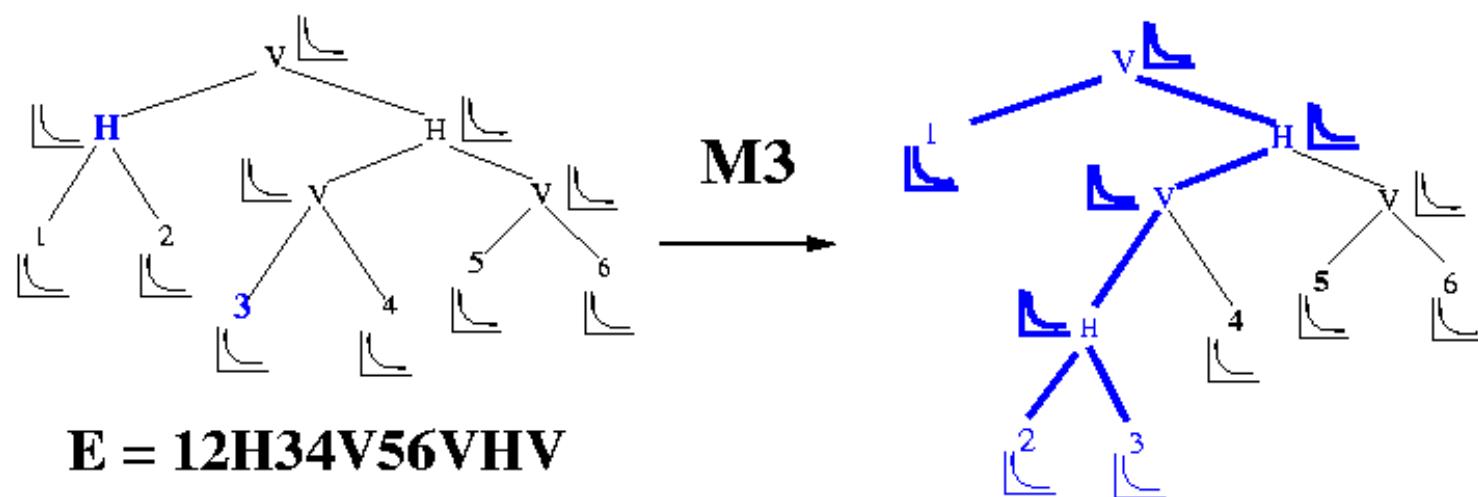
只要重新計算有被修改到的地方即可，避免重複計算浪費時間

Incremental Computation of Cost Function (cont'd)



E = 12H34V56VHV

E = 12H34V56HVH



E = 12H34V56VHV

E = 123H4V56VHV

Annealing Schedule

- Initial solution: $1V3V \dots nV$.

1	2	3			n
---	---	---	--	--	---

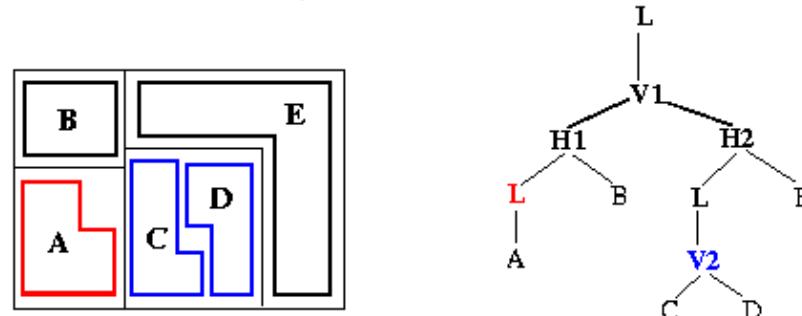
- $T_i = r^i T_0$, $i = 1, 2, 3, \dots$; $r = 0.85$.
- At each temperature, try kn moves ($k = 5-10$).
- Terminate the annealing process if
 - # of accepted moves < 5%,
 - temperature is low enough, or
 - run out of time.

Algorithm: Wong-Liu (P, ε, r, k)

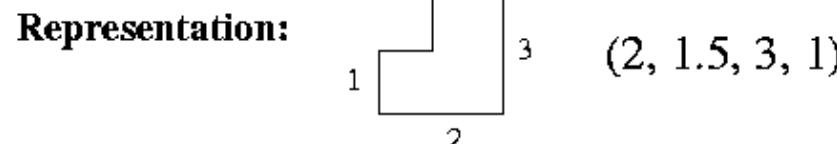
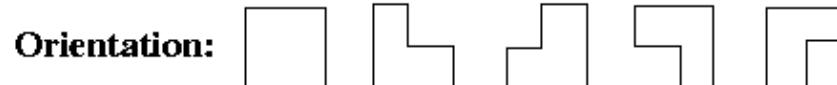
```
1 begin
2  $E \leftarrow 12V3V4V \dots nV$ ; /* initial solution */
3  $Best \leftarrow E; T_0 \leftarrow \frac{\Delta_{avg}}{ln(P)}$ ;  $M \leftarrow MT \leftarrow uphill \leftarrow 0; N = kn;$ 
4 repeat
5    $MT \leftarrow uphill \leftarrow reject \leftarrow 0;$ 
6   repeat
7     SelectMove( $M$ );
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        Choice 1: 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;
13'      Choice 2: Select two adjacent operator  $e_i$  and operand  $e_{i+1}$ ;
14'        if  $(e_i \neq e_{i+2})$  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  $(Random < \frac{-\Delta cost}{e_i T})$ 
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       $T \leftarrow rT$ ; /* reduce temperature */
25 until  $(reject/MT > 0.95)$  or  $(T < \varepsilon)$  or OutOfTime;
26 end
```

Extension to L-Shaped Modules

- Unary operator L : Change an L-shaped figure into a rectangle
- Binary operators V_1, V_2, H_1, H_2 : Combine 2 rectangles or L-shaped figures to form a rectangle or an L-shaped figure.
- Can generate non-slicing floorplan

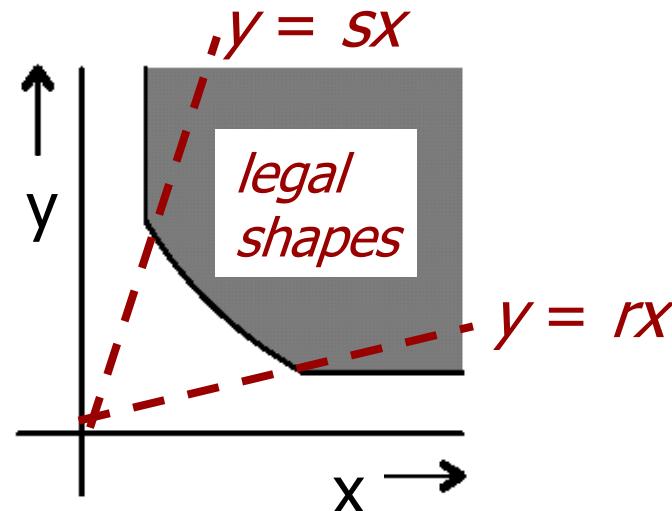
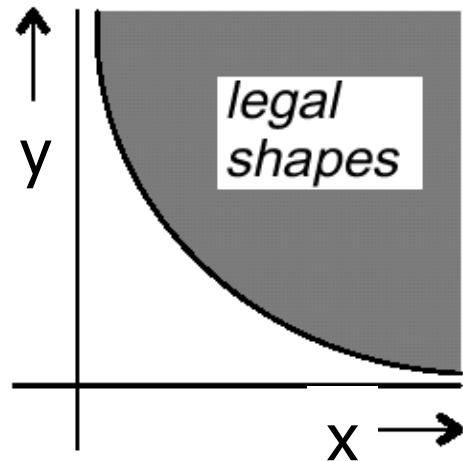


$$E = A \text{ L } B \text{ H1 } C \text{ D } V2 \text{ L } E \text{ H2 } V1 \text{ L}$$



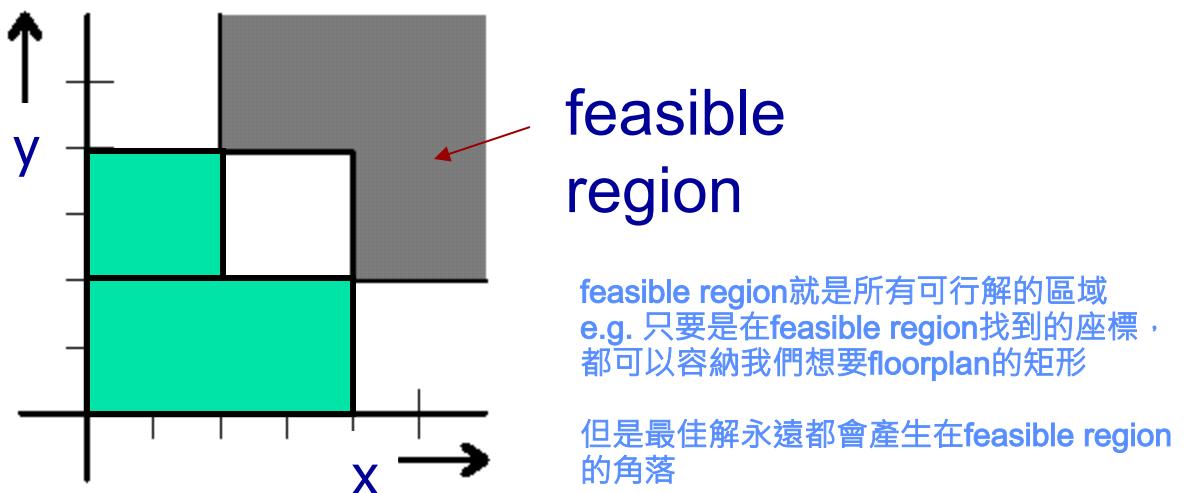
Shape Curve for Floorplan Sizing

- A soft (flexible) blocks b can have different aspect ratios, but is with a fixed area A .
- The shape function of b is a hyperbola: $xy = A$, or $y = A/x$, for width x and height y .
- Very thin blocks are often not interesting and feasible to design
 - Add two straight lines for the constraints on aspect ratios.
 - Aspect ratio: $r \leq y/x \leq s$.



Shape Curve

- Since a basic block is built from discrete transistors, it is not realistic to assume that the shape function follows the hyperbola continuously.
- In an extreme case, a block is rigid/hard: it can only be rotated and mirrored during floorplanning or placement.



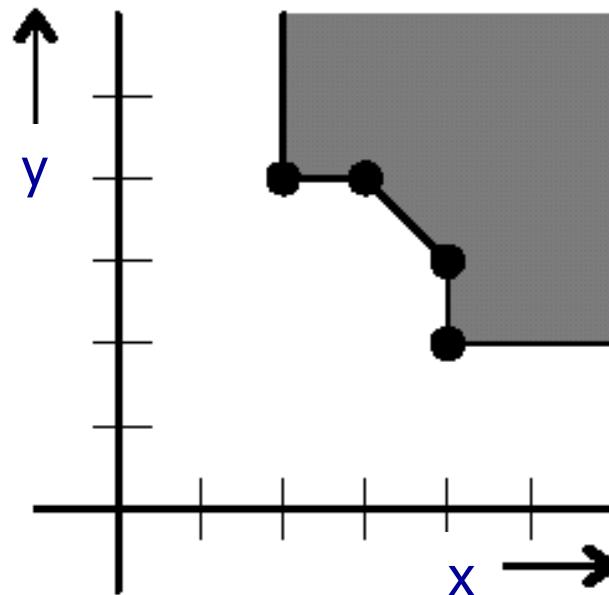
feasible region就是所有可行解的區域
e.g. 只要是在feasible region找到的座標
都可以容納我們想要floorplan的矩形

但是最佳解永遠都會產生在feasible region
的角落

The shape curve of a 2×4 hard block.

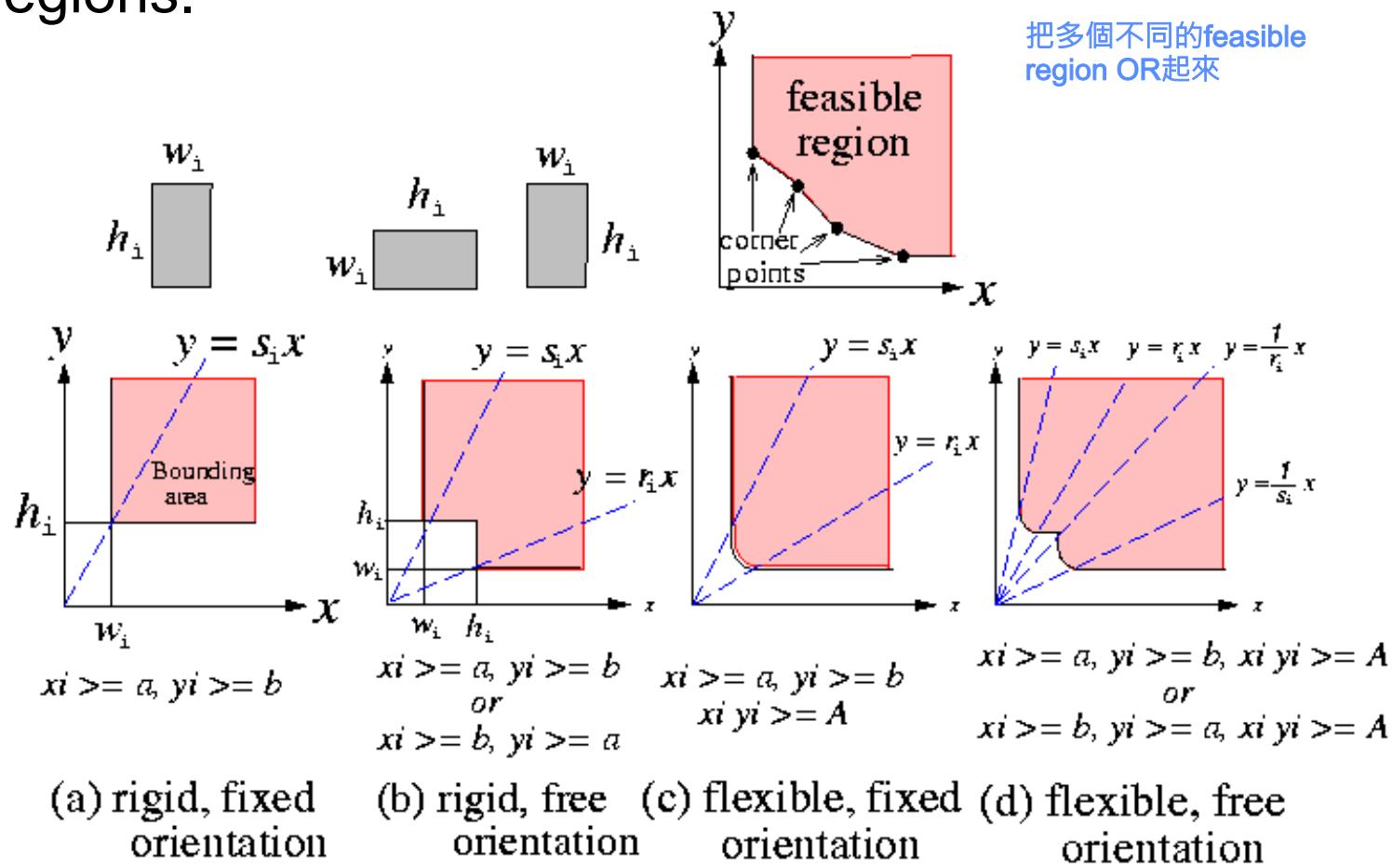
Shape Curve (cont'd)

- . In general, a *piecewise linear* function can be used to approximate any shape function.
- . The points where the function changes its direction, are called the *corner (break) points* of the piecewise linear function.



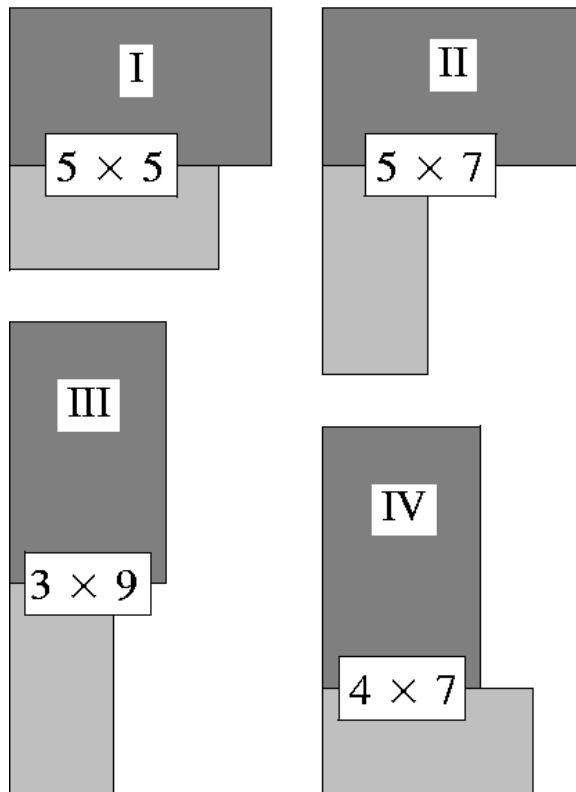
Feasible Implementations

- Shape curves correspond to different kinds of constraints where the shaded areas are feasible regions.

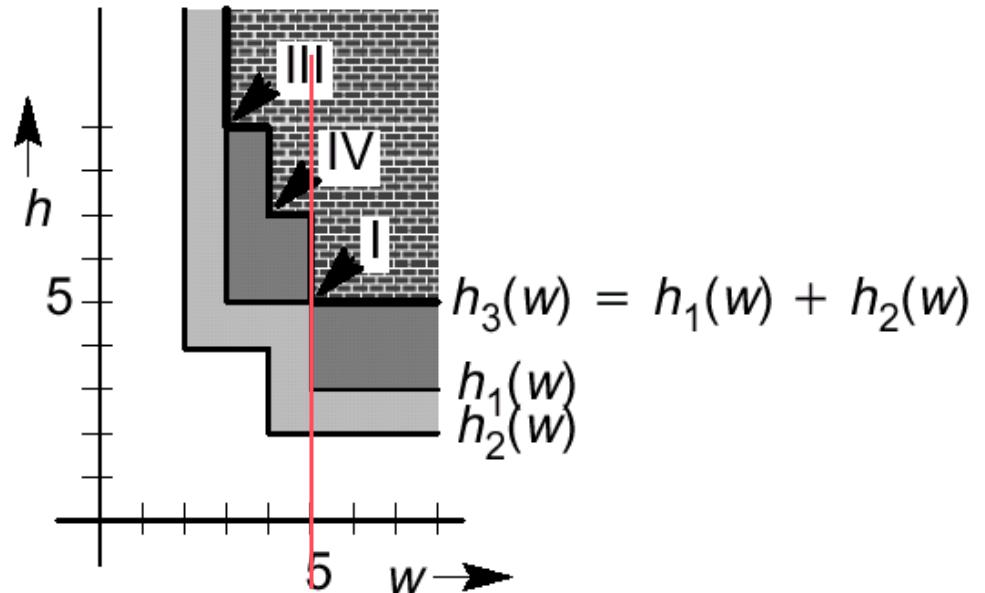


Vertical Abutment

- Composition by vertical abutment (horizontal cut) \Rightarrow the addition of shape functions.



IV dominates II
(IV is a better implementation.)



畫鉛直線，然後都找最後碰到的y值

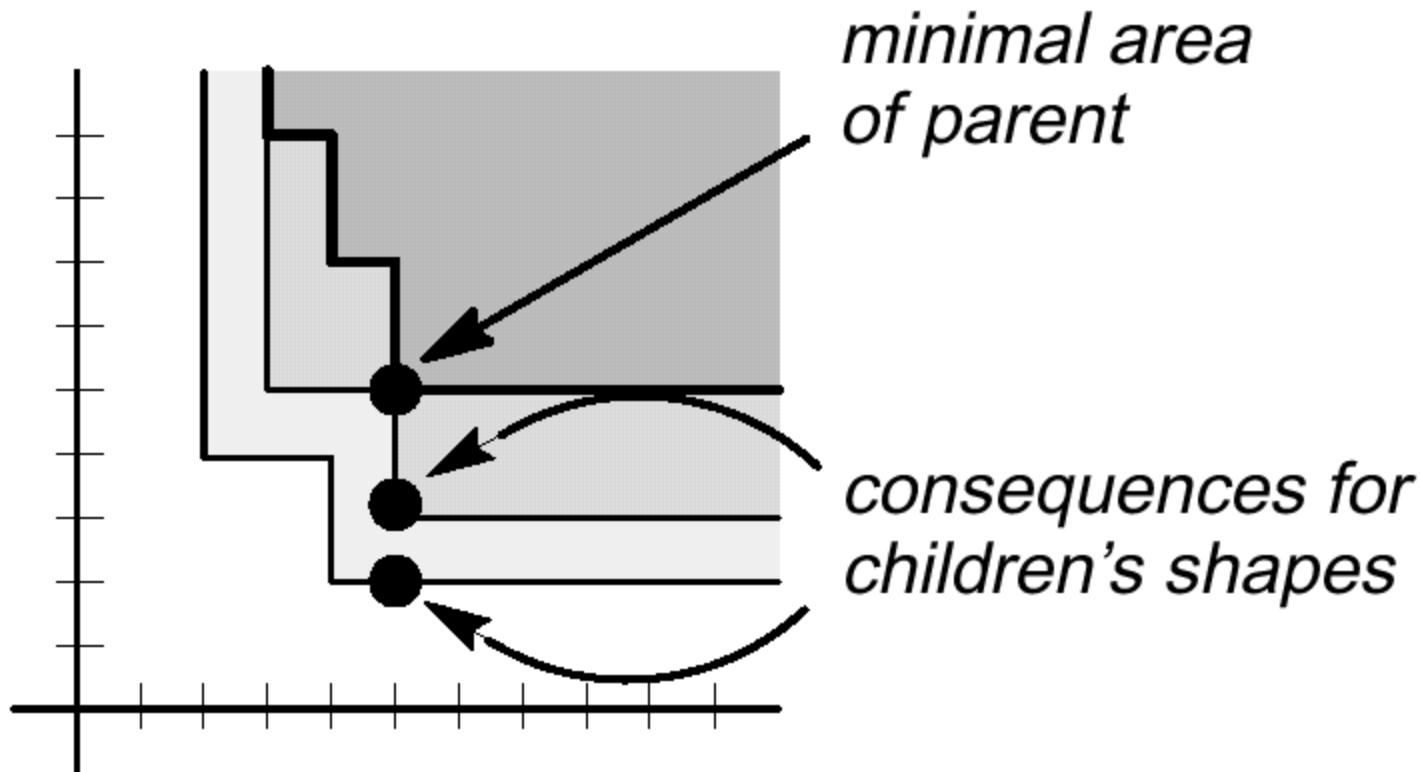
以這例子來說，鉛直線會碰到深色在 $y=3$ 的位置而非 $y=5$ (因為要找最後碰到的)，淺色會碰到 $y=2$ 的位置

因此他們結合(horizontal cut)的高度會是 $2+5=7$
所以磚塊紋路在 $x=5$ 的高度會是 $y=7$

如果要找vertical cut的結合方式
(水平結合)就要畫水平線

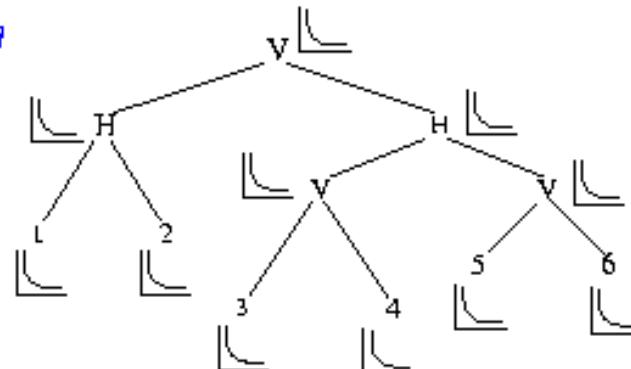
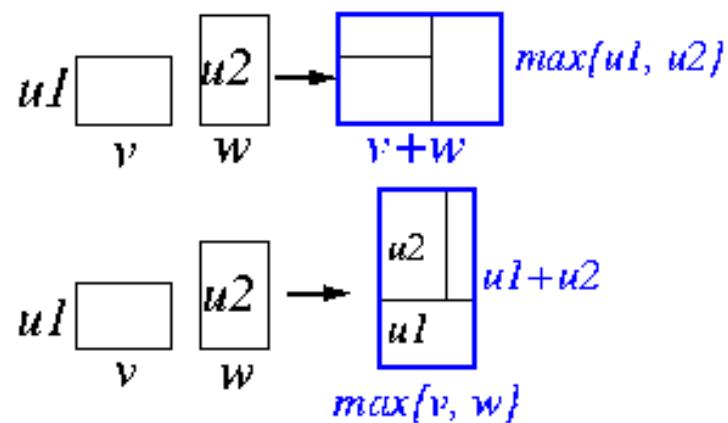
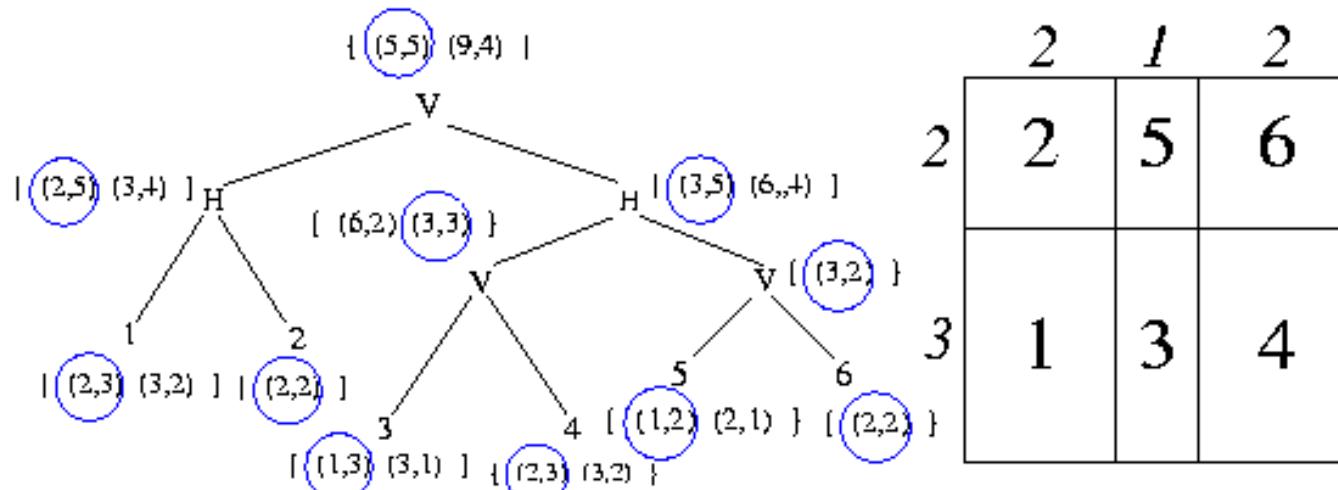
Deriving Shapes of Children

- A choice for the minimal shape of a **composite block** fixes the shapes of the shapes of its children blocks.



Area Computation for Hard Blocks Revisited

- Allow rotation

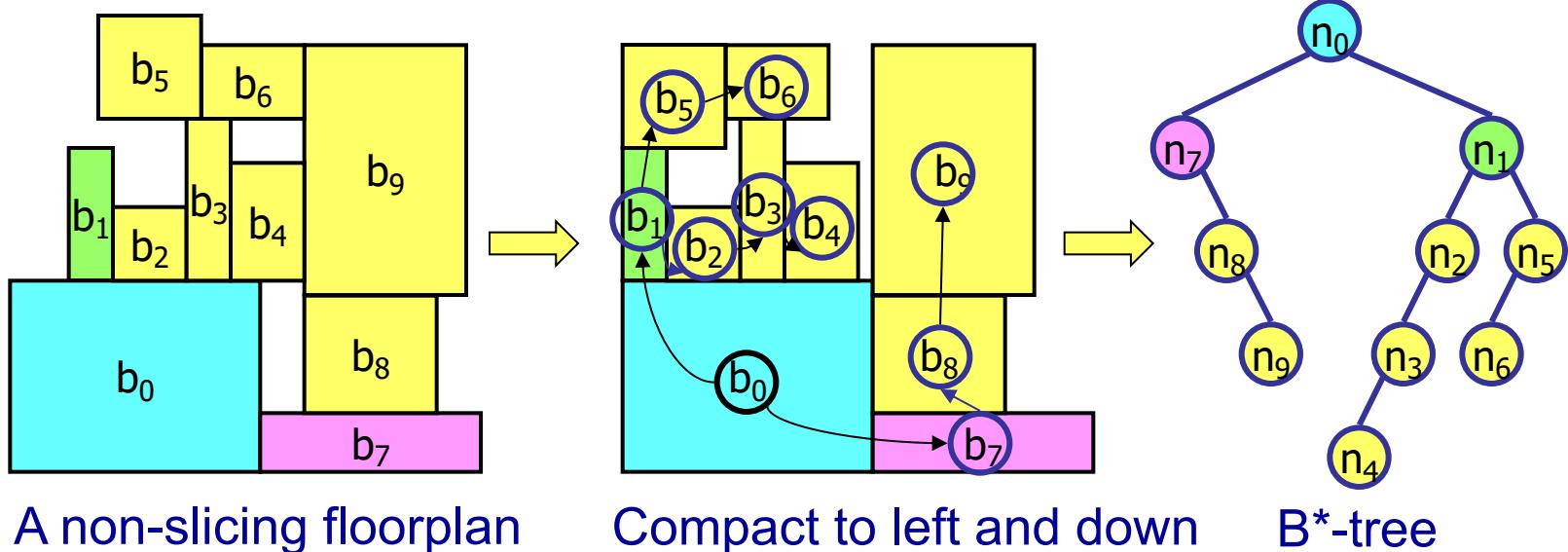


Slicing Floorplan Sizing

- . The shape functions of all **leaf blocks** are given as piecewise linear functions.
- . 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**).
- . The sizing algorithm runs in polynomial time for slicing floorplans
 - NP-complete for non-slicing floorplans

B*-Tree: Compacted Floorplan Representation

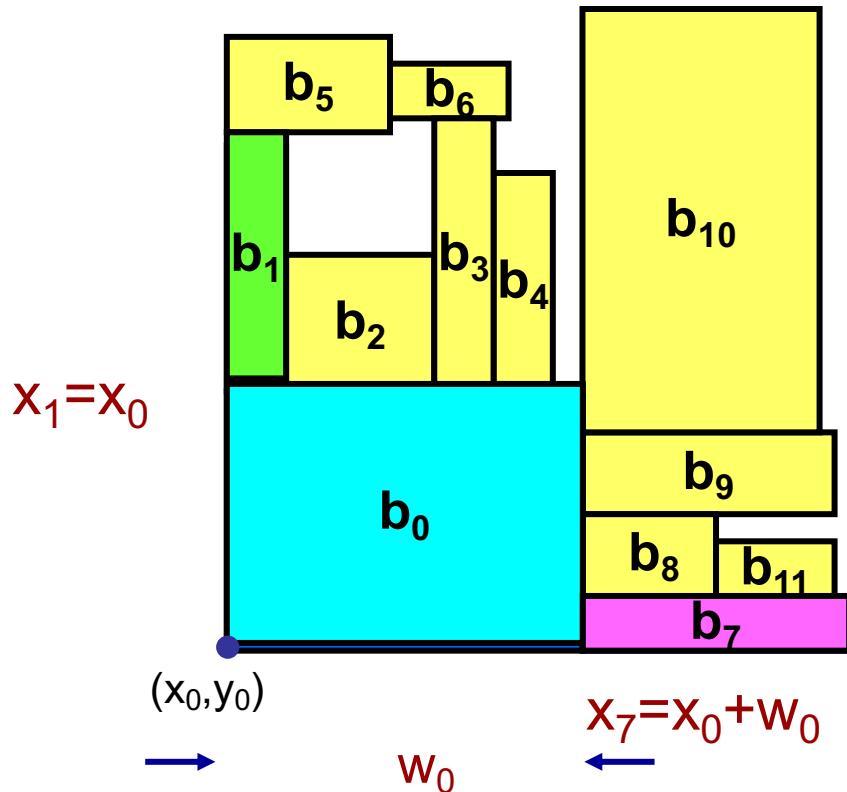
- Chang et. al., “B*-tree: A new representation for non-slicing floorplans,” DAC-2k.
將指定的floorplan向左下壓縮後，才能建立B*-tree
而最左下方的module就是B*-tree的root
 - Compact modules to left and bottom.
 - Construct an ordered binary tree (B*-tree).
 - > Left child: the lowest, adjacent block on the right ($x_j = x_i + w_i$).
 - > Right child: the first block above, with the same x-coordinate ($x_j = x_i$).



B*-tree Packing

將B* tree還原成block
的座標位置

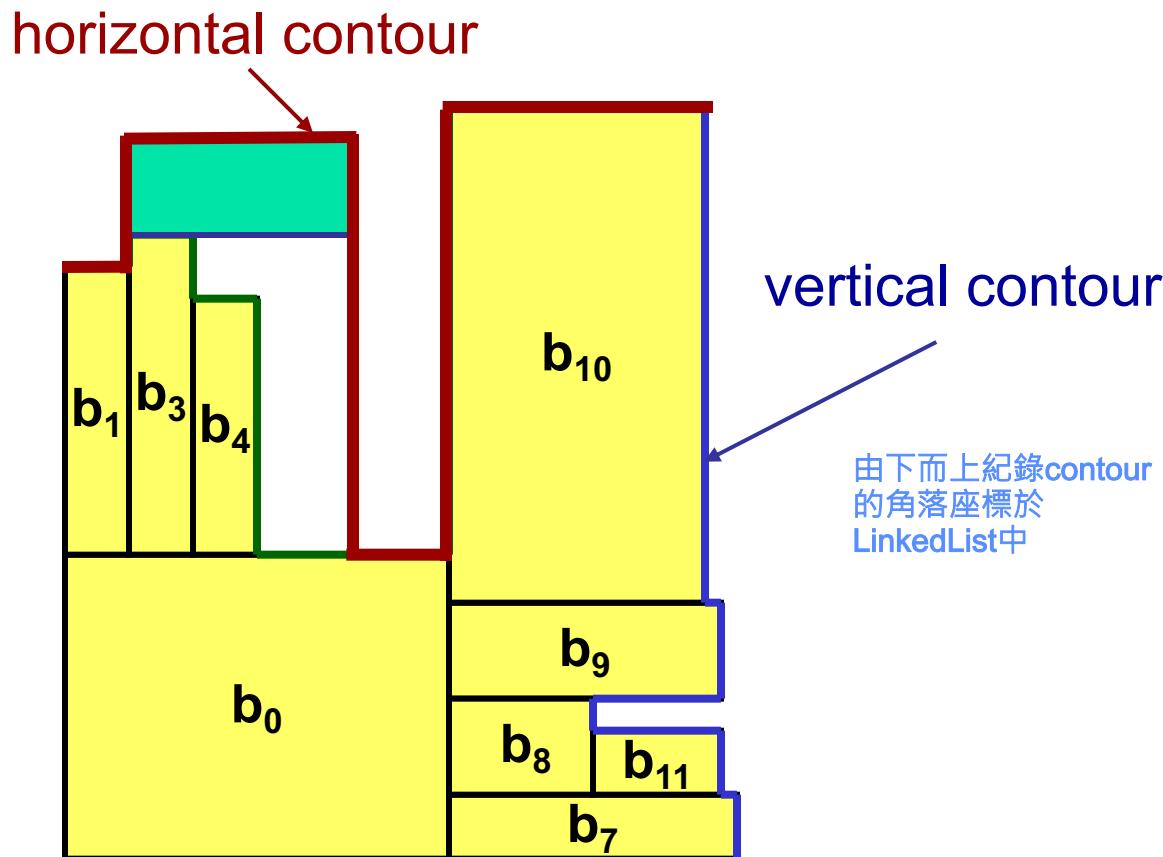
- x-coordinates can be determined by the tree structure.
 - Left child: the lowest, adjacent block on the right ($x_j = x_i + w_i$).
 - Right child: the first block above, with the same x-coordinate ($x_j = x_i$). 如果要利用B*-tree還原floorplan，則需要用pre-order traversal
- y-coordinates?



Computing y-coordinates

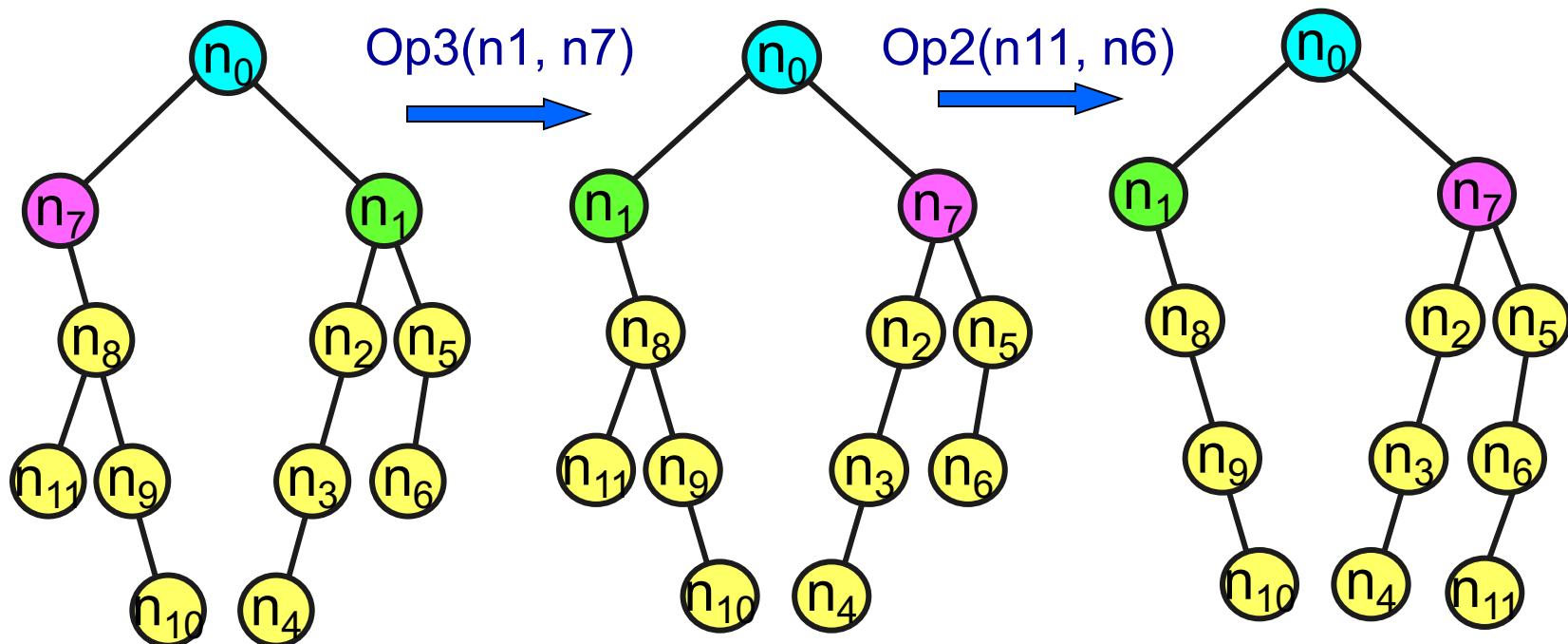
- Horizontal contour: Use a doubly linked list to record the current maximum y-coordinate for each x-range
- Reduce the complexity of computing a y-coordinate to amortized $O(1)$ time

紀錄從某個x座標到另一個x座標之間的y值為多少，可以利用LinkedList紀錄。如果加入了一個新的block，先決定新加入的block的x座標（可以根據B* tree該新block的parent來找），再算出新的block其頂端的y座標，然後更新被此新block覆蓋的y座標（被此新block覆蓋的區段要從linkedlist中刪除，然後insert此新的y值）



B*-Tree Perturbation

- Op1: rotate a macro 旋轉某個block
- Op2: delete & insert 將某個node移除後在其他位置insert，但是要注意每個node至多只能有兩個child，所以insert的時候要處理此情況
- Op3: swap 2 nodes
- Op4: resize a soft macro

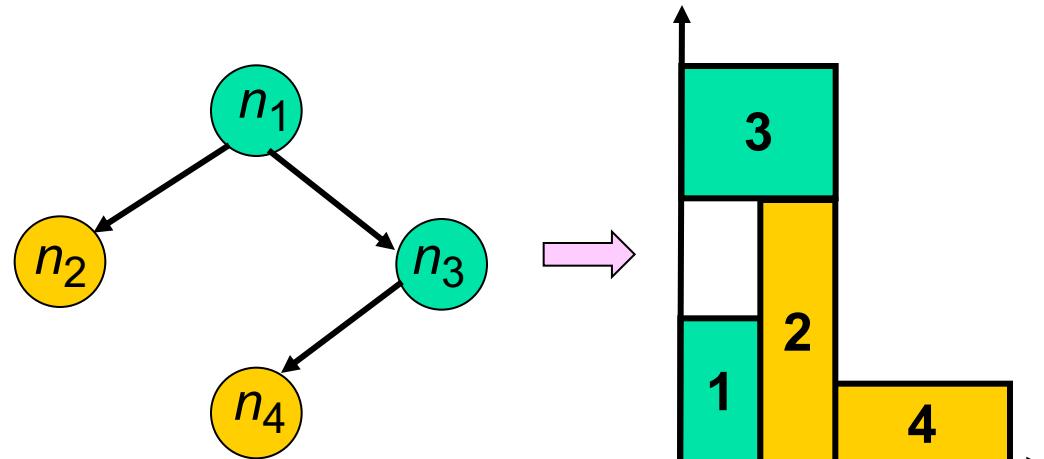


Strengths of B*-tree

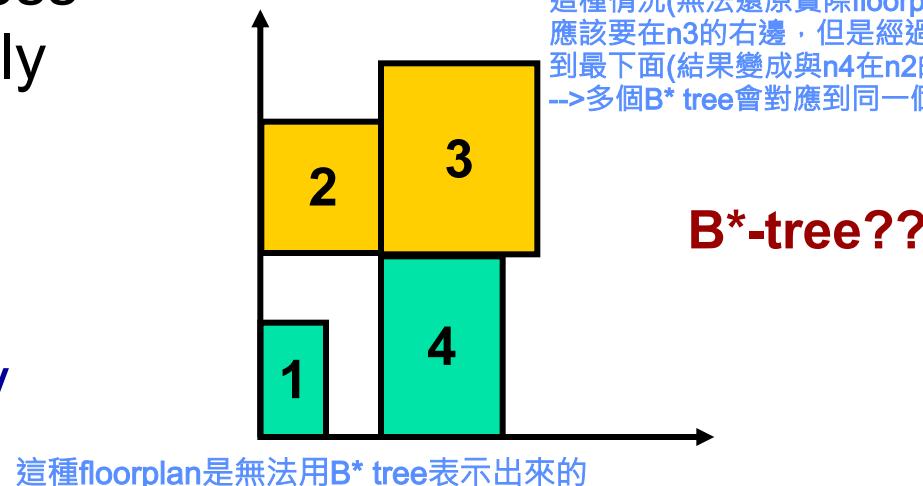
- Binary tree based, efficient and easy
- Flexible to deal with various placement constraints by augmenting the B*-tree data structure (e.g., preplaced, symmetry, alignment, bus position) and rectilinear modules
- Transformation between a tree and its placement takes only amortized linear time (vs. $O(n^2)$ or $O(n \lg \lg n)$ for sequence pair to be shown shortly)
- Operate on only one B*-tree (vs. two O-trees)
- Can evaluate area cost incrementally
只要計算每次更動的
地方的面積即可(難)
- Smaller solution space: only $O(n! 4^n/n^{1.5})$ combinations (vs. $O((n!)^2)$ for sequence pair)
- Directly corresponds to hierarchical and multilevel frameworks for large-scale floorplan designs
- Can be extended to 3D floorplanning & related applications

Limitations with B*-Trees (& Other Trees?)

- Representation may change after packing.
- Is only a partially topological representation; less flexible than a fully topological representation
 - B*-tree can represent only compacted placement



從tree還原成實際情況的時候，有時候會出現右邊這種情況(無法還原實際floorplan的樣子)， n_4 原本應該要在 n_3 的右邊，但是經過演算法還原後會掉到最下面(結果變成與 n_4 在 n_2 的left child一樣)
-->多個B*-tree會對應到同一個floorplan

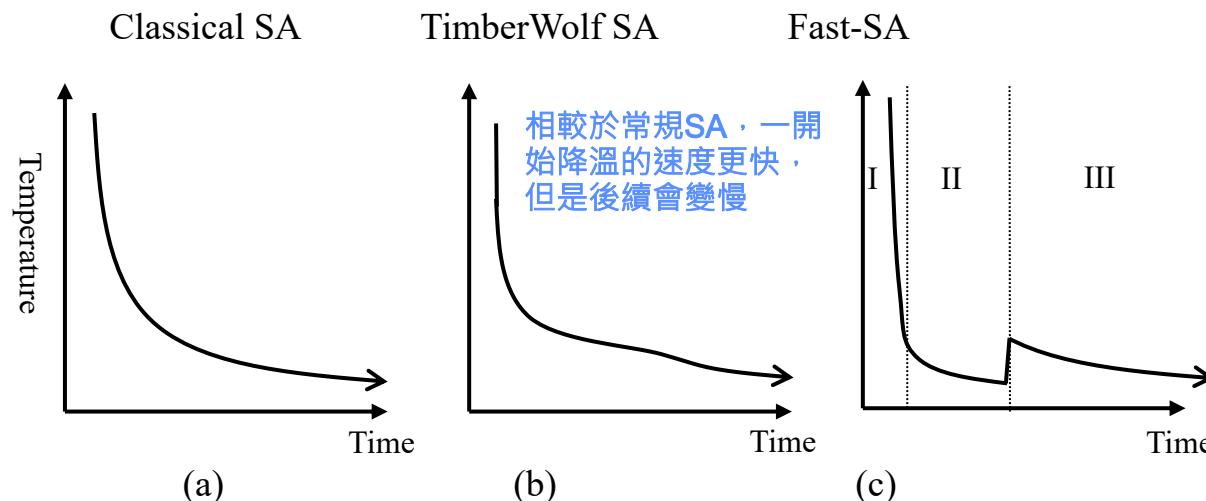


Simulated Annealing Schedules

- . Classical simulated annealing (SA)
 - Non-zero probability for up-hill move:
$$p = \min\{1, e^{-\Delta C/T}\}$$
 - Initial temperature: $T = |\Delta_{avg}| / \ln p$, p is the initial acceptance rate (typically, close to 1.0), Δ_{avg} is the average cost of the up-hill moves
 - Classical temperature updating function: λ is set to a fixed value (e.g., 0.85)
$$T_{new} = \lambda T_{old}, \quad 0 < \lambda < 1$$
- . TimberWolf annealing schedule (Sechen and Sangiovanni-Vincentelli, DAC-86)
 - Increase λ gradually from its lowest value (0.8) to its highest value (approximately 0.95) and then gradually decreases λ back to its lowest value.

Fast Simulated Annealing

- Chen and Chang, “Modern floorplanning based on fast simulated annealing,” ISPD-05 (TCAD-06)
- Comparisons for the temperature vs. search time:



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

Fast Simulated Annealing (2/2)

- Temperature update (T_1 : initial temperature)

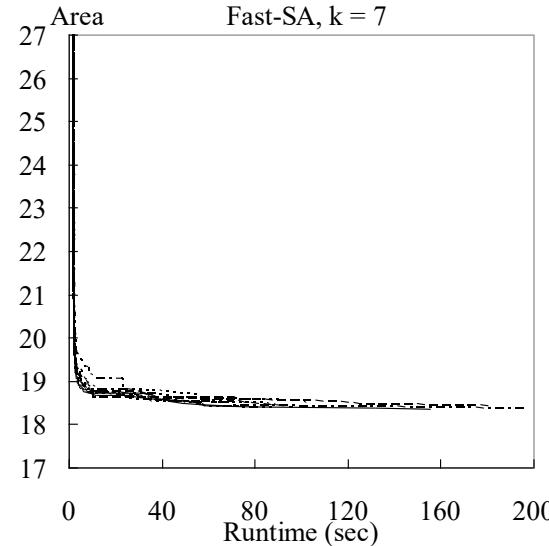
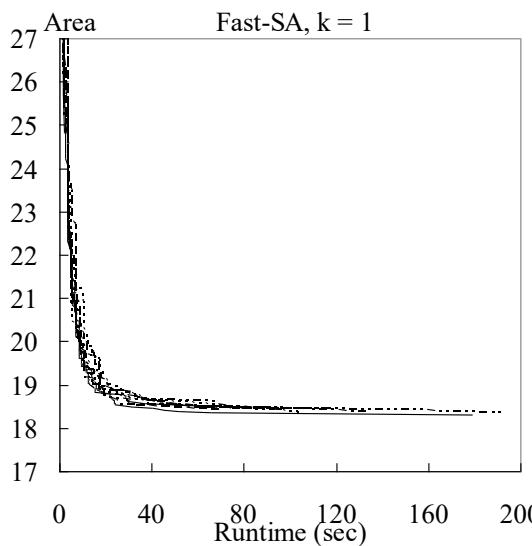
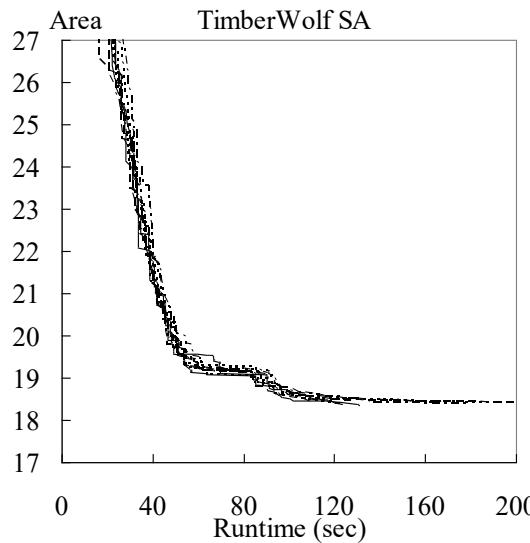
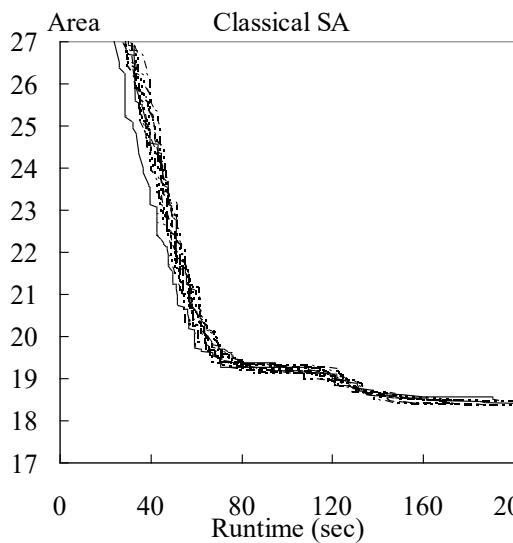
可以透過有方法的調整參數讓SA的收斂速度更快

$$T_r \square \begin{cases} \frac{\Delta_{avg}}{\ln p} & r \square 1 \\ \frac{T_1 \langle \Delta_{cost} \rangle}{rc} & 2 \leq r \leq k \\ \frac{T_1 \langle \Delta_{cost} \rangle}{r} & r \square k \end{cases}$$

Δ_{avg} Average uphill cost
 p Initial acceptance rate
 $\langle \Delta_{cost} \rangle$ Average cost change since the SA started
 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.

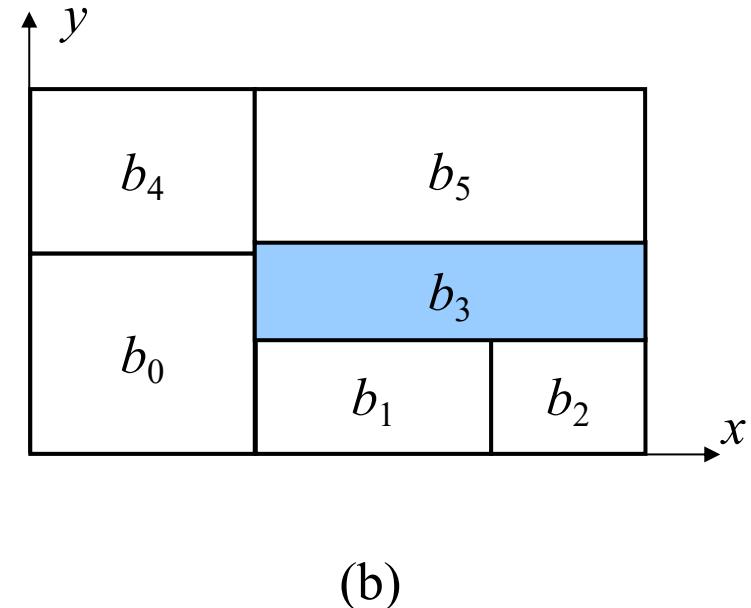
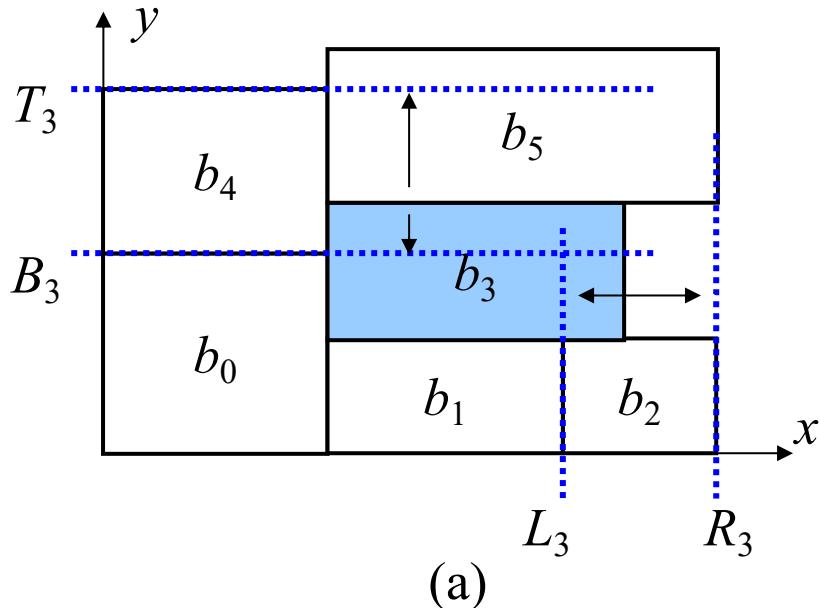
Convergence and Stability for Fast-SA (1/2)



- Classical SA
TimberWolf SA
Fast-SA, $k=1$ (no greedy local search)
Fast-SA, $k=7$
- Ran the circuit n100 for 10 times.
- Fast-SA has a better convergence speed than TimberWolf SA and classical SA.

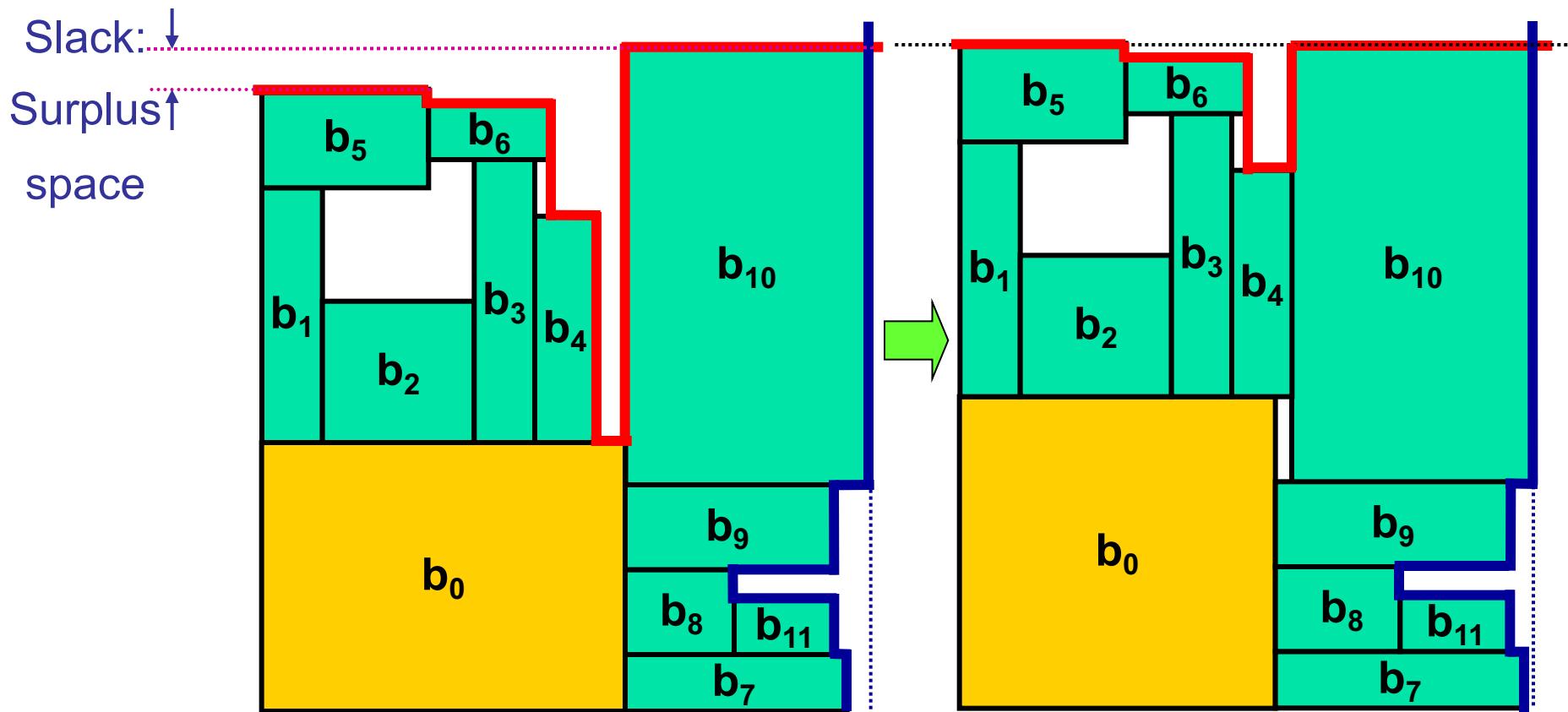
Simple Soft Block Sizing

- . Key: Line up a soft block with adjacent blocks
 - Each soft block has four candidates for the block dimension change to optimize the cost.
- . Advantage: fast and reasonably effective
- . Similar idea by Chi and Chen Chi, *Chung Yuan Journal*, 2003.



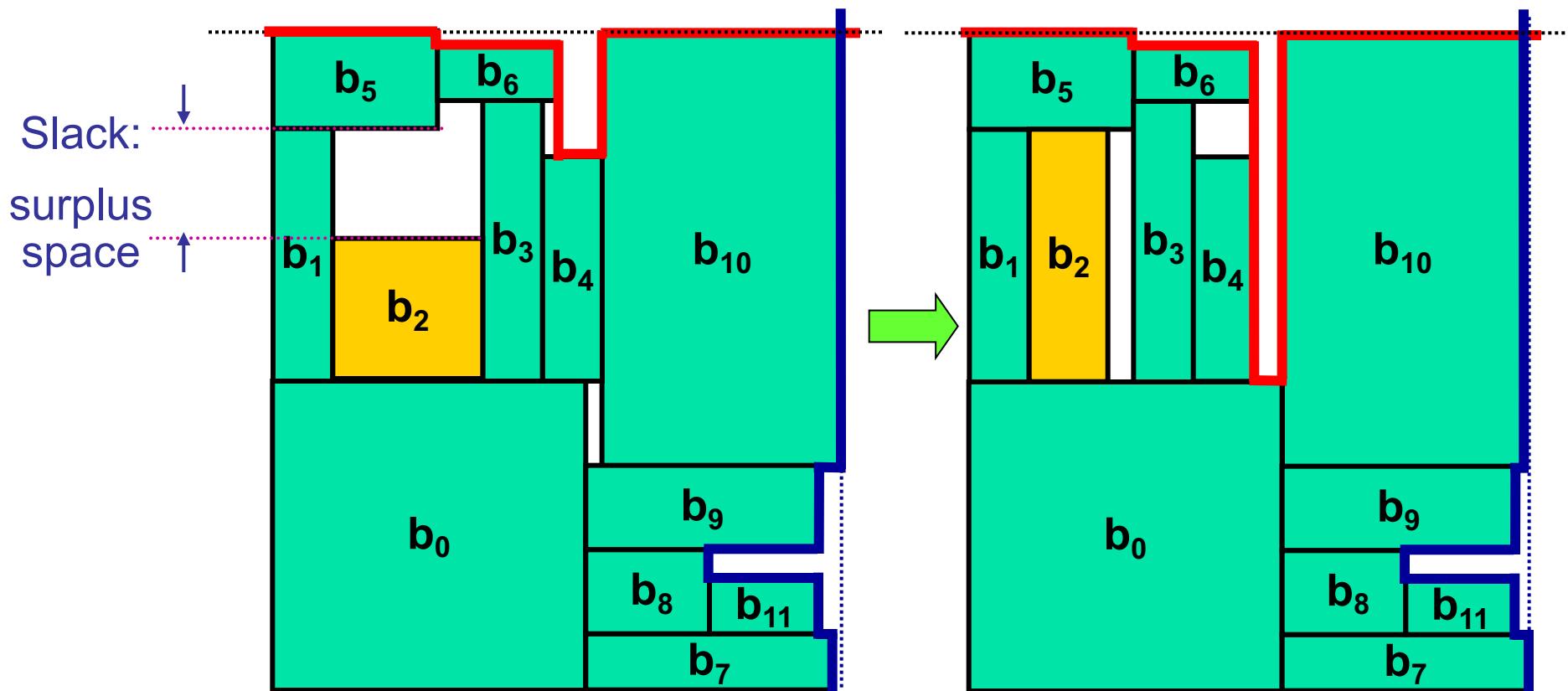
Slack-Based Sizing (1/4)

- Chang, et al., DAC-2K.
- Step1: Change the shape of the inserted soft block.
- Step2: Change the shapes of other soft blocks.

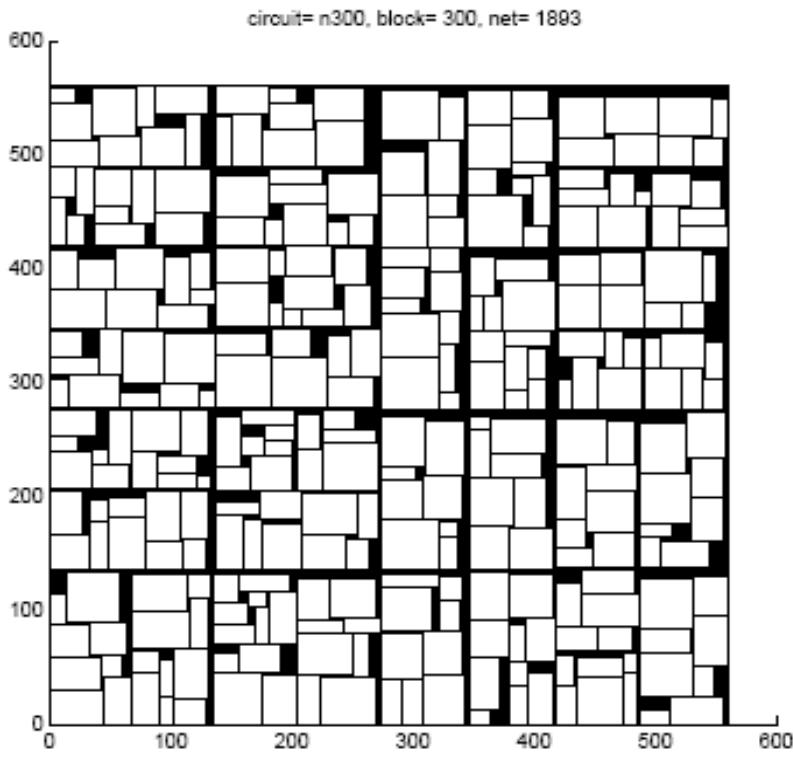


Slack-Based Sizing (2/4)

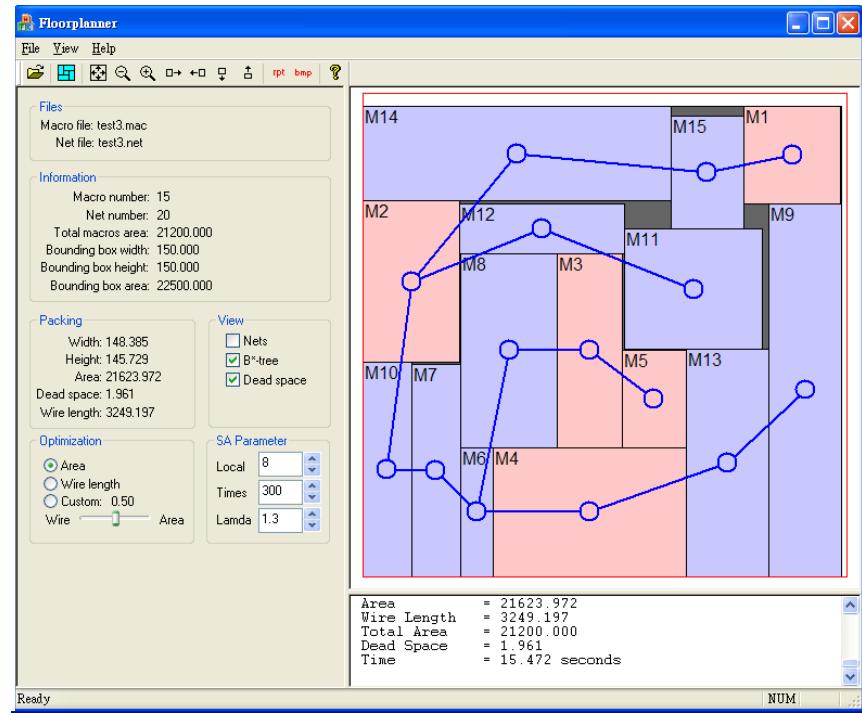
- . Step1: Change the shape of the inserted soft block
- . Step2: Change the shapes of other soft blocks



B*-tree Based Floorplanner



GSRC: n300 (300 modules)

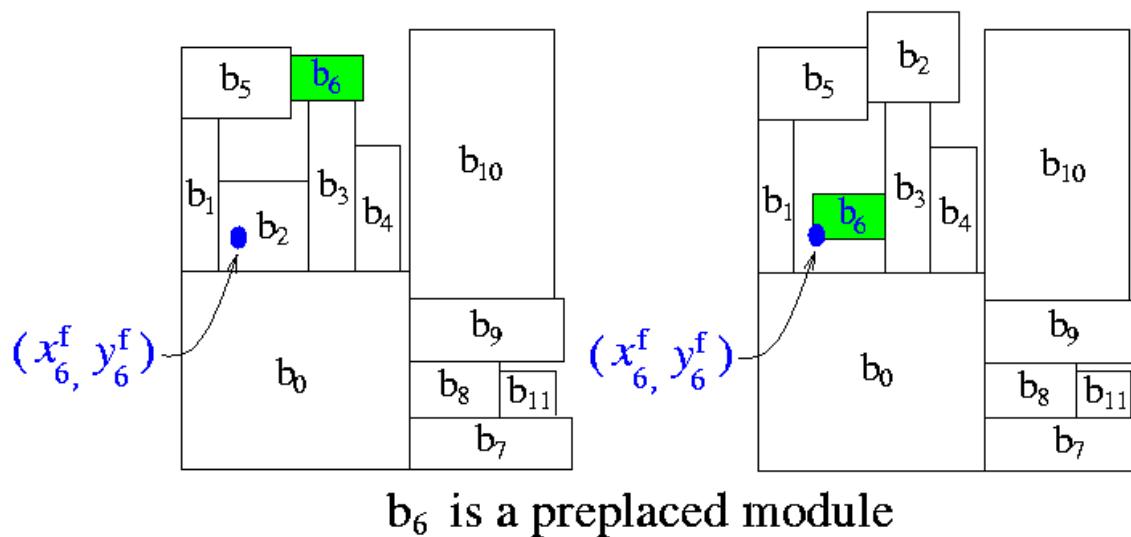


B*-tree floorplanning system

Courtesy of Tung-Chieh Chen

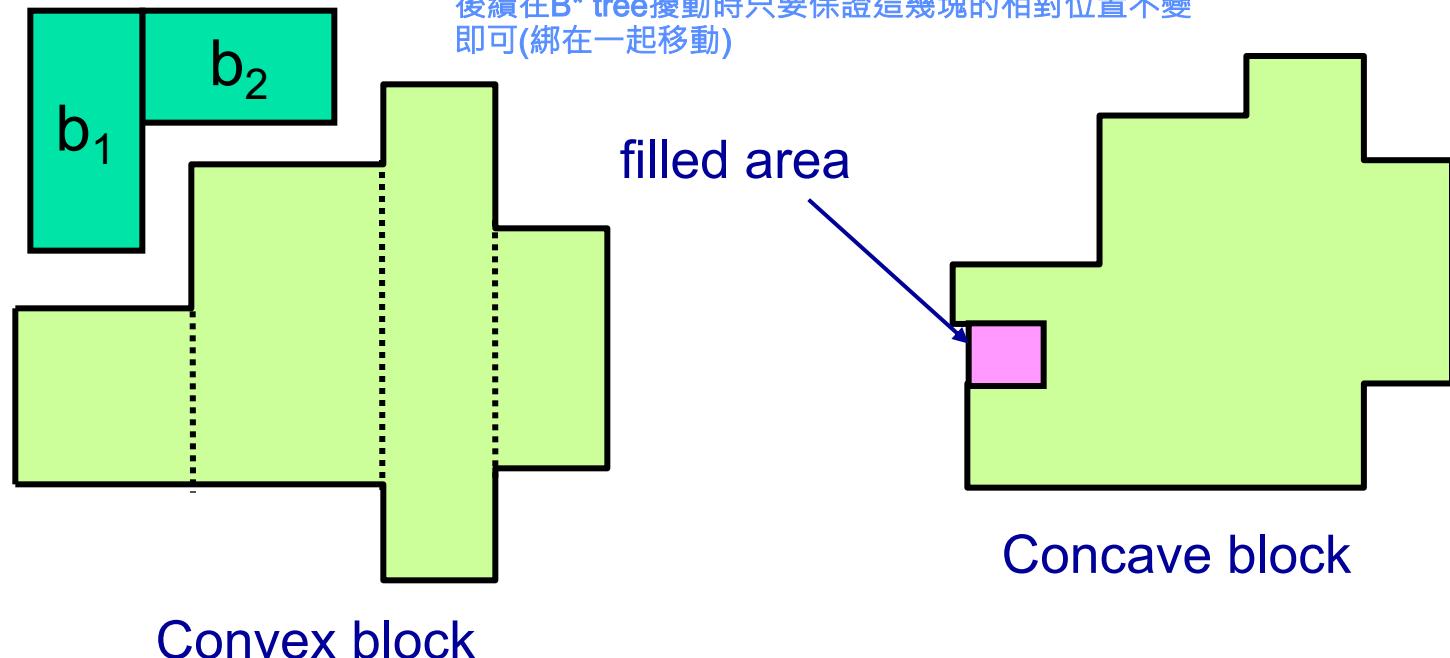
Coping with Pre-placed Modules

- If there are blocks ahead or lower than b_i so that b_i cannot be placed at its fixed position (x_i^f, y_i^f) , exchange b_i with the module in $D_i = \{b_j \mid (x_j, y_j) \leq (x_i^f, y_i^f)\}$ that is closest to (x_i^f, y_i^f) .
- Fix the placement for affected blocks (as in slack-based sizing)
一開始假裝沒看到 b_2 ，然後將 b_6 直接放進去想放的位置，然後去檢查有誰被 b_6 蓋住(發現 b_2 被蓋住)，然後就把被蓋住的障礙物移走



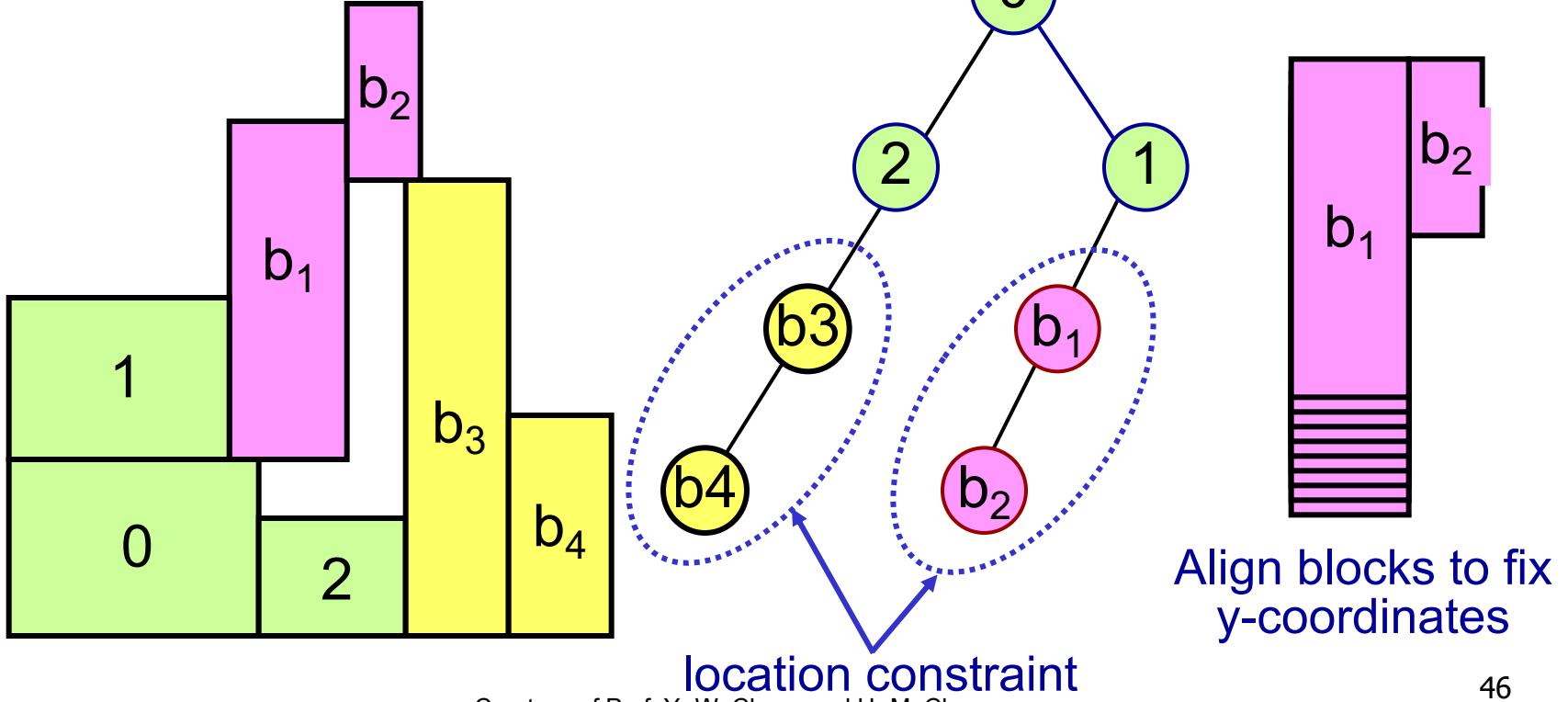
Coping with Rectilinear Blocks

- Wu, Chang, Chang, “Rectilinear block placement using B*-trees,” ACM TODAES, 2003 (ICCD-01)
- Convex block: partition a rectilinear block into rectangular sub-blocks
- Concave block: fill the concave holes of a concave block to make the block convex (other approximate schemes?)



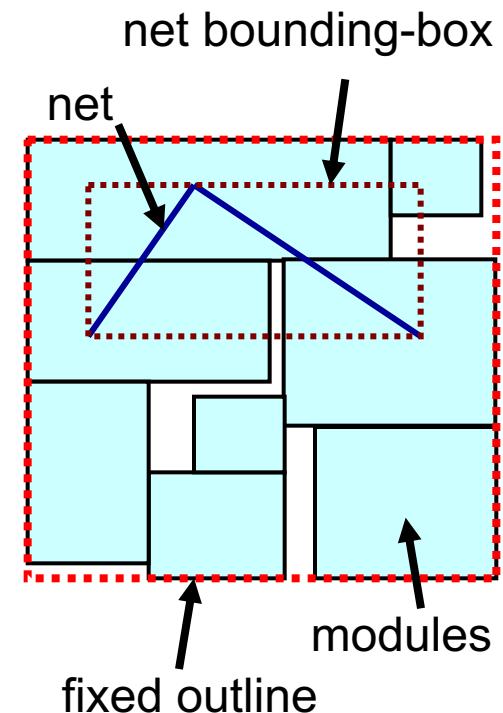
Location Constraint for Rectilinear Blocks

- Location Constraint: Keep b_2 as b_1 's left child in the tree.
- x-coordinates can be correctly determinated
 - y-coordinates? Align sub-blocks, if necessary.
- Treat the sub-blocks of a block as a whole during processing.



Fixed-Outline Floorplanning

- Chen and Chang, “Modern floorplanning based on fast simulated annealing,” ISPD-05 (TCAD-06)
- Fixed-outline floorplanning is more prevailing in modern VLSI design
- 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

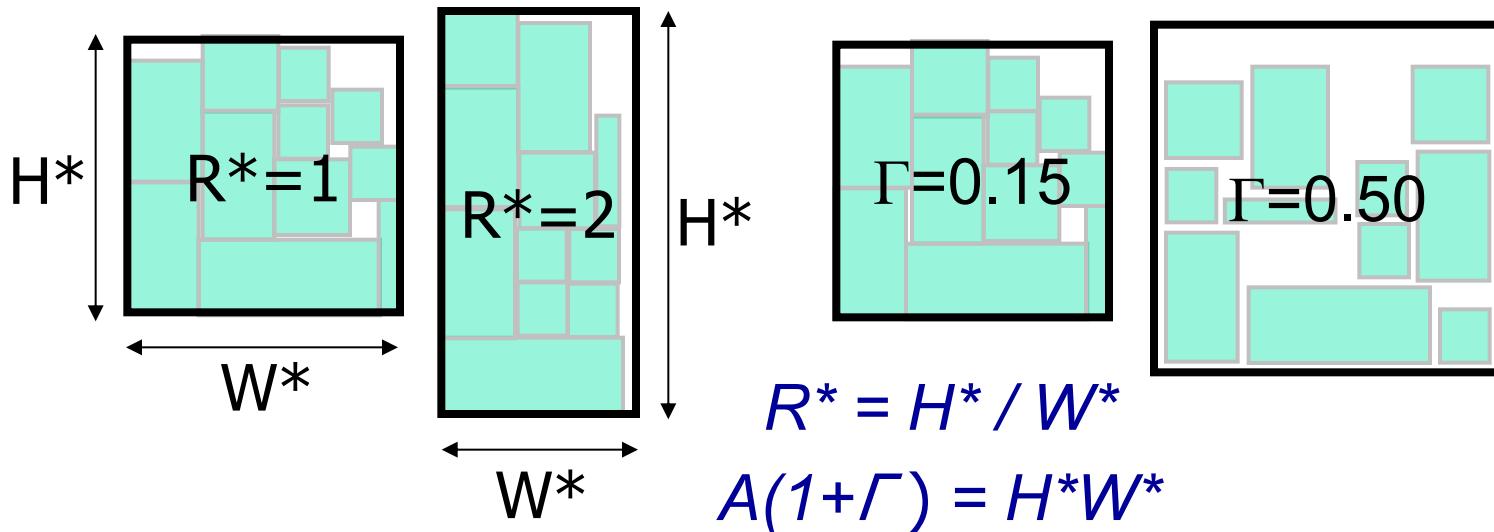


Fixed-Outline Constraints

- . Two user-specified parameters:
 - Γ : maximum white-space fraction, and
 - R^* : desired aspect ratio (height/width)
- . The outline (height H^* and width W^*) is defined by

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

- . Use the same formulation as Adya et al. (*ICCD-01*).



Cost Function for Fixed-Outline Floorplanning

- . Cost for a floorplan F

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

Chip area

Wirelength

Aspect ratio penalty

A Chip area

α Area weight

W Wirelength

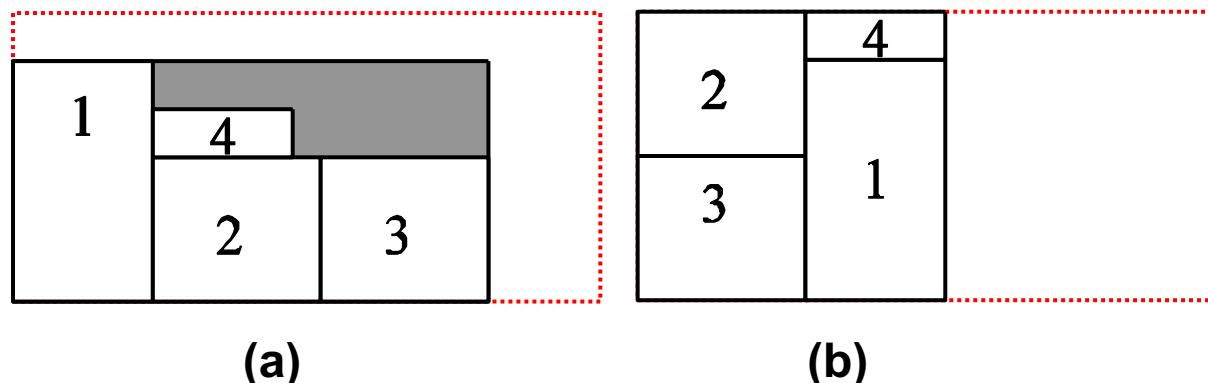
β Wirelength weight

R^* Desired aspect ratio

R Current floorplan aspect ratio

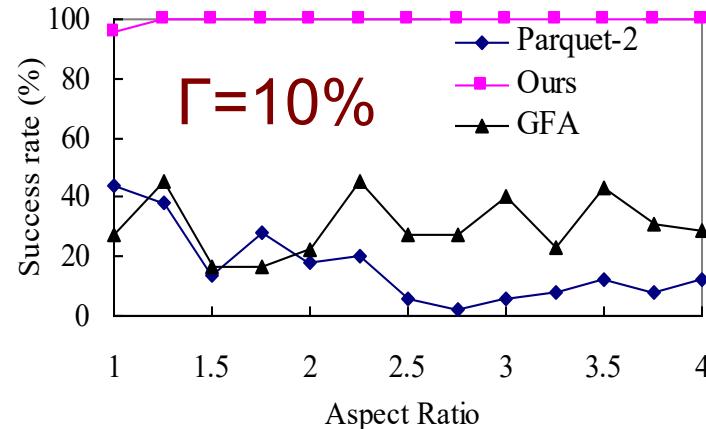
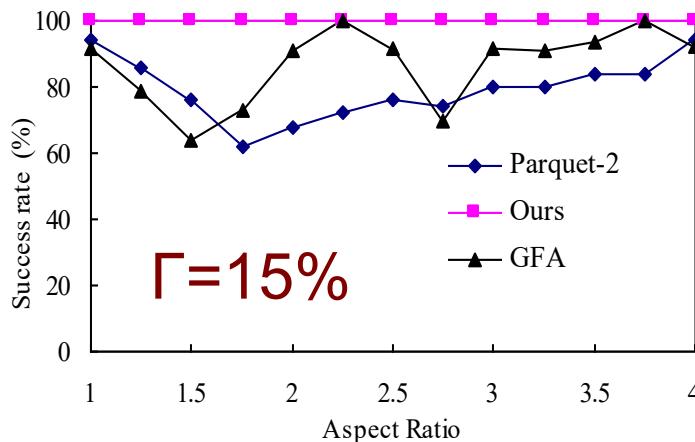
Adaptive Simulated Annealing

- . The aspect ratio of the best floorplan area in the fixed outline is not the same as that of the outline.
- . Shall decrease the weight of aspect ratio penalty $(1 - \alpha\beta)$ to concentrate more on the floorplan wirelength/area optimization (i.e., increase α and β).
 - 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 Floorplanning (1/2)

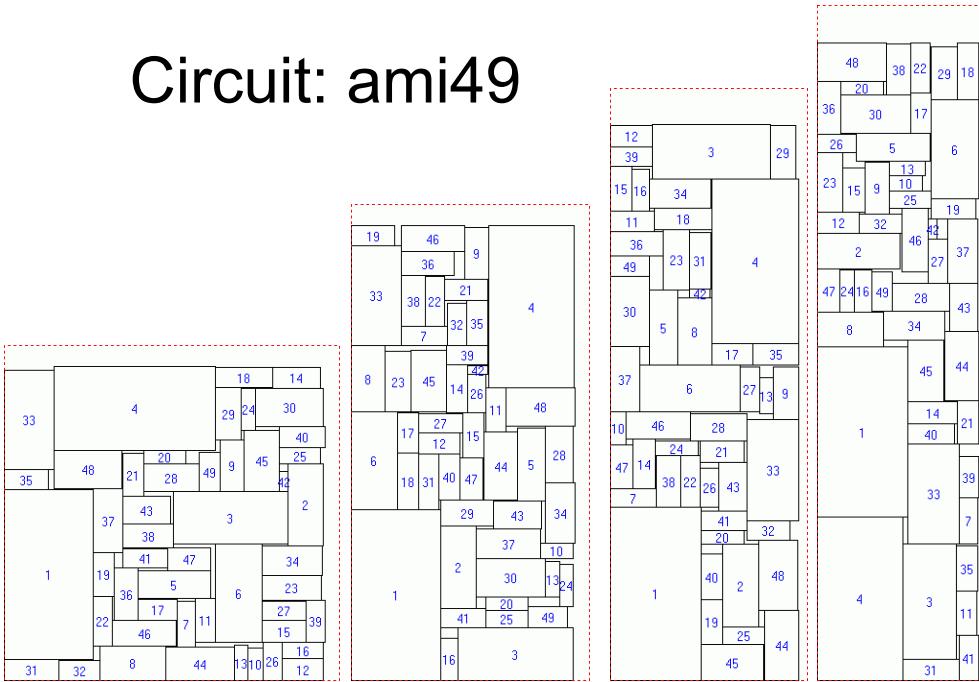
- Success probability vs. aspect ratio on circuit n100



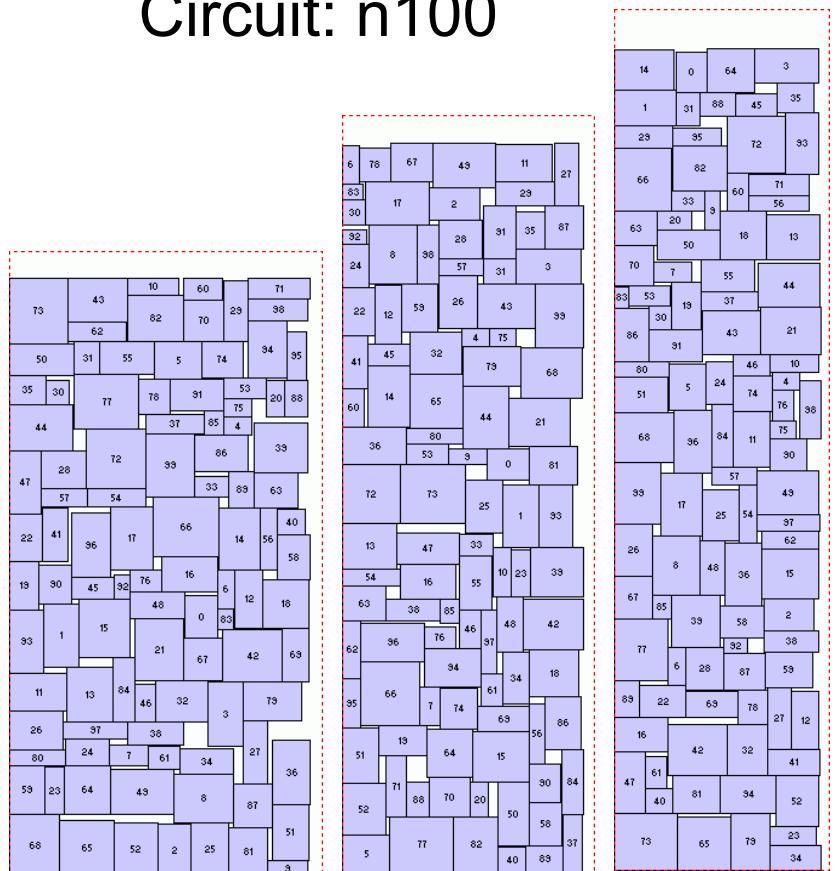
n100, $\Gamma=10\%$	Parquet-2: SP (TVLSI-2003)	GFA: NPE (ASPDAC-04)	Fast-SA: B*-tree (ISPD-05)
Avg. success rate	16.6%	30.3%	99.7%
Avg. dead space	7.32%	6.26%	5.79%
Avg. dead space ratio	1.26	1.08	1.00
Avg. runtime (sec)	40.2	44.5	27.6
Avg. runtime ratio	1.46	1.61	1.00

B*-tree Fixed-Outline Floorplanning Results

Circuit: ami49

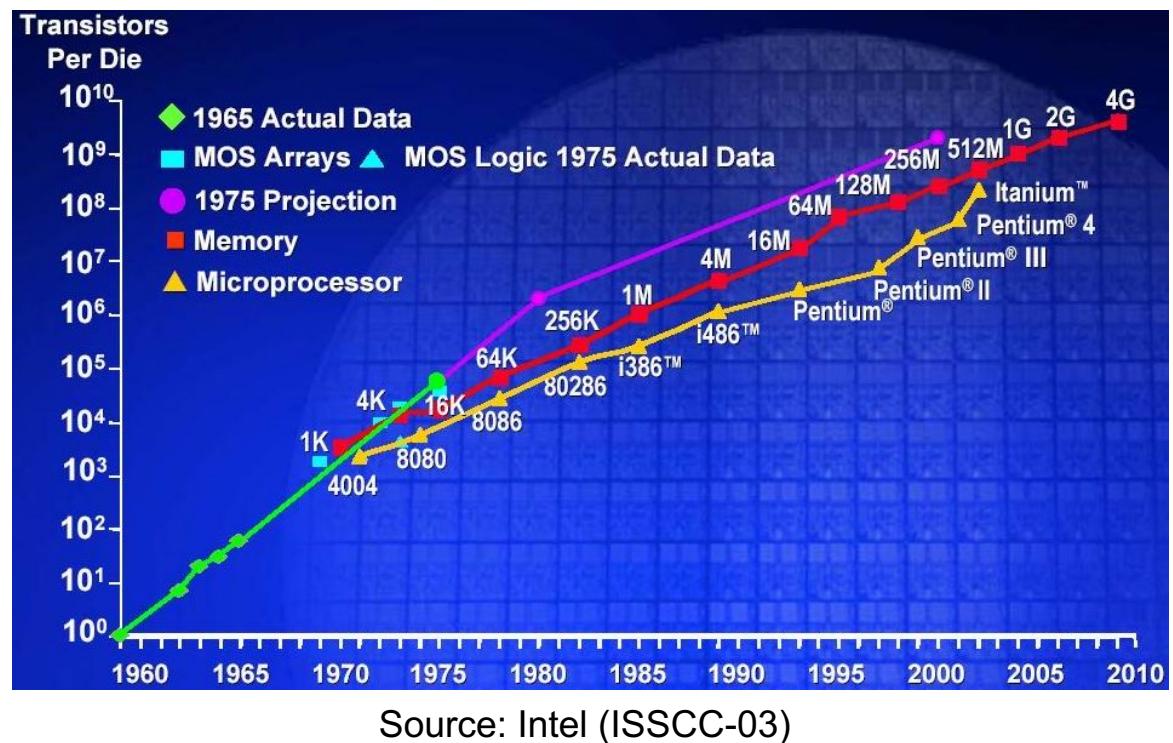
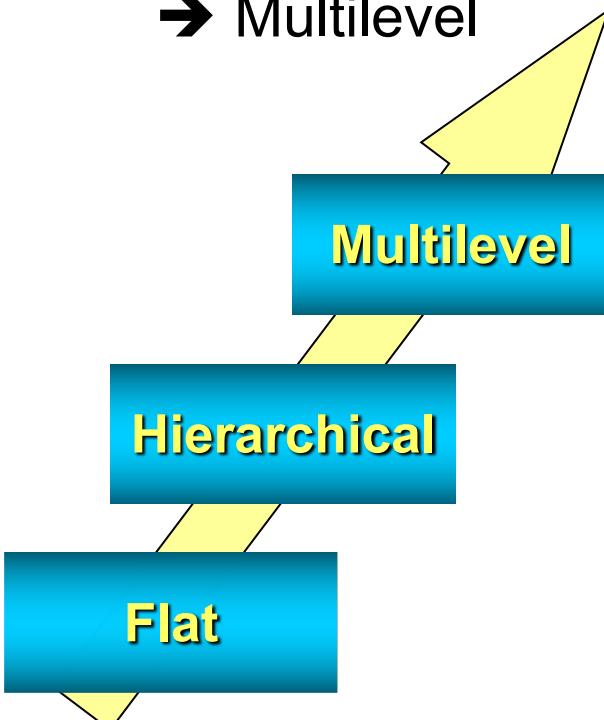


Circuit: n100



Framework Evolution Revisited

- . Billions of transistors may be fabricated in a single chip for nanometer technology.
- . Need tools for very large-scale designs.
- . Framework evolution for CAD tools: Flat → Hierarchical → Multilevel



Source: Intel (ISSCC-03)

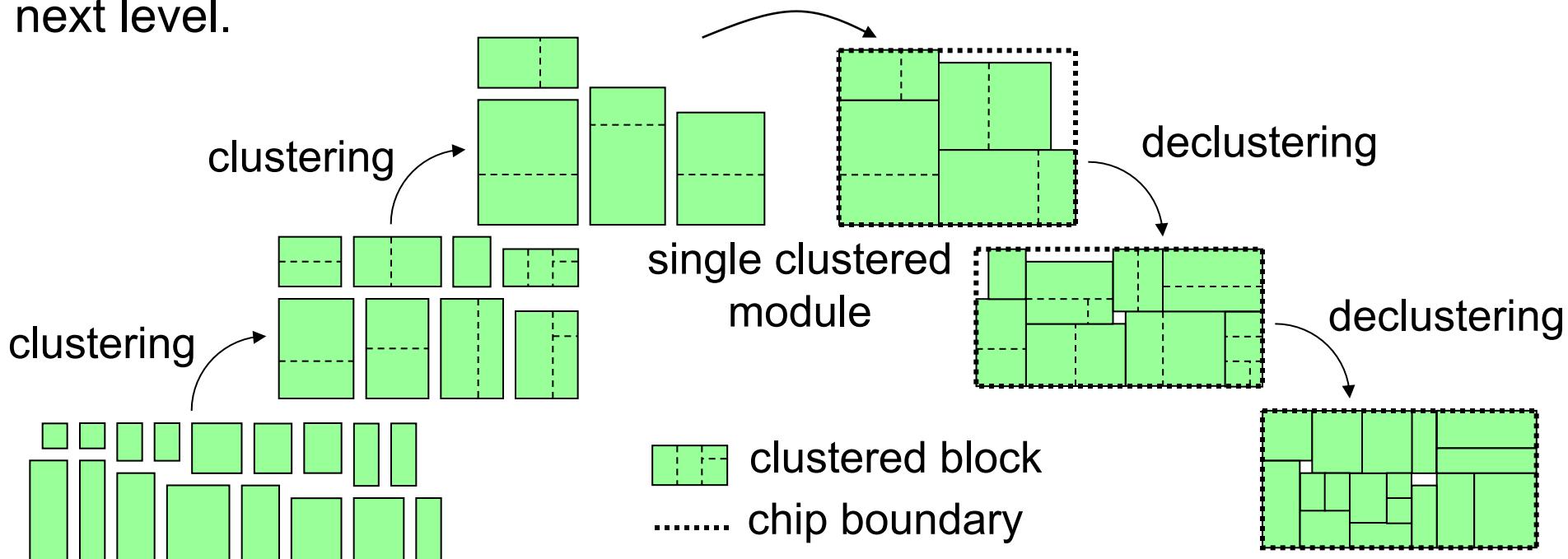
Multilevel B*-trees for Large-Scale Designs

- Lee, Hsu, Chang, Yang, “Multilevel floorplanning/placement for large-scale modules using B*-trees,” DAC-03 (TCAD-07)
- Two stages for MB*-tree: clustering followed by declustering
- 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

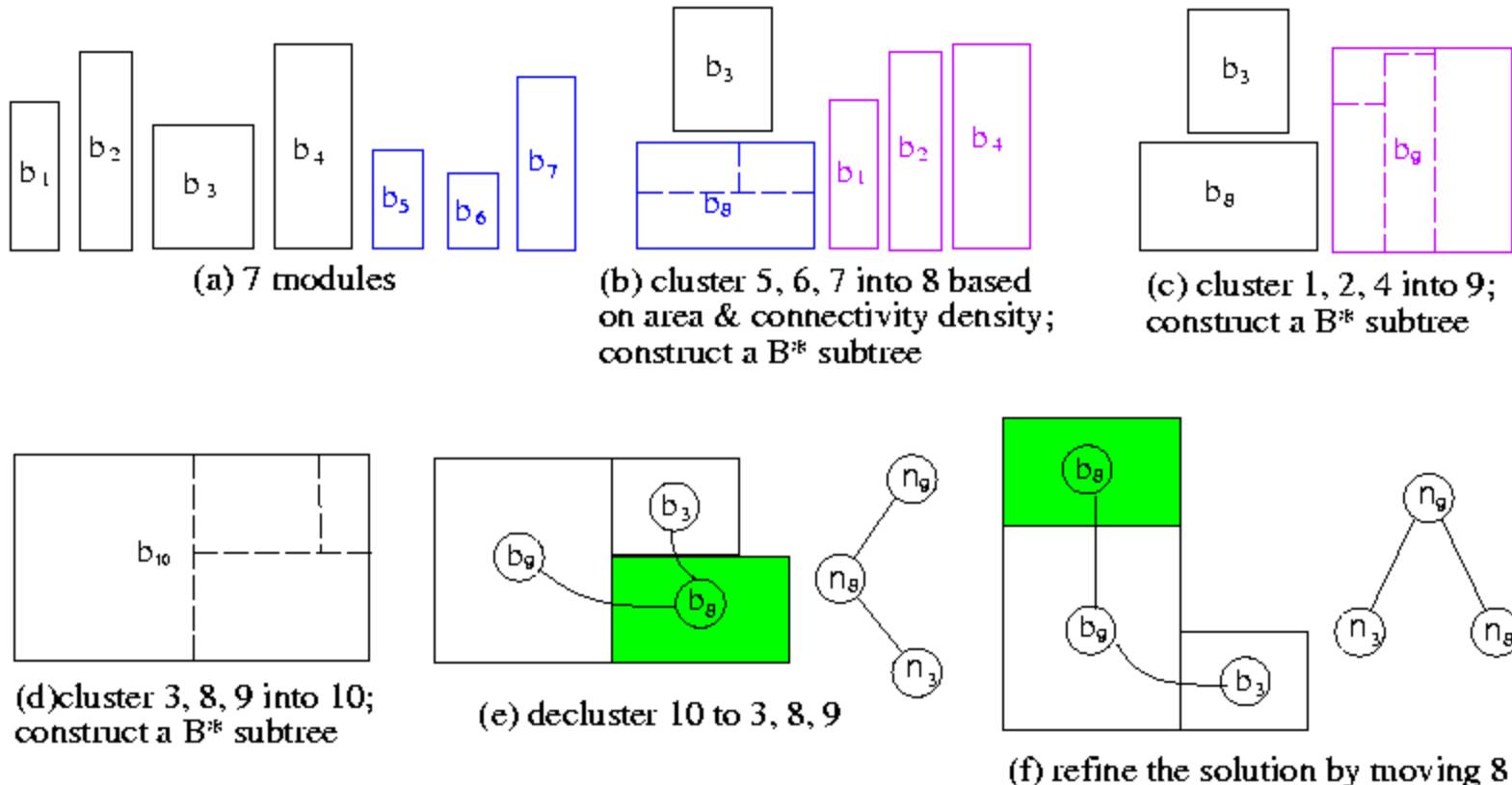
MB*-tree: Λ-Shaped Multilevel Floorplanning

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

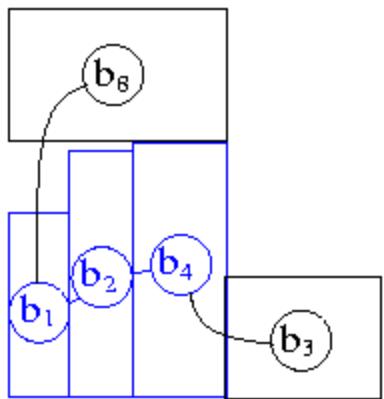
Recursively decluster the clusters and use simulated annealing to refine the floorplan.



Multilevel B*-tree Example

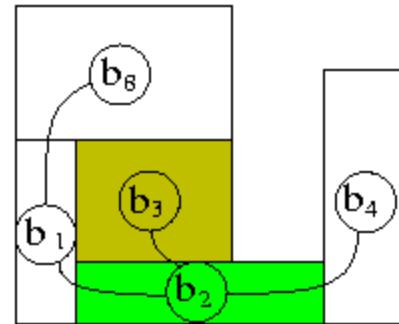
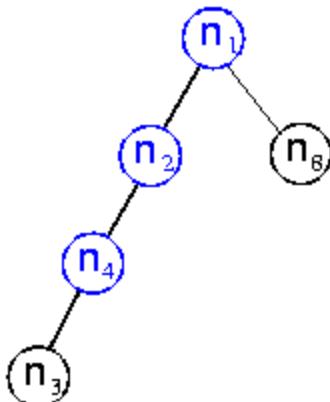


Multilevel B*-tree Example (cont'd)

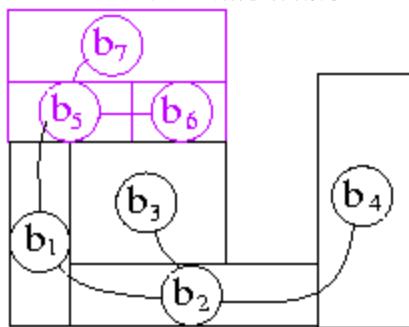


(g) decluster 9 to 1, 2, 4

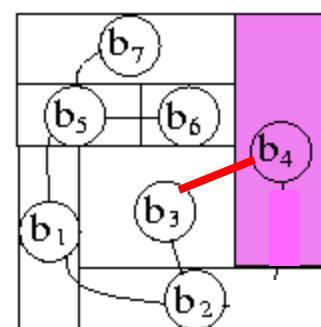
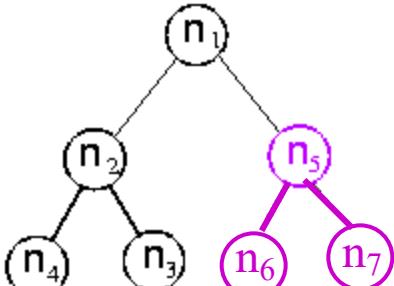
較晚被cluster的先展
開 · 然後微調



(h) refine the solution by moving 2, 3

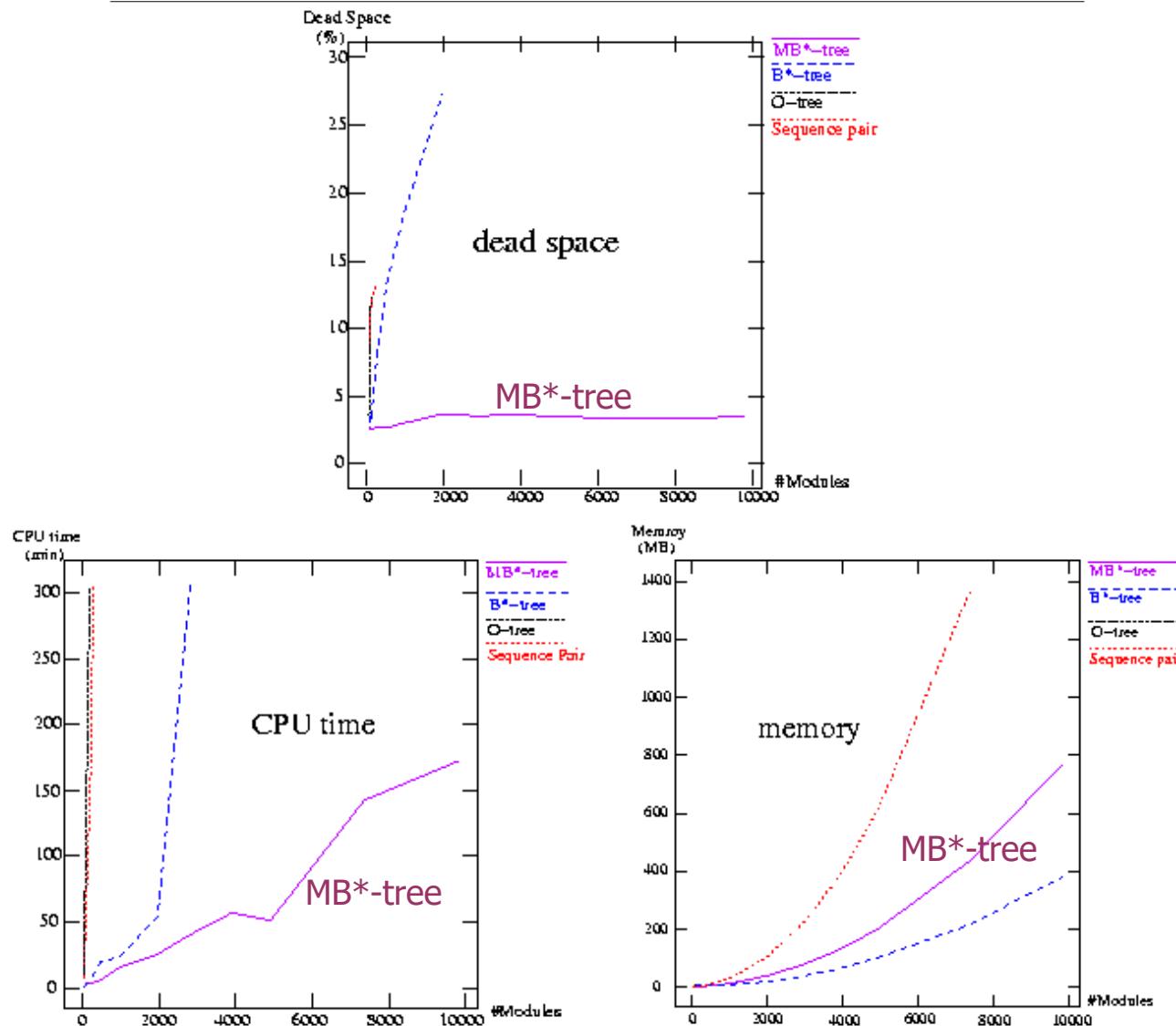


(i) decluster 8 to 5, 6, 7



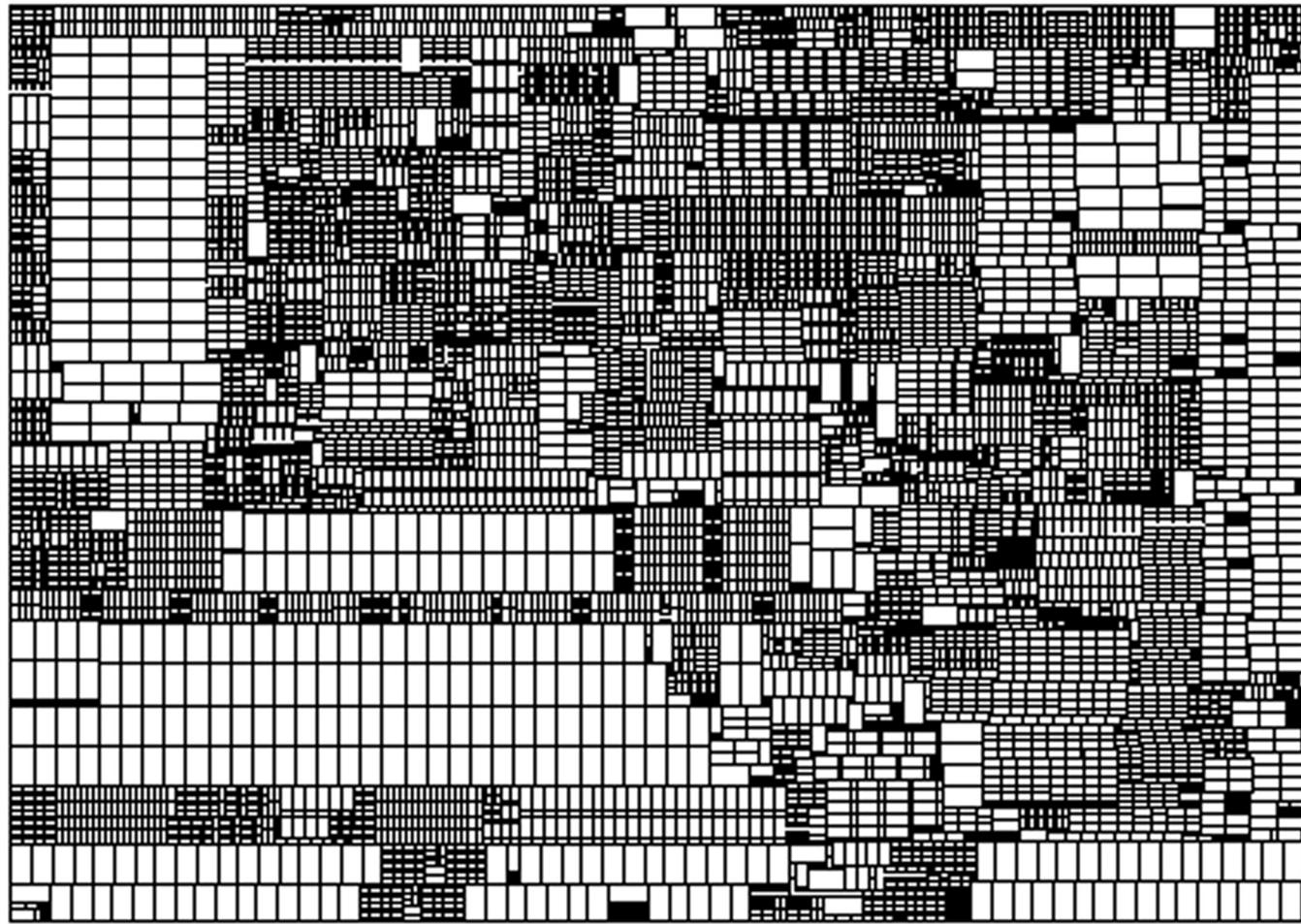
(j) refine the solution by moving 4

Comparison among Representations



Layout of ami49_200

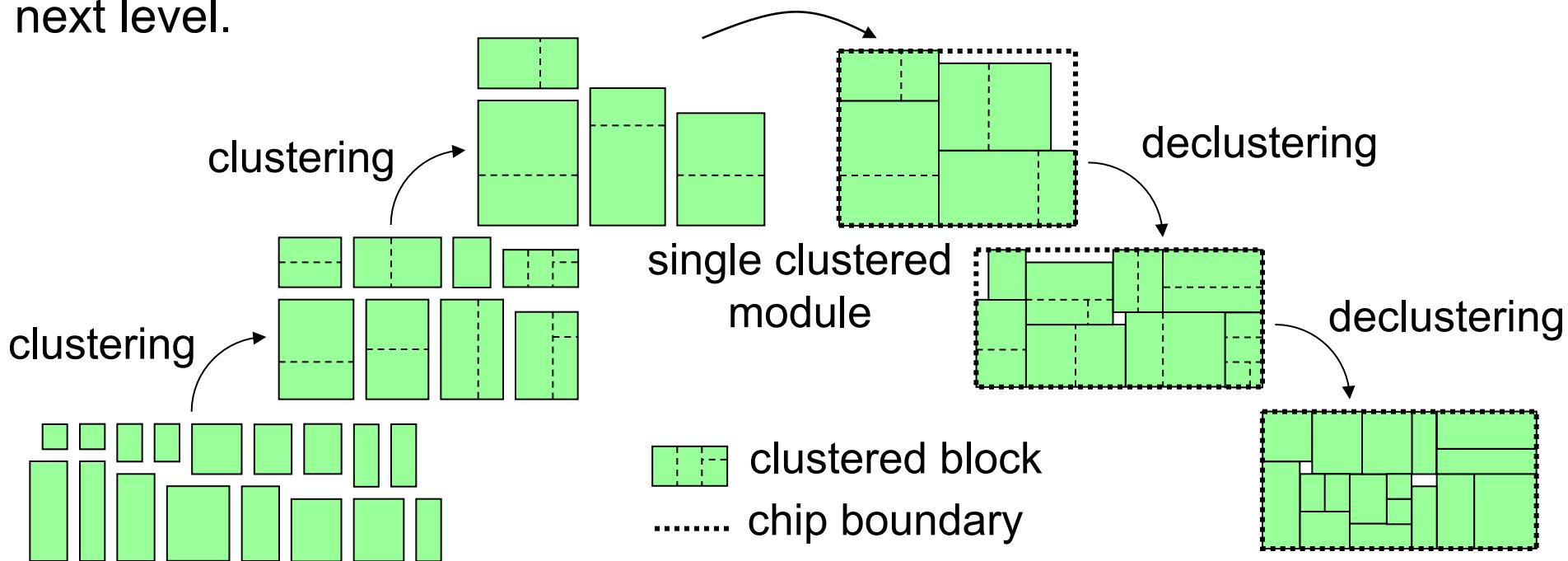
- MB*-tree: 9800 modules, dead space = 3.44%, CPU time = 256 min.



MB*-tree Multilevel Floorplanning Revisited

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

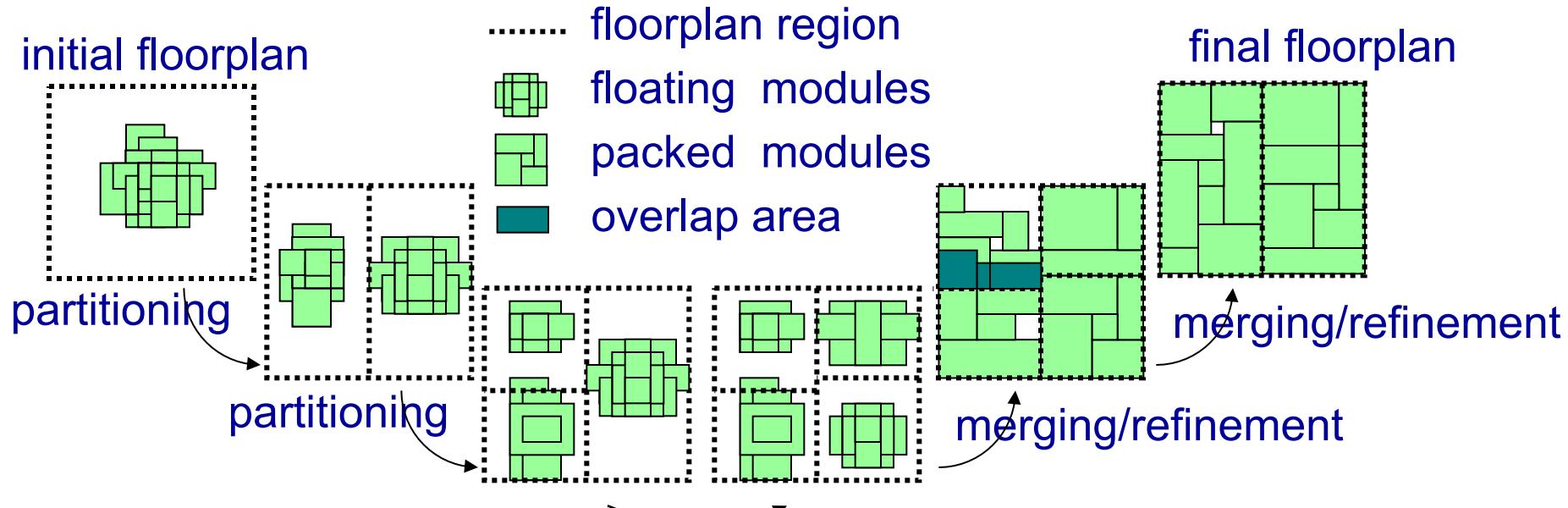
Recursively decluster the clusters and use simulated annealing to refine the floorplan.



Drawback: Does not have the view of the global configuration at the earlier stages

IMF: V-Shaped Multilevel Floorplanning

- Chen, Chang, Lin, “IMF: Interconnect-driven floorplanning for large-scale building-module designs,” ICCAD-05 (TCAD-06)



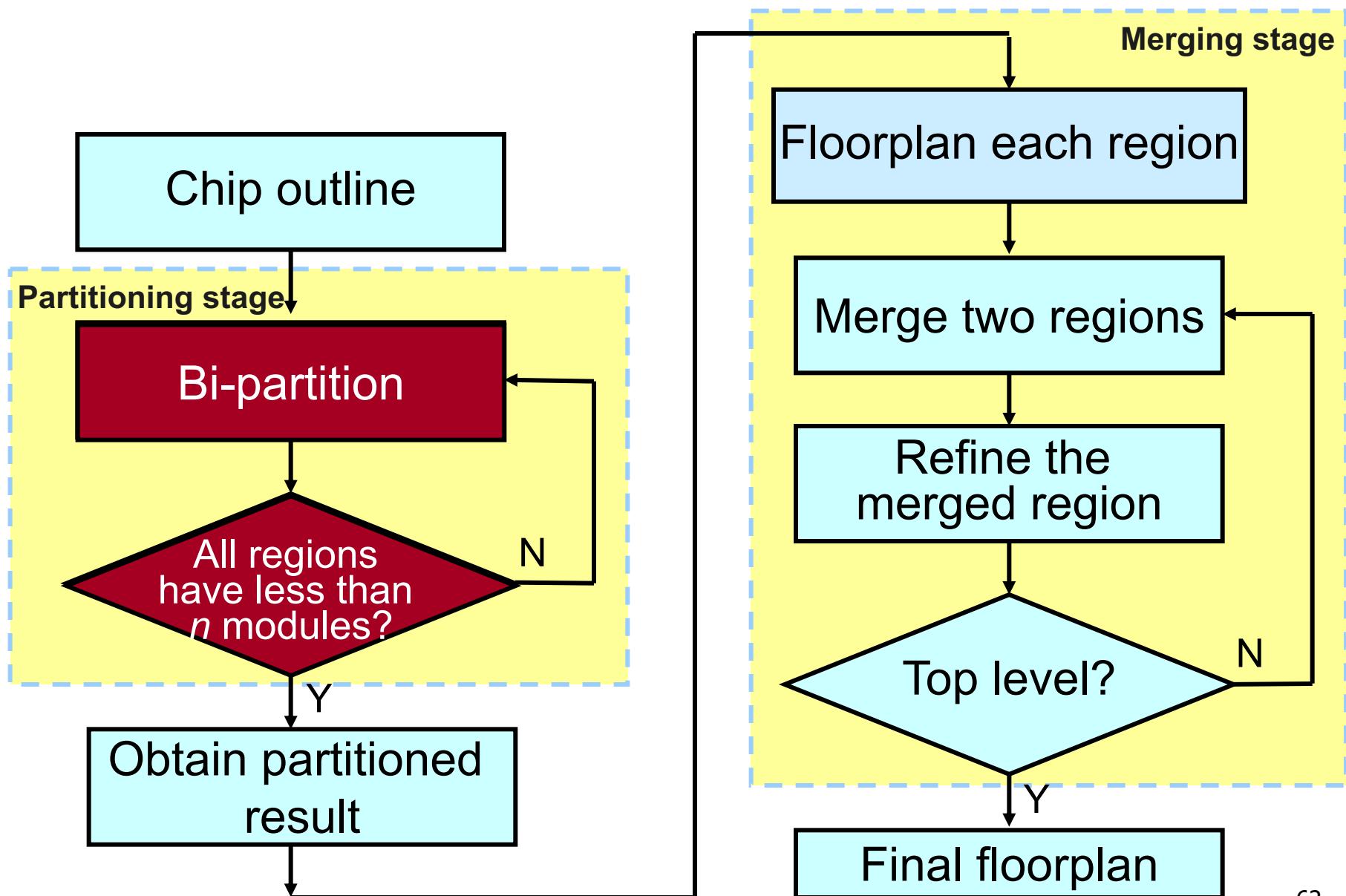
Perform partitioning to the circuit and determine the global locations of modules for the next level.

partitioned floorplan

Use the flat floorplanner to pack the modules in the partitions and legalize/refine the solution.

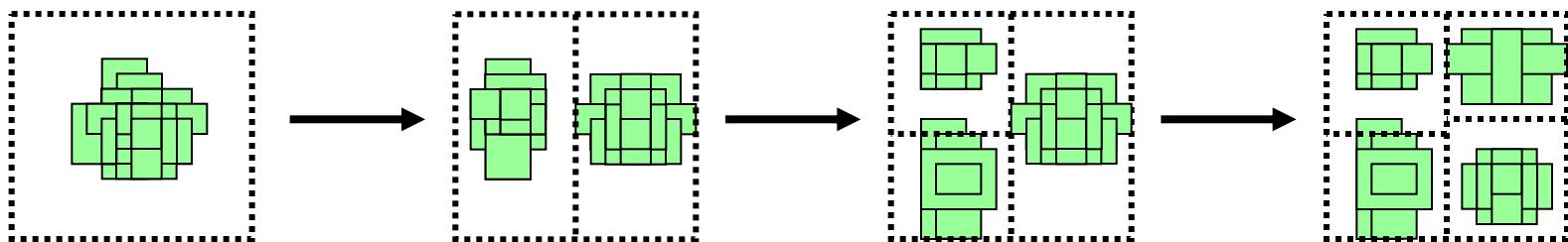
Consider the global interconnect at the earlier stages

Design Flow



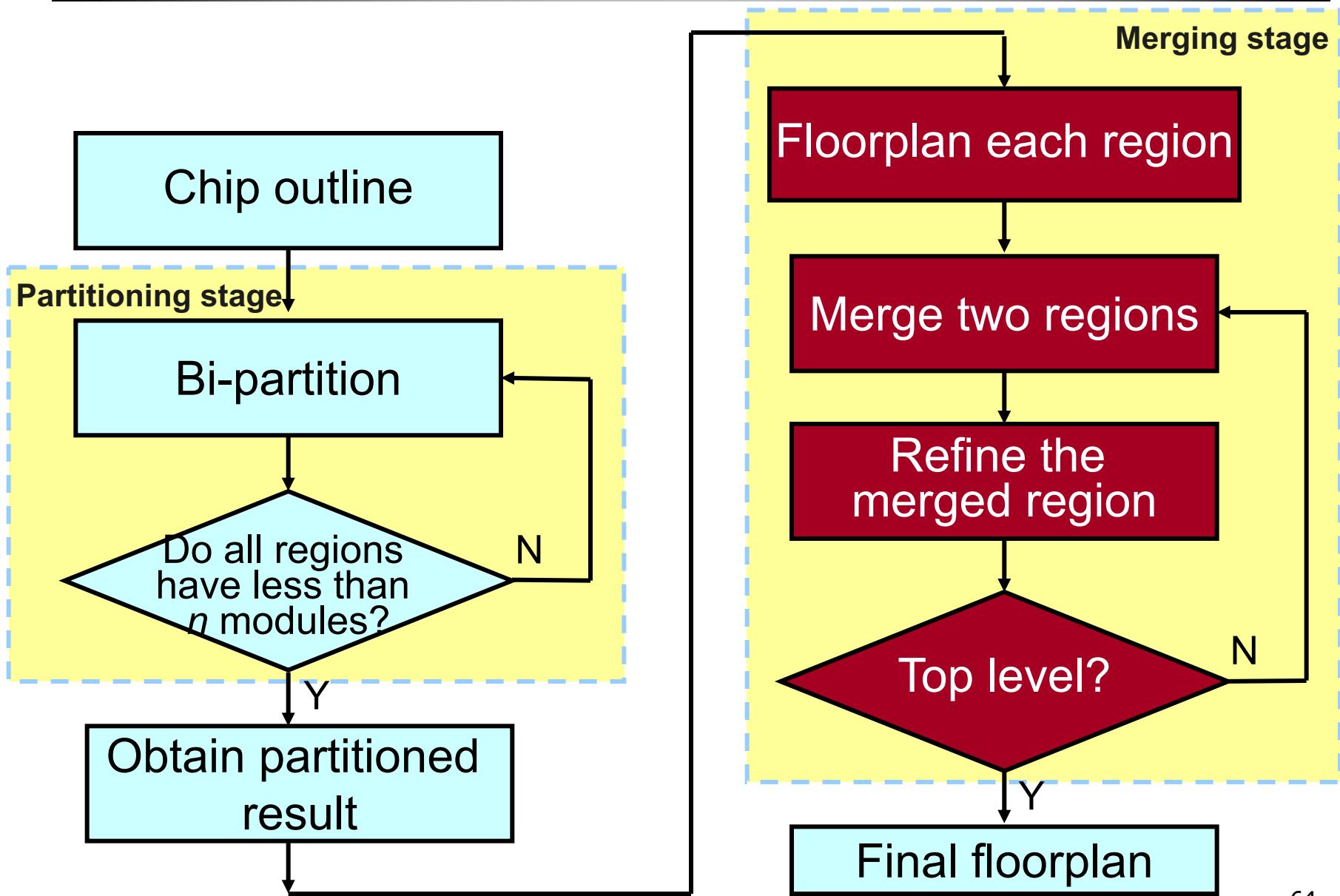
Stage 1: Partitioning Stage

- 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.

Design Flow



Stage 2: Merging Stage

- Construct a B*-tree and find the sub-floorplans for each sub-region (fixed-outline floorplanning).
- Cost function for the simulated annealing:
 - Fixed-outline floorplanning

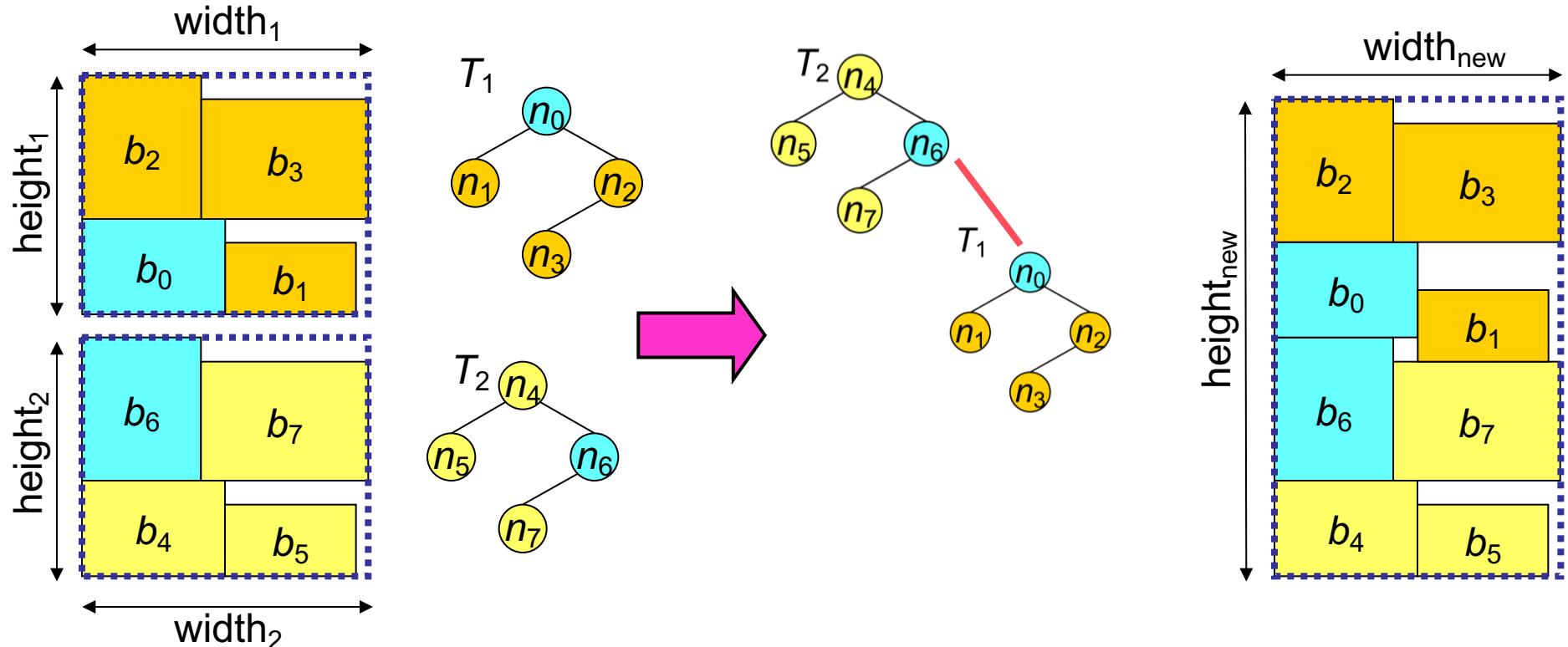
$$Cost = k_1 * area + k_2 * wirelength + k_3 * \left(\frac{W}{H} - \frac{W^*}{H^*} \right)^2$$

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

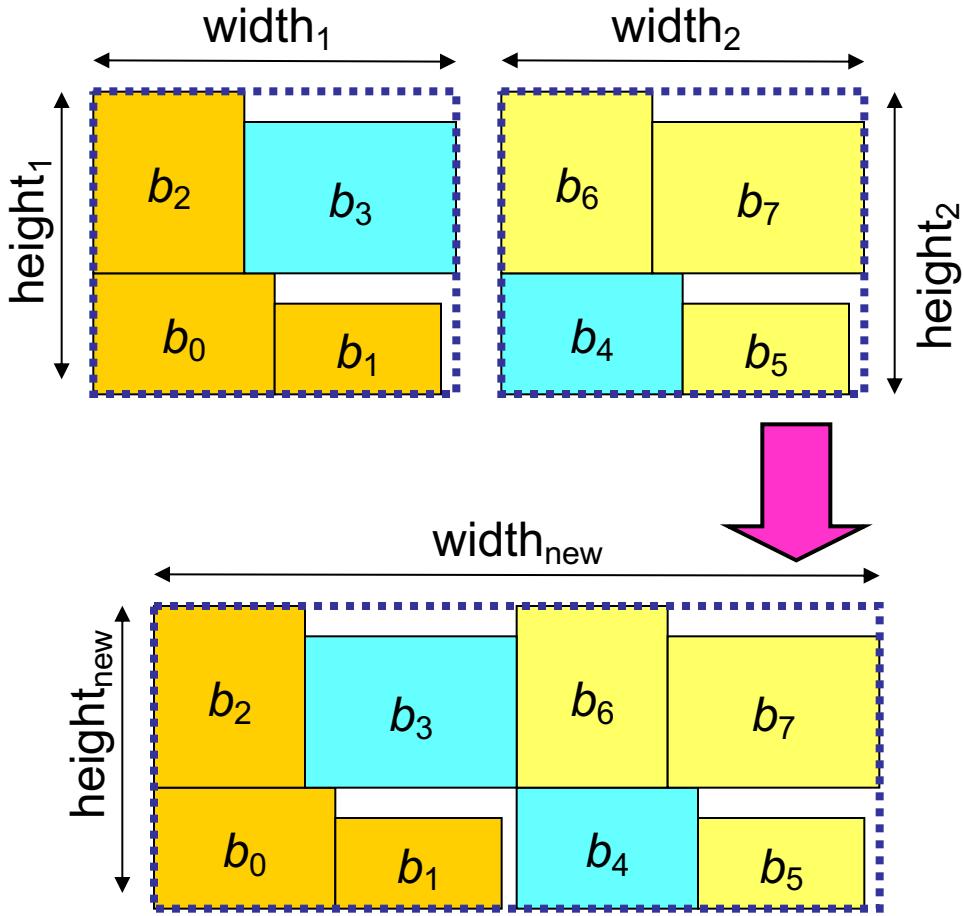
Make the root of the top B*-tree the right child of the right-most node of the bottom B*-tree.



$$height_{new} \leq height_1 + height_2$$

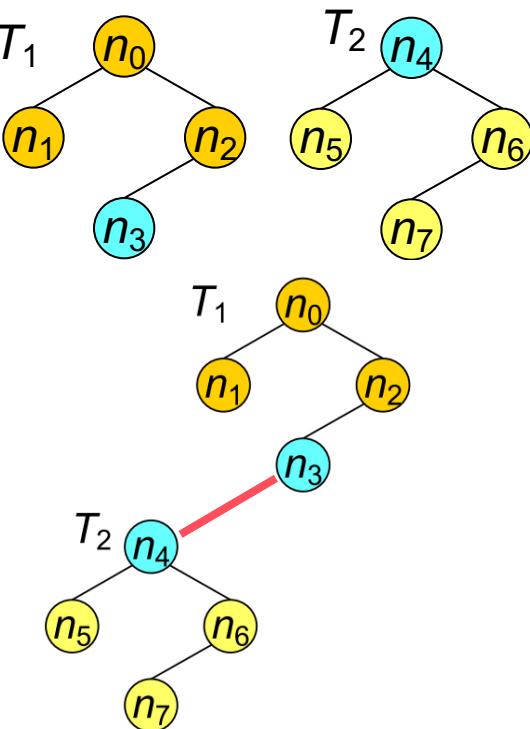
$$width_{new} = \max(width_1, width_2)$$

Horizontal Merging



$$height_{new} = \max(height_1, height_2)$$

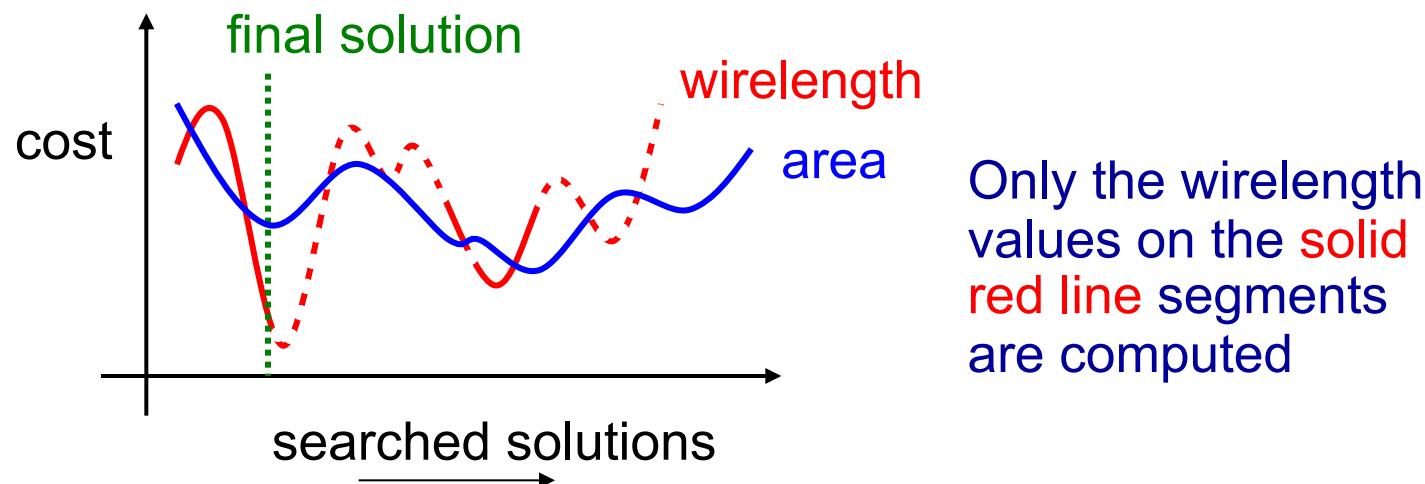
$$width_{new} = width_1 + width_2$$



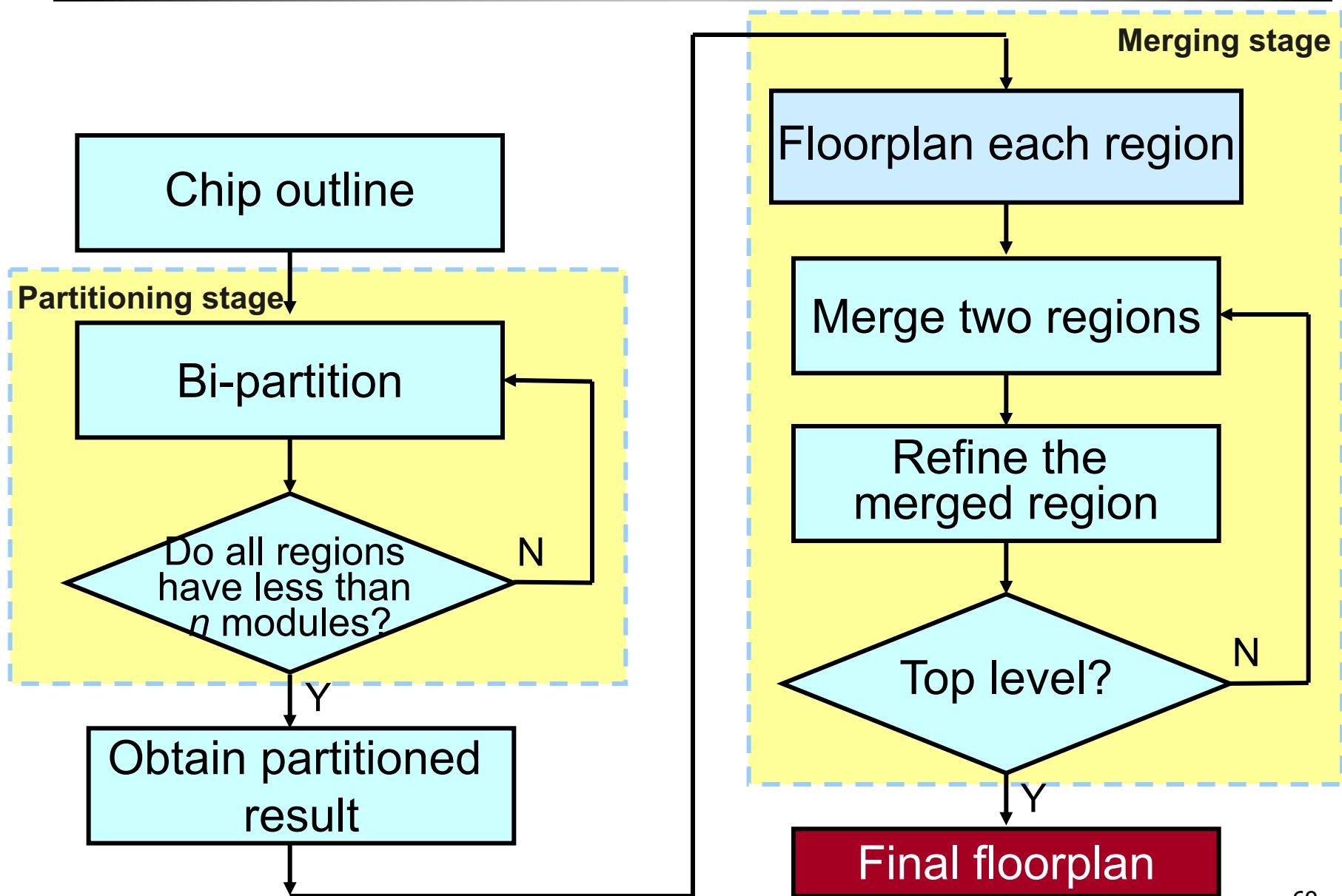
Make the root of the right B*-tree the left child of the node corresponding to the right-most module of the left B*-tree.

Accelerative Fixed-outline Floorplanning (AFF)

- . Floorplanning spends most time in computing the wirelength.
- . To speed up floorplanning, we perform area-driven fixed-outline floorplanning (set $k_2=0$).
- . Calculate the wirelength only when the floorplan has a smaller area and can fit into the bounding box.
- . Effective and efficient, especially for large-scale circuits.



Design Flow



Floorplanning Framework Comparison

Packages	Characteristics
Parquet [TVLSI-03]	<ul style="list-style-type: none">. Flat framework. Variable-die/Fixed-die
Capo [ICCAD-04]	<ul style="list-style-type: none">. Top-down hierarchical framework. Fixed-die. Partitioning and fixed-outline floorplanning
PATOMA [ASPDAC-05]	<ul style="list-style-type: none">. Top-down hierarchical framework. Fixed-die. Look-ahead floorplanning
MB*-tree [DAC-03, TCAD-07]	<ul style="list-style-type: none">. Λ-shaped multilevel framework. local (clustering) → global (declustering). Variable-die
IMF [ICCAD-05, TCAD-08]	<ul style="list-style-type: none">. V-shaped multilevel framework. global (partitioning) → local (merging). Fixed-die

Benchmarks: MCNC, GSRC

- IMF obtains 10%, 31%, 11%, 8% less HPWL than Parquet's, PATOMA's, Capo 9.0's, and Capo 9.4's, respectively
- IMF+AFF has 4X speedup on average compared with IMF.

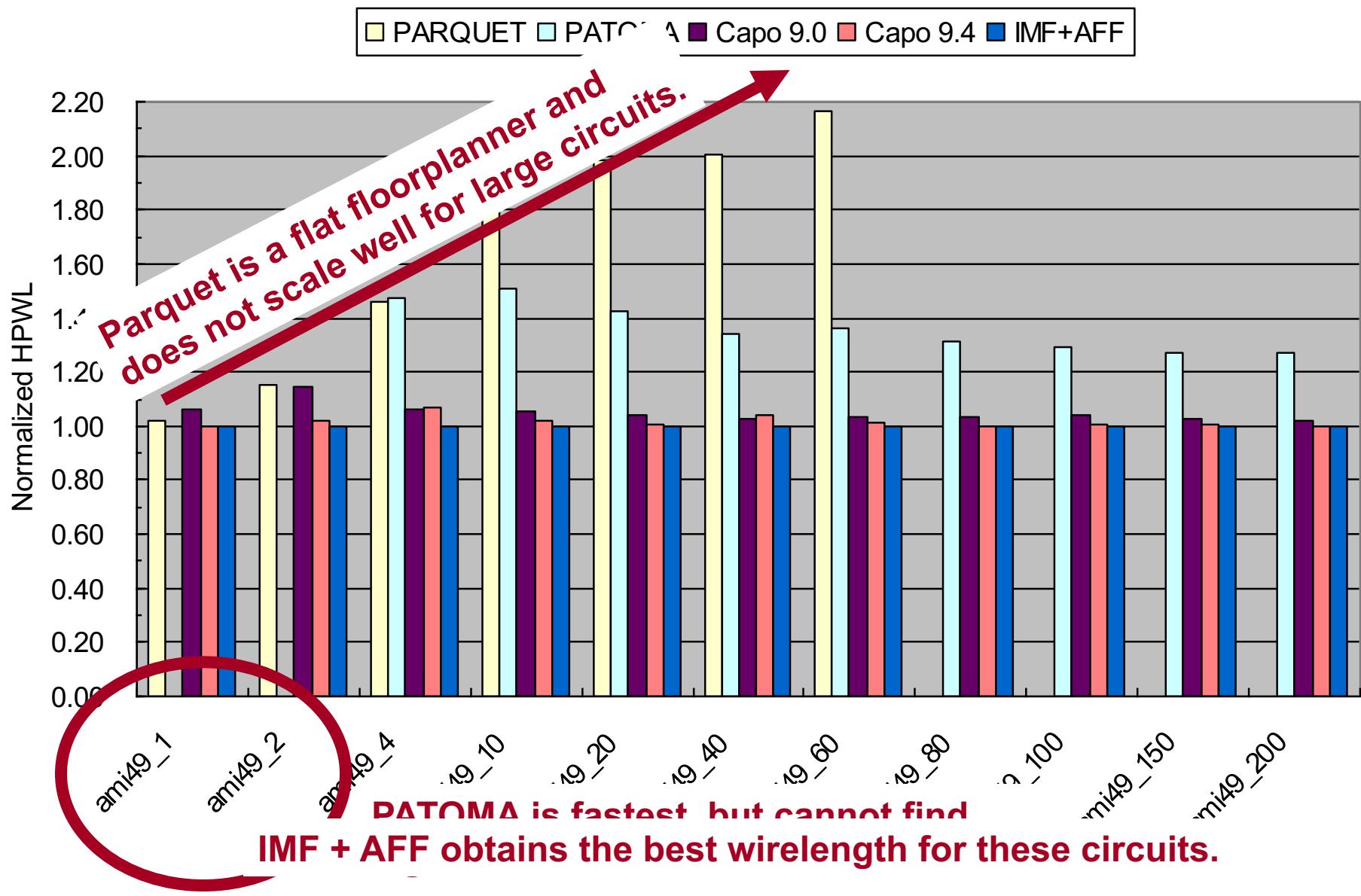
	HPWL				Time (sec)							
	Parquet	PATO-MA	Capo 9.0	Capo 9.4	IMF (Ours)	IMF+AF F (Ours)	Parquet	PATO-MA	Capo 9.0	Capo 9.4	IMF (Ours)	IMF+AF F (Ours)
apte	4.48e5	5.42e5	5.49e5	4.78e5	4.25e5	5.00e5	0.2	0.0	0.2	0.3	0.7	0.1
xerox	5.24e5	NR	5.31e5	5.21e5	5.05e5	5.56e5	0.6	NR	1.2	0.3	0.9	0.2
hp	1.30e5	NR	1.30e5	1.66e5	1.24e5	1.75e5	0.4	NR	1.2	1.1	0.7	0.1
ami33	6.80e4	NR	7.32e4	6.24e4	6.20e4	6.52e4	1.7	NR	3.4	2.7	2.0	0.8
ami49	8.83e5	NR	1.00e6	9.22e5	8.68e5	9.48e5	5.5	NR	3.9	5.1	5.4	1.2
n10	3.80e4	NR	4.19e4	4.05e4	3.61e4	3.92e4	0.3	NR	0.4	0.5	0.5	0.2
n30	1.14e5	NR	1.10e5	1.12e5	1.07e5	1.08e5	2.3	NR	0.8	1.4	1.7	0.6
n50	1.47e5	NR	1.41e5	1.40e5	1.34e5	1.36e5	4.6	NR	2.3	2.5	7.0	1.5
n100	2.42e5	NR	2.24e5	2.15e5	2.08e5	2.09e5	17.5	NR	4.6	4.8	11.5	2.3
n200	4.33e5	4.56e5	3.86e5	3.77e5	3.70e5	3.73e5	77.2	0.6	15.0	14.1	64.6	4.2
n300	6.47e5	6.94e5	5.23e5	4.99e5	4.90e5	4.94e5	166.2	0.6	13.5	14.9	91.7	5.5
Comp.	1.10	1.31	1.11	1.08	1.00	1.09	1.04	0.02	0.74	0.68	1.00	0.23

Benchmarks: ami49_x

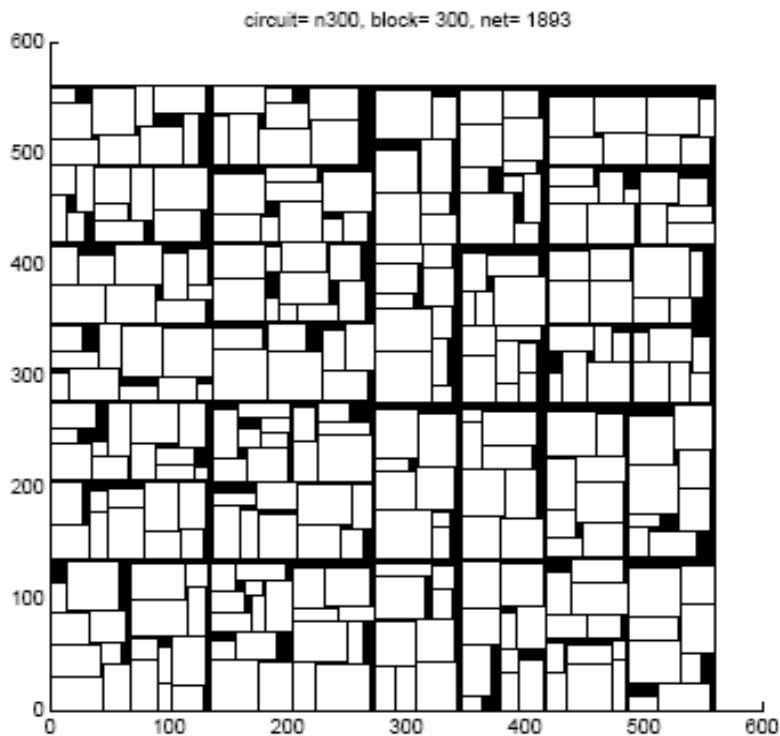
- Achieve 57%, 36%, 5%, 2% less HPWL than Parquet's, PATOMA's, Capo 9.0's, and Capo 9.4's, respectively
- Obtain 100X, 2.5X, 4X speedup compared to Parquet, Capo 9.0, and Capo 9.4, respectively

	HPWL					Time (min)				
	Parquet	PATOMA	Capo 9.0	Capo 9.4	IMF+AFF (Ours)	Parquet	PATOMA	Capo 9.0	Capo 9.4	IMF+AFF Ours
ami49_1	1.88e5	NR	1.96e5	1.81e5	1.81e5	0.1	NR	0.0	0.2	0.0
ami49_2	5.38e5	NR	5.36e5	4.78e5	4.67e5	0.3	NR	0.1	0.1	0.0
ami49_4	1.56e6	1.58e6	1.14e6	1.15e6	1.07e6	1.3	0.0	0.3	0.5	0.1
ami49_10	6.03e6	4.97e6	3.48e6	3.35e6	3.29e6	9.6	0.0	0.7	0.7	0.2
ami49_20	1.61e7	1.16e7	8.48e6	8.17e6	8.13e6	44.5	0.1	1.5	1.5	0.5
ami49_40	4.04e7	2.70e7	2.07e7	2.10e7	2.02e7	272.9	0.1	3.3	5.5	1.3
ami49_60	7.41e7	4.65e7	3.54e7	3.46e7	3.42e7	734.4	0.2	5.5	5.3	2.2
ami49_80	NR	6.65e7	5.22e7	5.06e7	5.05e7	NR	0.4	8.8	7.1	3.5
ami49_100	NR	8.85e7	7.13e7	6.90e7	6.86e7	NR	0.5	10.6	8.9	5.1
ami49_150	NR	1.53e8	1.24e8	1.22e8	1.21e8	NR	1.0	13.7	14.4	11.0
ami49_200	NR	2.28e8	1.83e8	1.80e8	1.80e8	NR	1.7	19.8	18.4	19.1
Comp.	1.57	1.36	1.05	1.02	1.00	100.6	0.10	2.53	4.06	1.00

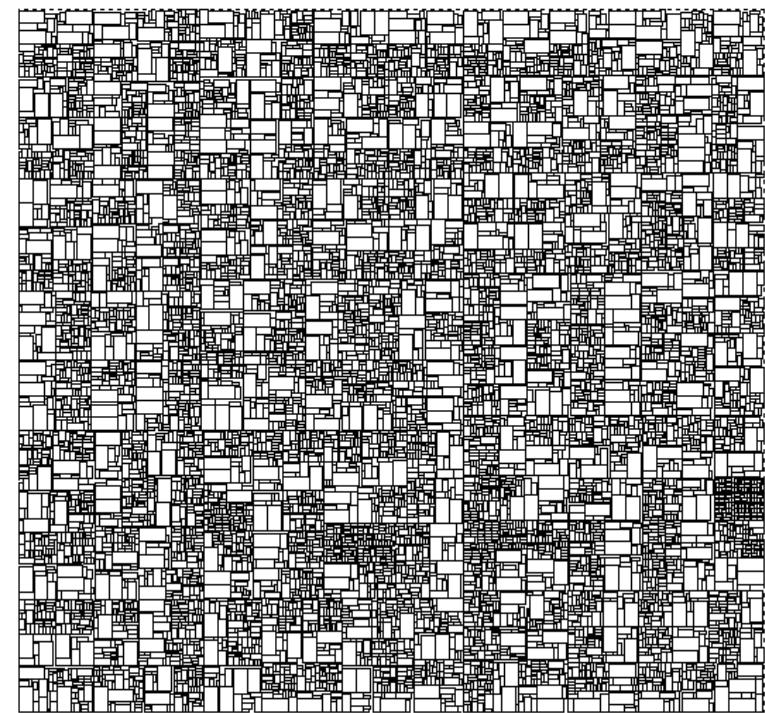
HPWL Comparisons



Resulting Floorplans



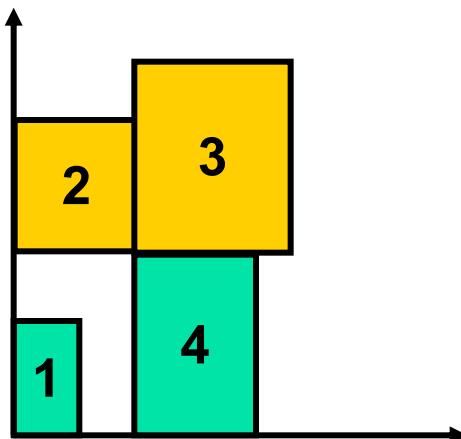
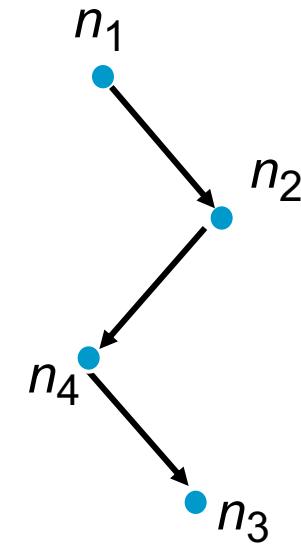
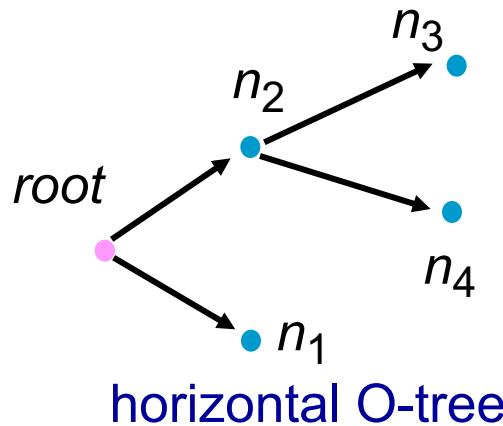
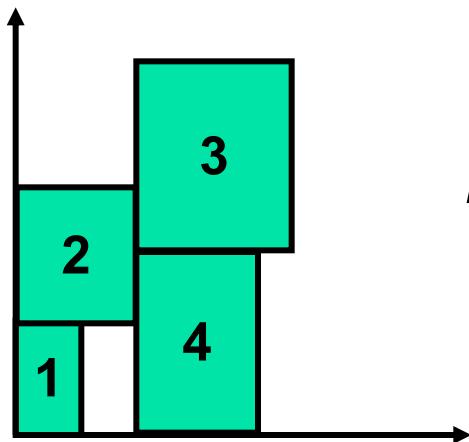
n300 (300 modules)



ami49_200 (9800 modules)

Problems with Tree-based Representations

- Can represent only compacted placements.
 - May lose an optimal solution for wirelength optimization.

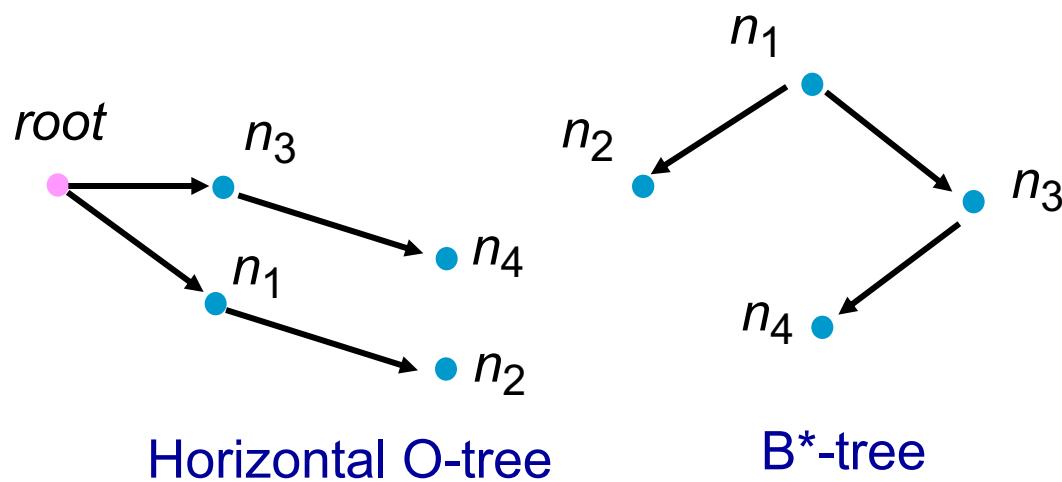
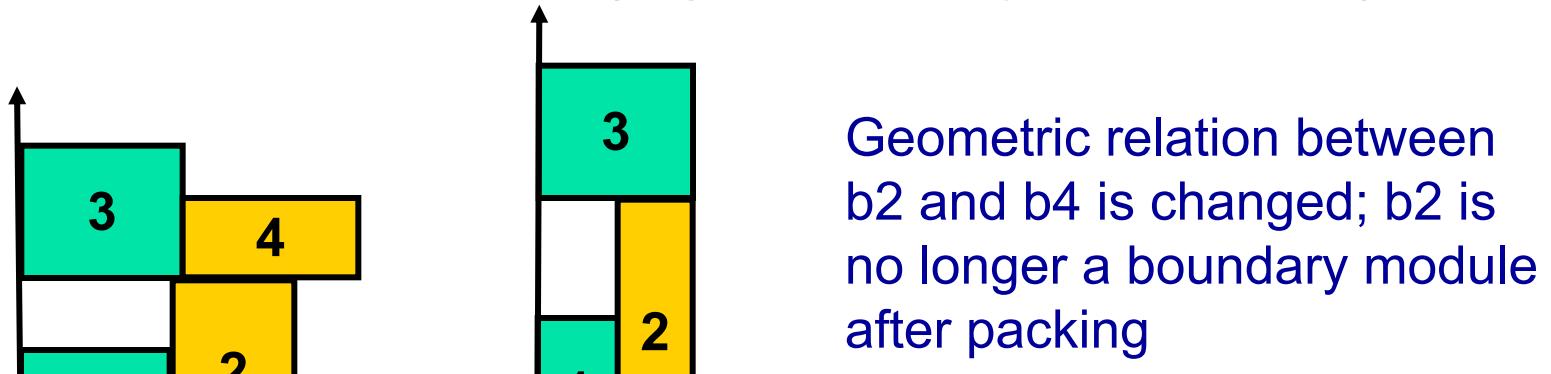


O-tree??

B*-tree??

Problems with Tree-based Representations (cont'd)

- Harder to deal with the 2-D area sizing problem or some placement constraints (e.g., boundary constraints).

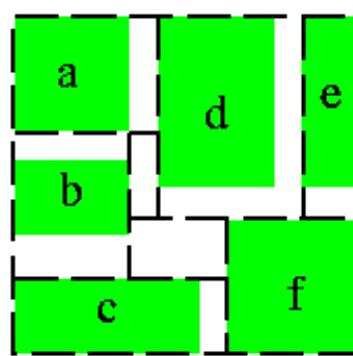
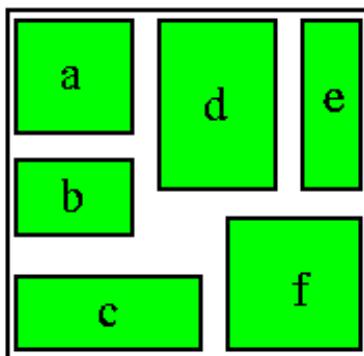


P*-admissible Solution Space

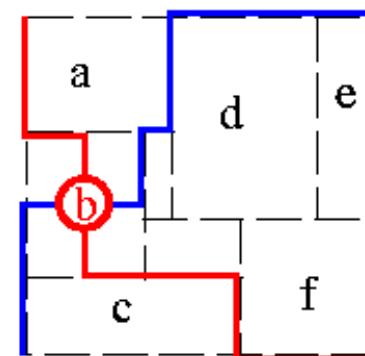
- . **P-admissible** solution space for Problem P (Murata et al., ICCAD-95)
 1. the solution space is finite,
 2. every solution is feasible,
 3. evaluation for each configuration is possible in polynomial time and so is the implementation of the corresponding configuration, **and**
 4. the configuration corresponding to the best evaluated solution in the space coincides with an optimal solution of P.
- . **P*-admissible** solution space (Lin & Chang, DAC-02, TCAD-04)
 5. The relationship between any two blocks is defined in the representation (fully topological representation).
- . Slicing floorplan is **not** P-admissible. Why?
- . B*-trees are not a P*-admissible representation.
- . A P*-admissible floorplan representation: **Sequence Pair**.

Sequence Pair (SP)

- Murata, Fujiyoshi, Nakatake, Kajitani, “Rectangle-Packing Based Module Placement,” ICCAD-95.
- Represent a packing by a pair of module-name sequences (e.g., $(abdecf, cbfade)$).
 - Size of the solution space: $(n!)^2$
- Correspond all pairs of the sequences to a P-admissible (**P***admissible) solution space.
- Search in the P-admissible (**P***-admissible) solution space (by SA).
 - Swap two nodes only in a sequence
 - Swap two nodes in both sequences



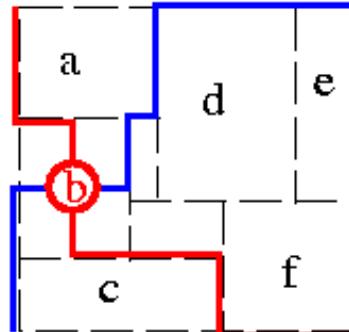
A floorplan



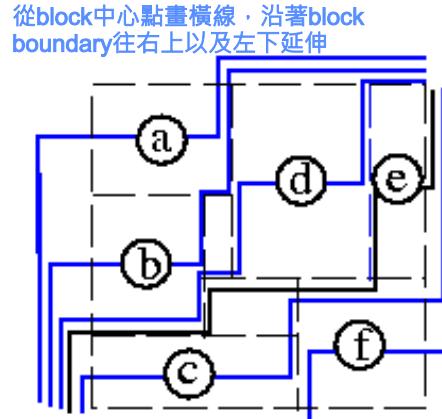
Loci of module b

Relative Module Positions

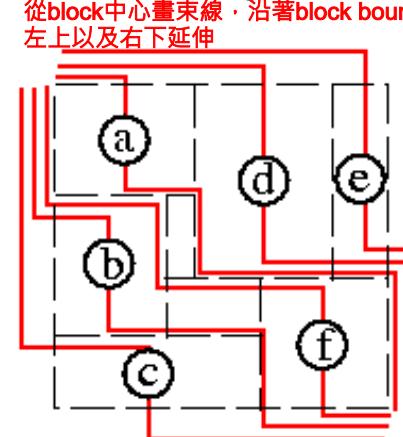
- A floorplan is a partition of a chip into **rooms**, each containing at most one block.
- Locus** (right-up, left-down, up-left, down-right)
 - Take a non-empty room.
 - Start at the center of the room, walk in two alternating directions to hit the sides of rooms **or previous loci**.
 - Continue until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



Loci of module b



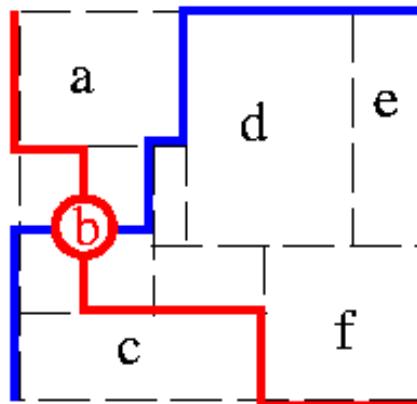
Positive loci: abdecf
從左上看到右下



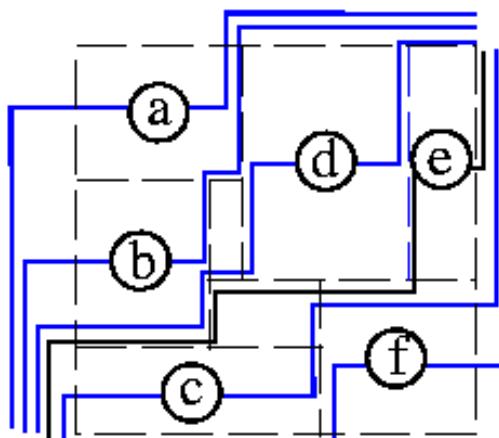
Negative loci: cbfade
從左下看到右上

Geometrical Information

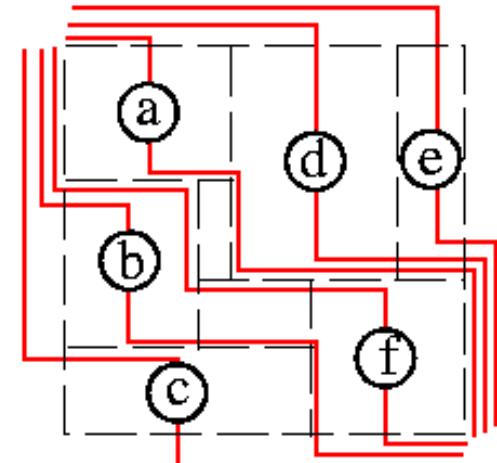
- No pair of positive (negative) loci cross each other, i.e., **loci are linearly ordered**.
- SP uses two sequences (Γ_+ , Γ_-) to represent a floorplan.
 - H-constraint:** $(\dots a \dots b \dots, \dots a \dots b \dots)$ iff a is on the left of b
 - V-constraint:** $(\dots a \dots b \dots, \dots b \dots a \dots)$ iff b is below a



Loci of module b



Positive loci: abdecdf

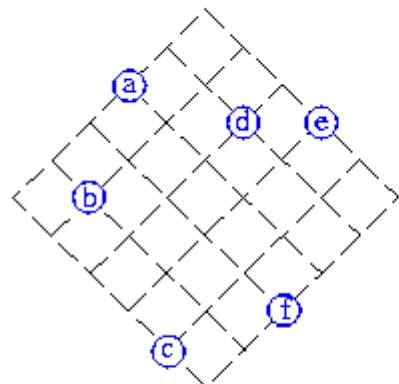


Negative loci: cbfade

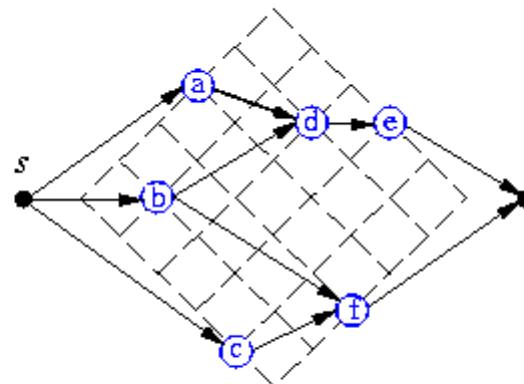
$$(\Gamma_+, \Gamma_-) = (\textcolor{red}{abdecdf}, \textcolor{red}{cbfade})$$

(Γ_+, Γ_-) -Packing

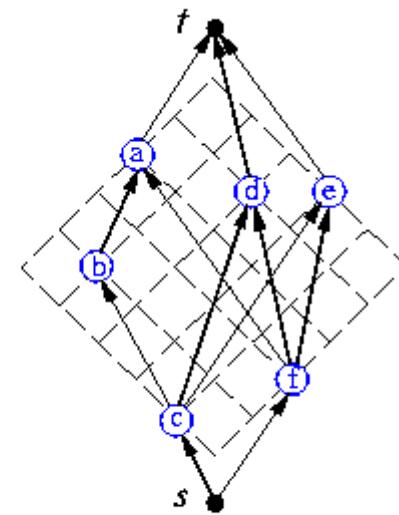
- For every SP (Γ_+, Γ_-) , there is a (Γ_+, Γ_-) packing.
- Horizontal constraint graph** $G_H(V, E)$ (similarly for $G_V(V, E)$):
 - V : source s , sink t , n vertices for modules.
 - E : (s, x) and (x, t) for each module x , and (x, y) iff x must be left to y .
 - Vertex weight:** 0 for s and t , width of module x for the other vertices.



Packing for sequence pair:
 $(ubdecf, cbfude)$



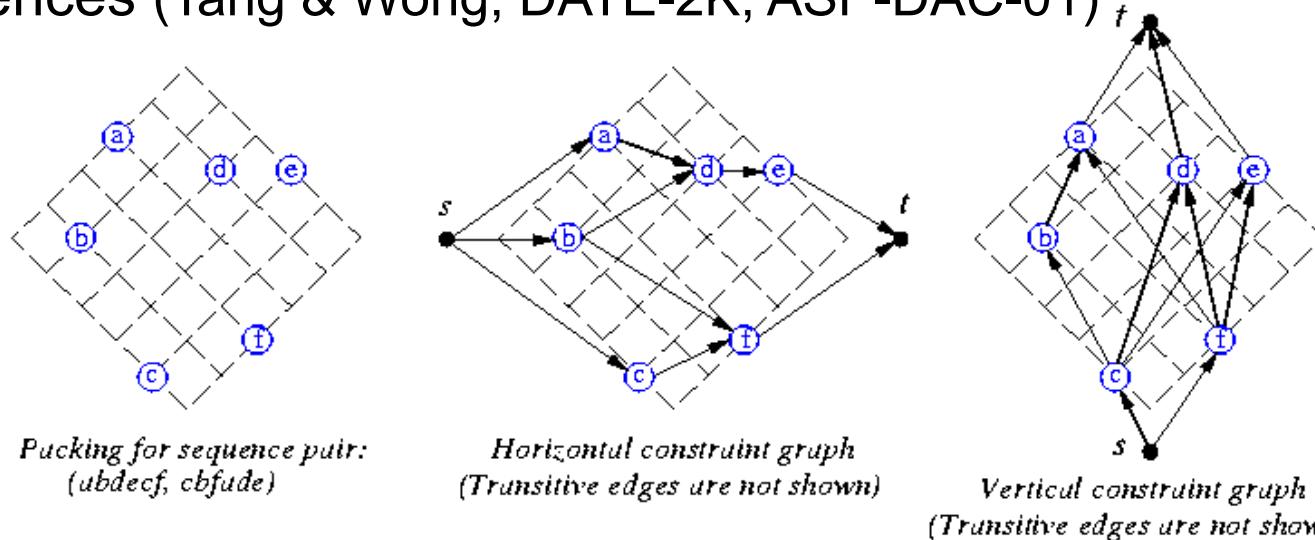
Horizontal constraint graph
(Transitive edges are not shown)



Vertical constraint graph
(Transitive edges are not shown)

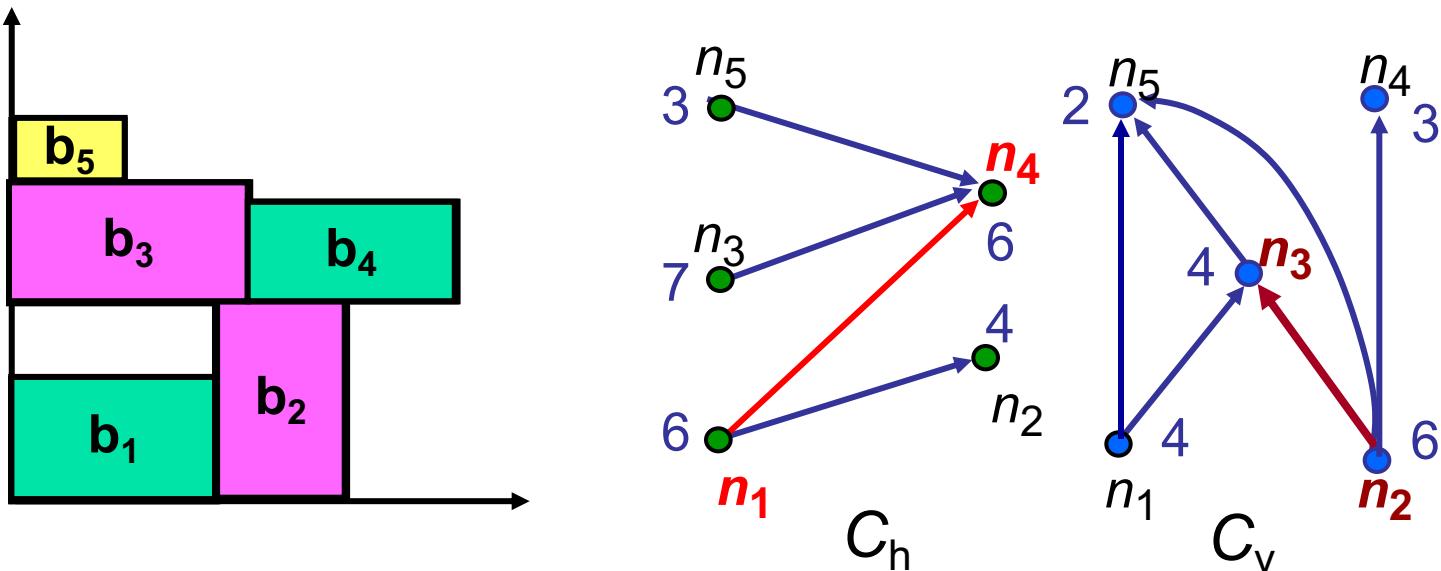
Cost Evaluation

- **Optimal (Γ_+, Γ_-) -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 the values of the longest path lengths between s and the corresponding vertex in G_H and G_V , respectively.
- Cost evaluation can be done in $O(n \lg \lg n)$ time by computing the longest common subsequence of the two sequences (Tang & Wong, DATE-2K, ASP-DAC-01)



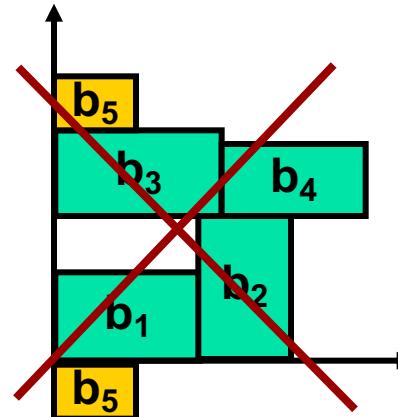
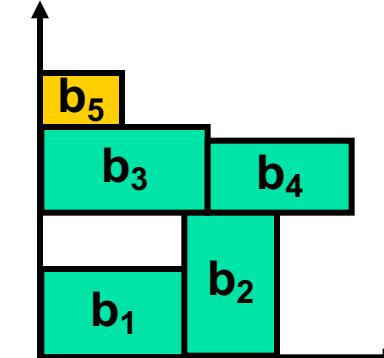
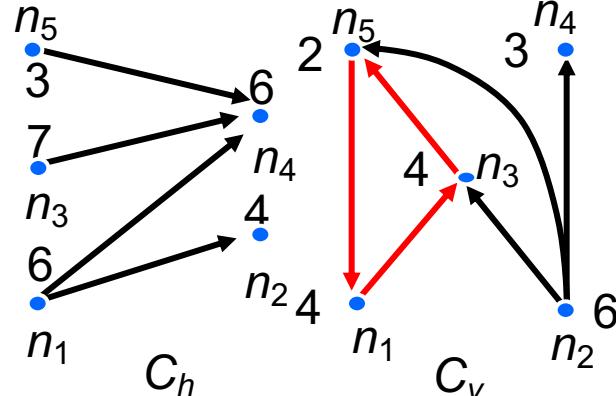
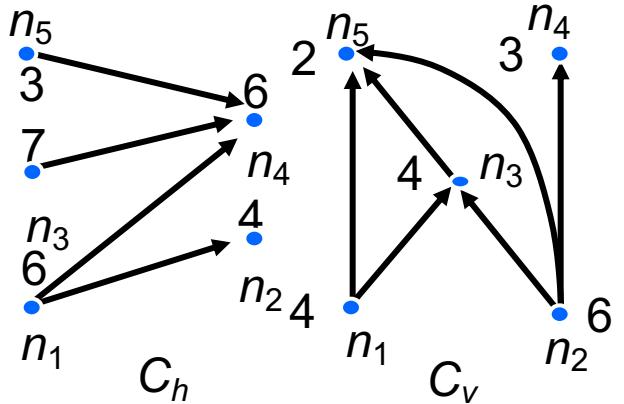
P^* -admissible Transitive Closure Graph (TCG)

- Lin & Chang, “TCG: A transitive closure graph representation for non-slicing floorplans,” DAC-01 (TVLSI-04)
- TCG = (C_h, C_v) : pair of vertical and horizontal constraint graphs.
TCG與sequence pair有1-to-1的關係
 - C_h (C_v) represents the horizontal (vertical) geometric relations between modules.
 - Transforms diagonal relations into horizontal relations if no vertical constraints among modules.
- Vertex weights denote module widths (heights).



Feasibility of TCG

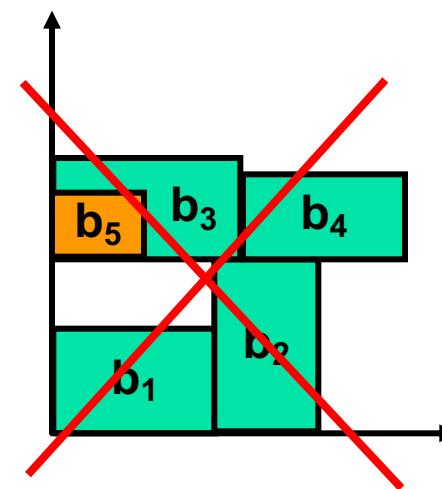
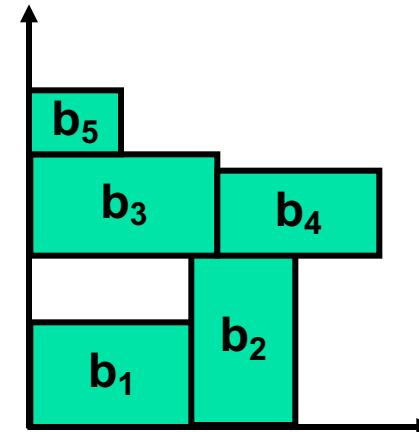
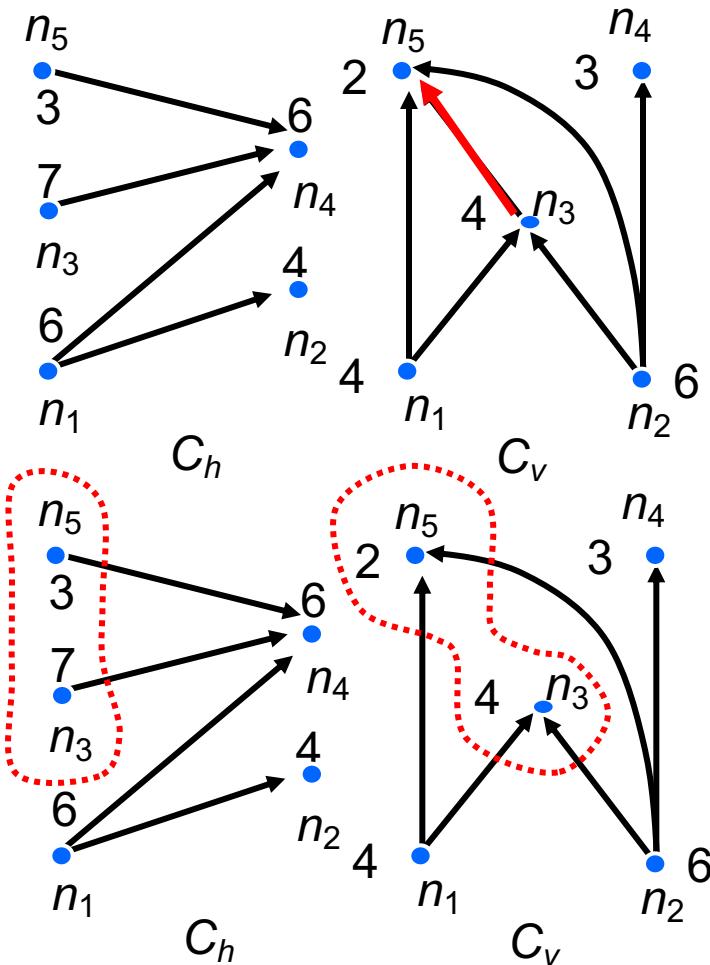
1. C_h and C_v are acyclic.
2. Each pair of nodes must be connected by exactly one edge either in C_h or in C_v .
3. The transitive closure of C_h (C_v) is equal to C_h (C_v) itself.



Infeasible
placement

Feasibility of TCG

2. Each pair of nodes must be connected by exactly one edge either in C_h or in C_v to prevent any overlap among modules.

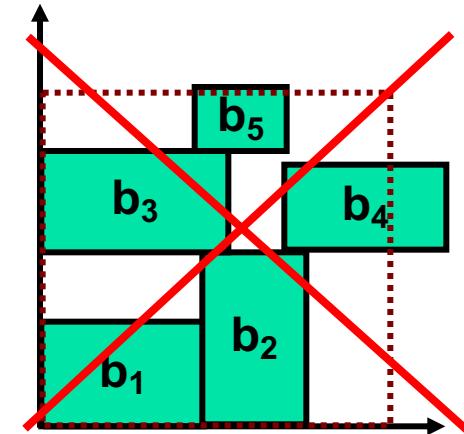
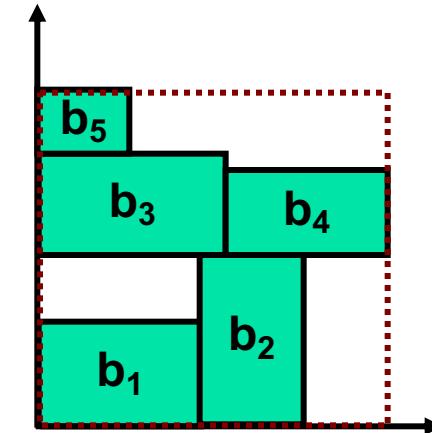
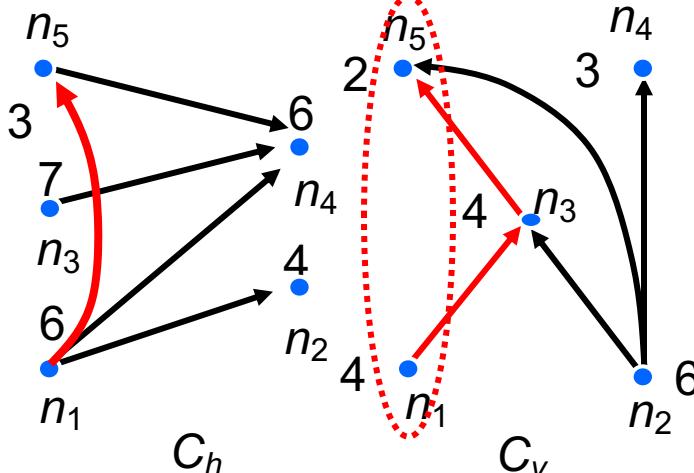
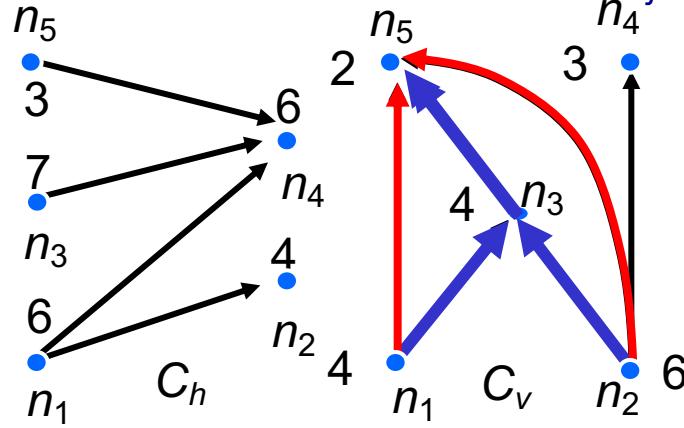


b_3 and b_5
overlap

Feasibility of TCG

3. The transitive closure of C_h (C_v) is equal to C_h (C_v) itself to reduce the solution space.

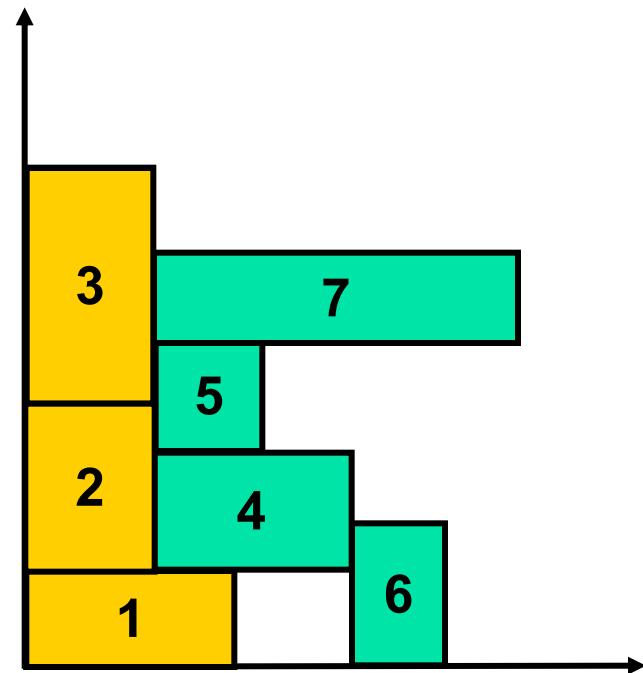
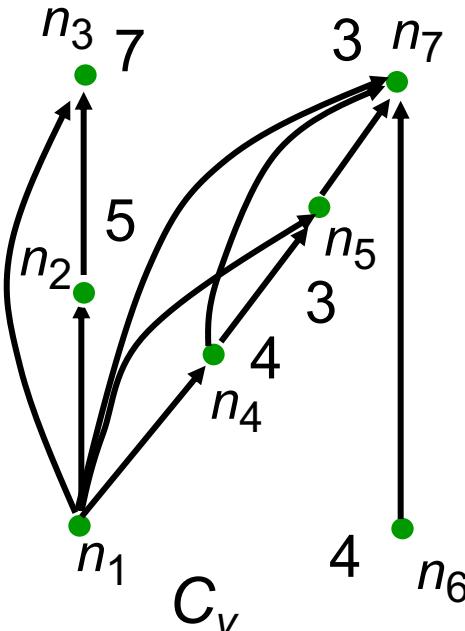
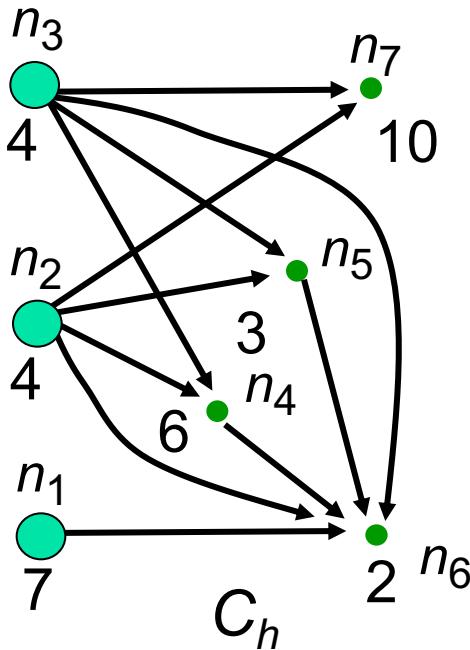
- The transitive closure of a directed graph $G=(V, E)$ is the graph $G'=(V, E')$, where $E'=\{(n_i, n_j): \text{there is a path from } n_i \text{ to } n_j \text{ in } G\}$.



Boundary Modules in TCG

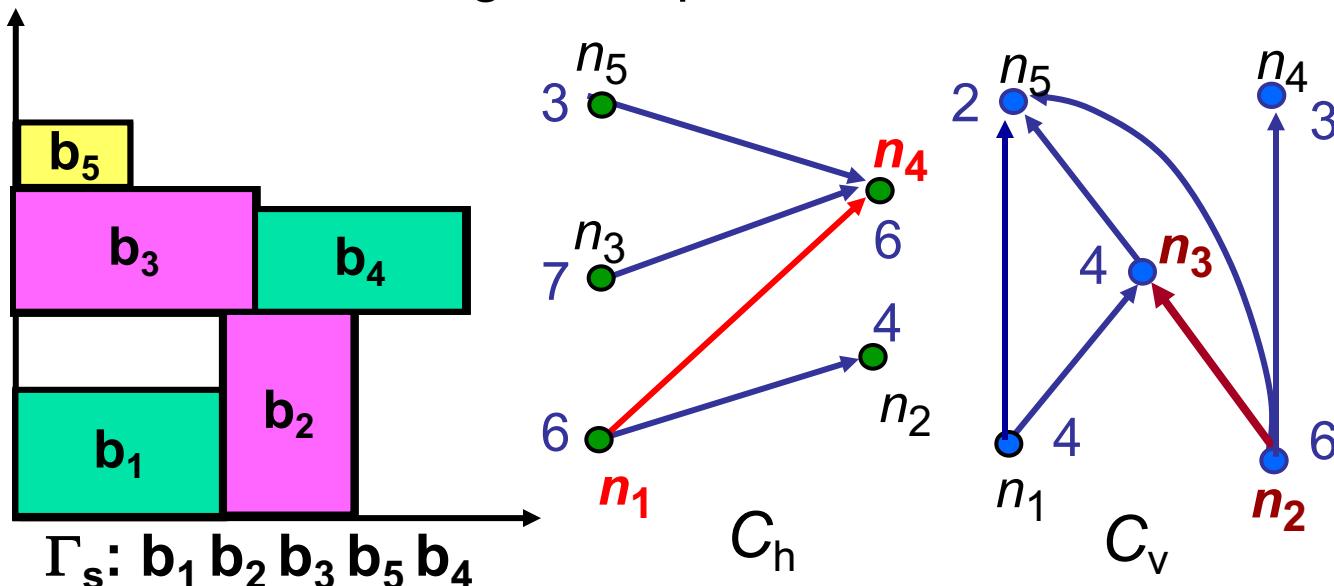
- b_i is a **left** (**right**) boundary module if n_i 's **in-degree** (**out-degree**) in C_h is zero.
- b_i is a **bottom** (**top**) boundary module if n_i 's **in-degree** (**out-degree**) in C_v is zero.
若有一個node:
1. 沒有箭頭指向他-->該node在最左邊
2. 沒有指向他人-->該node在最右邊

The in-degrees of n_1 , n_2 , and n_3 are zero



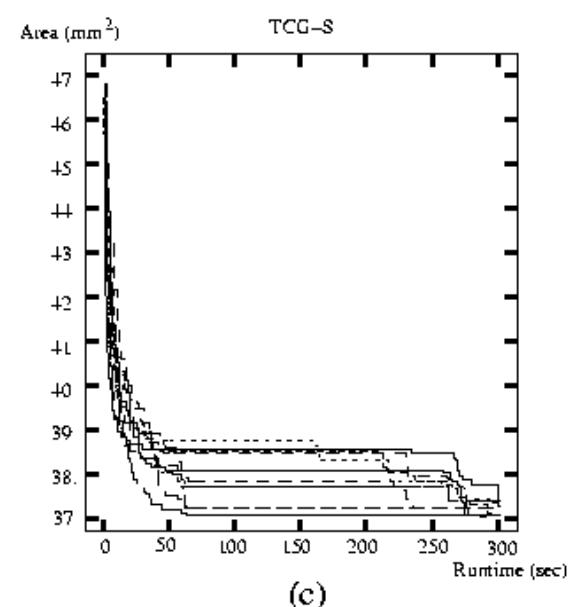
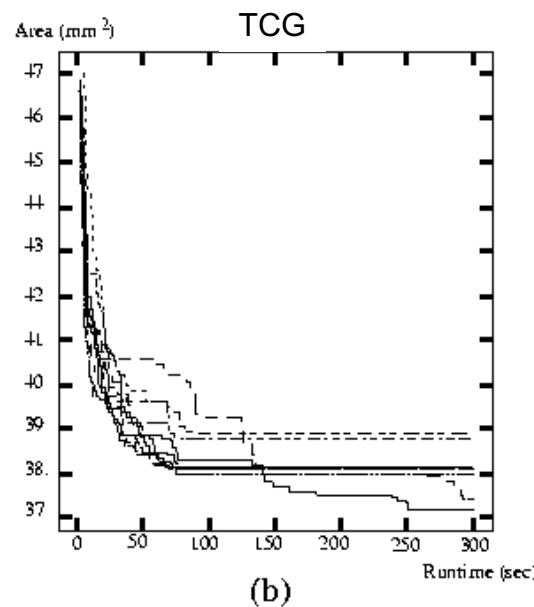
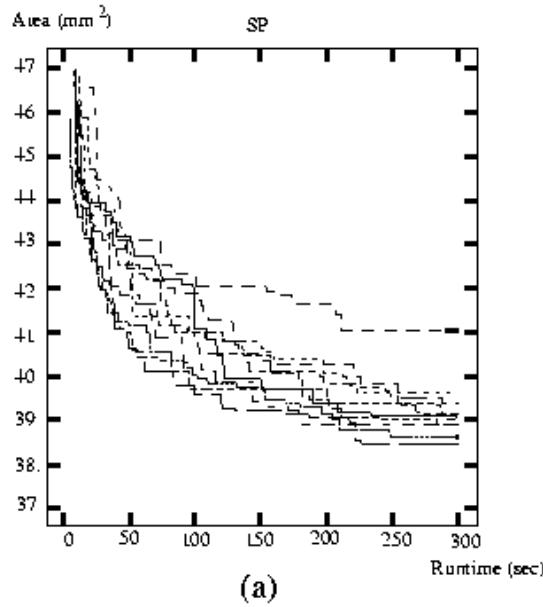
P*-admissible TCG-S

- Lin & Chang, “TCG-S: An orthogonal coupling of P*-admissible representations for general floorplans,” DAC-2002 (TCAD-2004)
- TCG-S = (C_h, C_v, Γ_s) : TCG + a module sequence.
 - Γ_s represents the packing sequence of modules
 - Iteratively traverse the module in the leftmost with all modules below it having been traversed.
 - Is used for speeding up the packing scheme ($O(n \lg n)$ time).
- Leads to faster convergence speed and more stable results.



Convergence Speed & Stability

- **Convergence speed** and **stability** are two important criteria to evaluate the quality of a representation.
- TCG-S converges very fast and is very stable.
- Convergence speed and stability: TCG-S > TCG > SP.



Comparisons

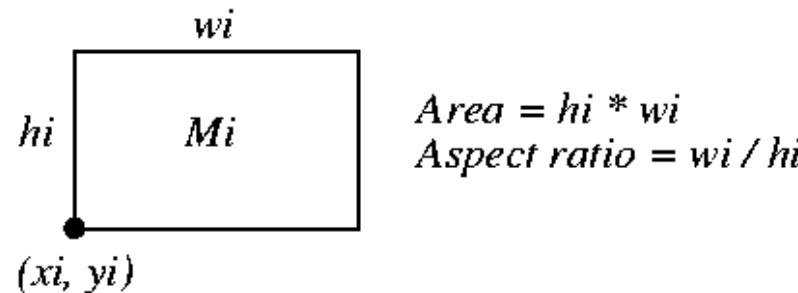
Represent.	Solution Space	Packing Time	Guarantee Feasible Perturbations?	Flexibility
SP	$(n!)^2$	$O(n^2)$	O	4
BSG	$n! C(n^2, n)$	$O(n^2)$	O	4
TCG	$(n!)^2$	$O(n^2)$	O	4
TCG-S	$(n!)^2$	$O(n \lg n)$	O	4
O-tree	$O(n!2^{2n}/n^{1.5})$	$O(n)$	\otimes	3
B*-tree	$O(n!2^{2n}/n^{1.5})$	$O(n)$	\otimes	3
CS	$O((n!)^2)$	$O(n)$	\otimes	3
CBL	$O(n!2^{3n})$	$O(n)??$	X	2
Normalized Polish Exp.	$O(n!2^{2.6n}/n^{1.5})$	$O(n)$	O	1

Flexibility: Can represent 4 (general; P*-admissible);

3 (compacted; P-admissible); 2 (mosaic); 1 (slicing)

Floorplanning 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 (ILP)** formulation for the floorplan design.
 - **Linear** constraints? Objective function?



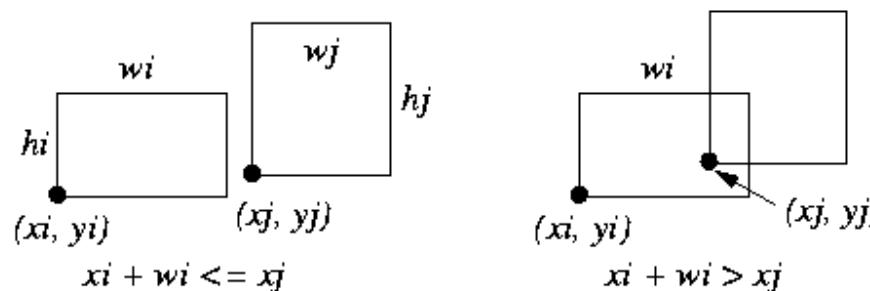
Nonoverlap Constraints

- Two modules M_i and M_j are non-overlapping, if at least one of the following linear constraints is satisfied (cases encoded by p_{ij} and q_{ij}):

M_i to the left of M_j :	$x_i + w_i \leq x_j$	p_{ij}	q_{ij}
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.
- 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 $\text{Area} = xy$, **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.

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

- . Size of the mixed ILP: for n modules,
 - # continuous variables: $O(n)$; # integer variables: $O(n^2)$; # linear constraints: $O(n^2)$.
 - Unacceptably huge program for a large n ! (How to cope with it?)
 - . Popular LP software: LINDO, Ip_solve, CPLEX, etc.

Mixed ILP for Floorplanning (cont'd)

Mixed ILP for the floorplanning problem: rigid, freely oriented modules.

$$\min \quad y$$

subject to

$$x_i + r_i h_i + (1 - r_i) w_i \leq W, \quad 1 \leq i \leq n \quad (9)$$

$$y_i + r_i w_i + (1 - r_i) h_i \leq y, \quad 1 \leq i \leq n \quad (10)$$

$$x_i + r_i h_i + (1 - r_i) w_i \leq x_j + M(p_{ij} + q_{ij}), \quad 1 \leq i < j \leq n \quad (11)$$

$$y_i + r_i w_i + (1 - r_i) h_i \leq y_j + M(1 + p_{ij} - q_{ij}), \quad 1 \leq i < j \leq n \quad (12)$$

$$x_i - r_j h_j - (1 - r_j) w_j \geq x_j - M(1 - p_{ij} + q_{ij}), \quad 1 \leq i < j \leq n \quad (13)$$

$$y_i - r_j w_j - (1 - r_j) h_j \geq y_j - M(2 - p_{ij} - q_{ij}), \quad 1 \leq i < j \leq n \quad (14)$$

$$x_i, y_i \geq 0, \quad 1 \leq i \leq n \quad (15)$$

$$p_{ij}, q_{ij} \in \{0, 1\}, \quad 1 \leq i < j \leq n \quad (16)$$

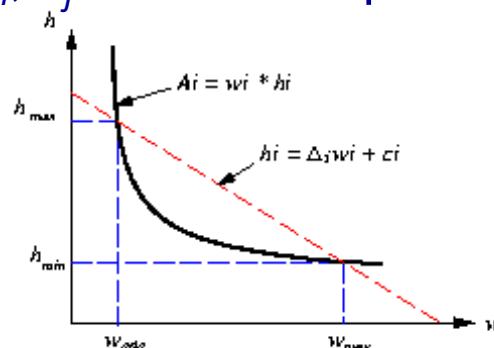
- . For each module i with free orientation, associate a 0-1 variable r_i :
 - $r_i = 0$: 0° rotation for module i .
 - $r_i = 1$: 90° rotation for module i .
- . $M = \max\{W, H\}$.

Flexible/Soft Modules

- Assumptions: w_i, h_i are unknown; area lower bound: A_i .
- Module size constraints: $w_i, h_i \geq A_i; a_i \leq w_i / h_i \leq b_i$.
- 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?
 - Can apply a first-order approximation of the equation: a line passing through (w_{min}, h_{max}) and (w_{max}, h_{min}) .

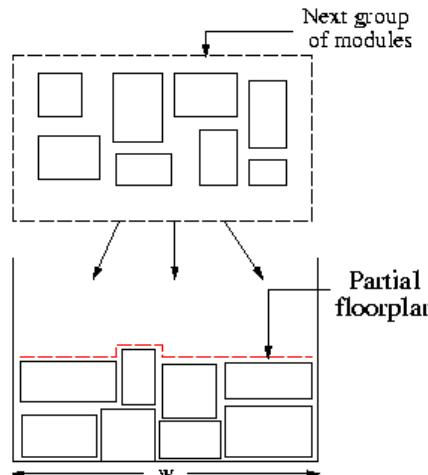
$$\begin{aligned} h_i &= \Delta_i w_i + c_i && /* y = mx + c */ \\ \Delta_i &= \frac{h_{max} - h_{min}}{w_{min} - w_{max}} && /* slope */ \\ c_i &= h_{max} - \Delta_i w_{min} && /* c = y_0 - mx_0 */ \end{aligned}$$

- Substitute $\Delta_i w_i + c_i$ for h_i to form linear constraints (x_i, y_i, w_i are unknown; $\Delta_i, \Delta_j, c_i, c_j$ can be computed as above).



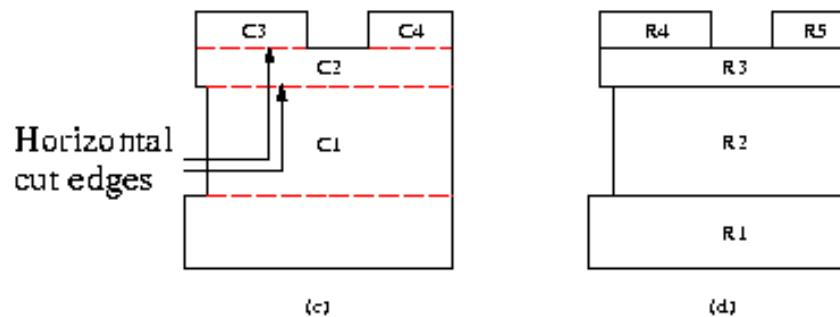
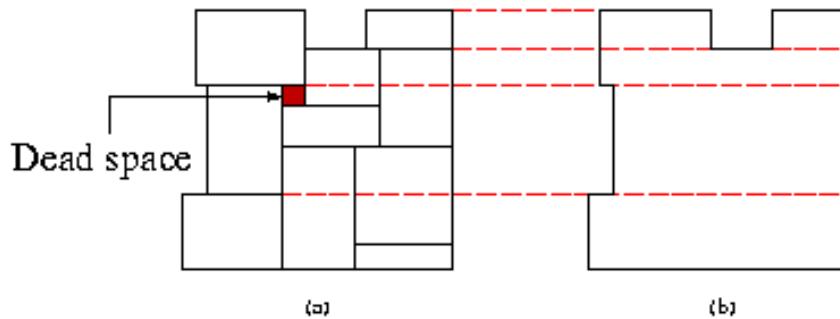
Reducing the Size of the Mixed ILP

- . 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? ↗ linear ordering based on connectivity.
 - How to minimize the # of required variables?



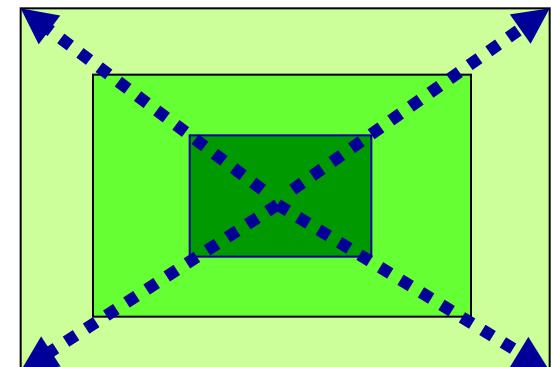
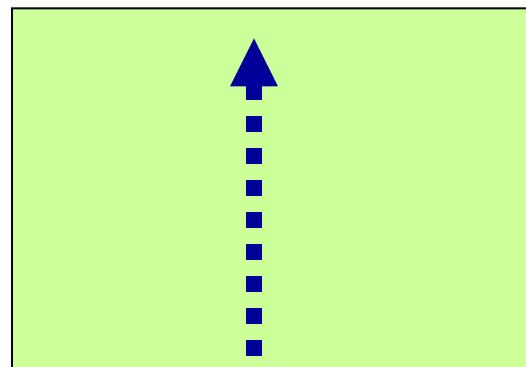
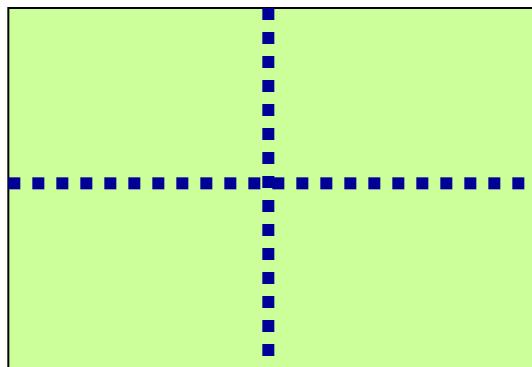
Reducing the Size of the Mixed ILP (cont'd)

- . 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.



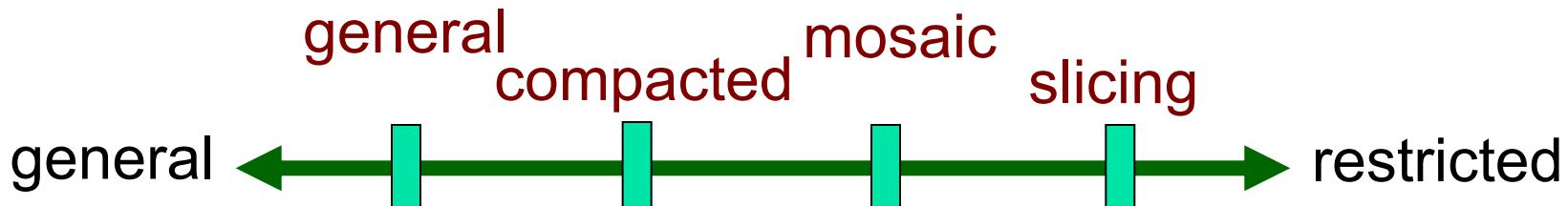
Notes on ILP Based Approaches

- . Always analyze the complexity
 - # of variables
 - # of constraints
- . Always try to reduce the problem size
 - Divide-and-conquer
 - Successive/progressive: 1D or 2D
 - Other reduction method?



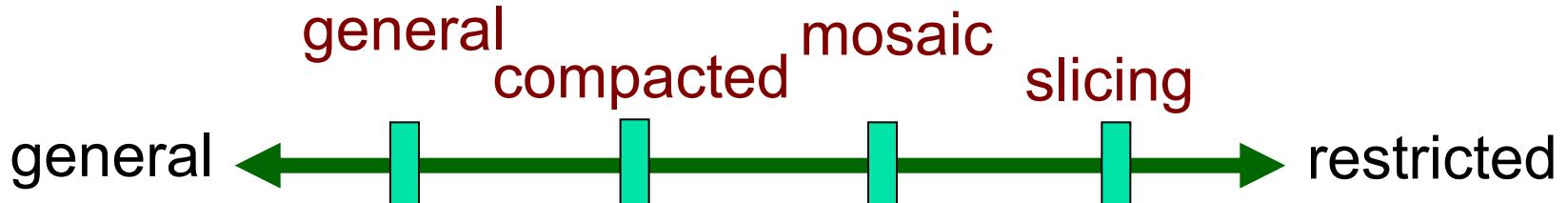
Summary on Floorplan Representations

- . Generic floorplanning objectives: (1) minimize area, (2) meet timing constraints, (3) maximize routability (minimize congestion), ((4) determine shapes of soft modules)
- . Existing representations
 - **Slicing:** slicing tree (DAC-82), normalized Polish expression (DAC-86, ASPDAC-05), generalized Polish expression (ASPDAC-04)
 - **Mosaic:** CBL (ICCAD-2k), Q-Sequence (AP-CAS-2k, DATE-02), Twin binary tree (ISPD-01)
 - **Compacted:** O-tree (DAC-99), B*-tree (DAC-2k), CS (TVLSI, 2003), T-tree (3D; ICCAD-04)
 - **General:** SP (ICCAD-95), BSG (ICCAD-96), TCG (DAC-01), TCG-S (DAC-02), ACG (ICCD-04), 3D-subTCG (ASPDAC-04), Sequence triplet (3D: IEICE)



More on Floorplan Representations

- P*-admissible representations: for general floorplans.
- P-admissible, non-P*-admissible representations (for area): all for compacted floorplans.
- What makes a good representation?
 - Easy, effective, efficient, flexible, stable
 - Must simultaneously consider solution space, packing time, flexibility, neighborhood structure for evaluation (cf. CISC vs. RISC instruction sets?)
- Since each representation has its pros and cons, can we
 - Integrate two or more representations to get a better one (e.g., TCG-S, DAC-02)
 - Apply different representations at different stages?



Other Floorplanning Problems

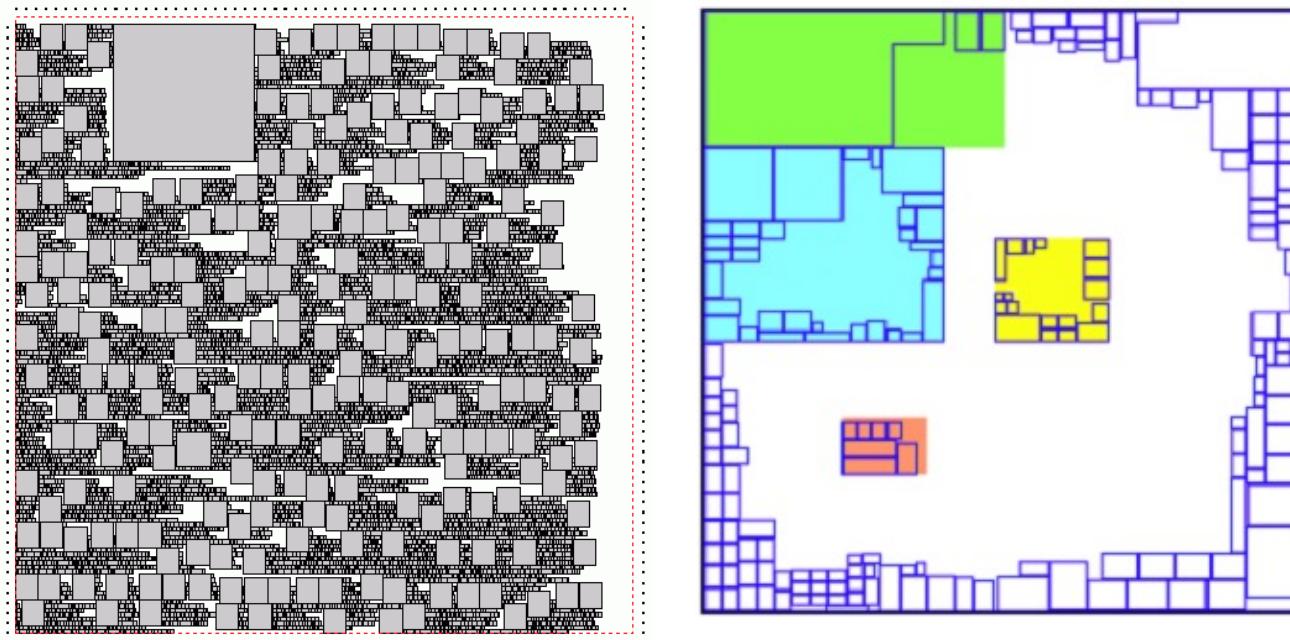
- **Soft module:** shape curve (NPE, DAC-86), analytical techniques (DAC-90, DAC-2k, ASPDAC-06), stretching range (B^* -tree, DAC-2k), Lagrangian relaxation (SP, ISPD-2k), ICCAD-08 (SP)
- **Preplaced module:** ASPDAC-98 (BSG), ASPDAC-01 (SP), DAC-2K (B^* -tree), ISCAS-01 (B^* -tree), DAC-02 (TCG-S)
- **Symmetry module:** DAC-99 (SP), ICCAD-02 (B^* -tree), ASPDAC-05 (B^* -tree), ICCAD-06 (SP), DAC-07 (B^* -tree), DAC-08 (B^* -tree)
- **Rectilinear module:** TCAD-2K (SP), ICCAD-98 (SP), ISPD-98 (SP), ISPD-01 (SP), ISPD-01 (O-tree), DATE-02 (TCG), TVLSI-02 (TCG), ICCD-2K (B^* -tree), ACM TODAES-03 (B^* -tree).
- **Fixed-outline constraint:** ICCD-01 (SP), ASPDAC-04 (NPE), ISPD-05 (B^* -tree), ISPD-07 (SP), DAC-08 (NPE)
- **Abutment constraint:** (slicing), (SP), ICCD-04 (B^* -tree)
- **Bus-driven constraint:** ICCAD-2003 (SP), ISPD-05 (B^* -tree)
- **Range constraint:** ISPD-99 (NPE), ASPDAC-01 (SP), DAC-02 (TCG-S), ICCD-04 (B^* -tree)
- **Boundary constraint:** ASPDAC-01 (SP), DAC-02 (TCG-S), IEE Proc.-02 (B^* -tree)

Other Floorplanning Problems (cont'd)

- Large-scale module floorplanning/placement (MB*-tree, DAC-03; IMF: ICCAD-05; NPE: DAC-08)
- Mixed sized cell/block floorplanning/placement (ISPD-02, ASPDAC-03, ICCAD-03, ICCAD-04, ISPD-05, DAC-07, ICCAD-08)
- Co-synthesis with floorplanning
 - Buffer planning (ICCAD-99, ISPD-2K, DAC-01, ASPDAC-03)
 - Wire planning (ICCAD-99)
 - Noise-aware floorplanning (ASPDAC-03, ICCAD-04)
 - Power supply planning (ASPDAC-01, DAC-04, ISPD-06)
 - SoC test scheduling (ICCAD-03, ASPDAC-05)
 - Architecture-driven floorplanning (DAC-04)
- B*-tree has minimized the gap between the representations for **slicing** and **non-slicing** floorplans.
- B*-tree-v1.0, TCG, and TCG-S packages are available at <http://eda.ee.ntu.edu.tw/research.htm>.

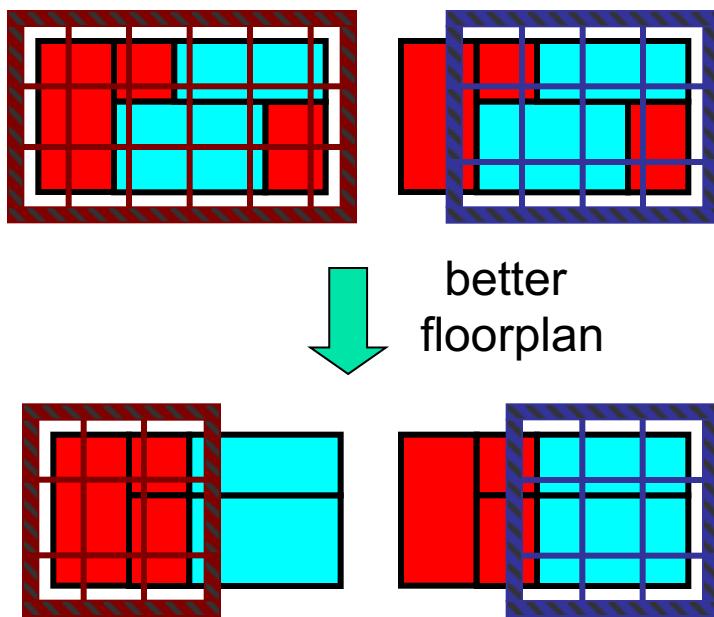
Future Work on Modern Floorplanning (1/4)

- Mixed sized cell/block floorplanning/placement (DAC-07, DAC-08, ICCAD-08)
 - Apply floorplanning techniques for macros to address various design constraints, e.g., range constraints, block rotation, block sizing

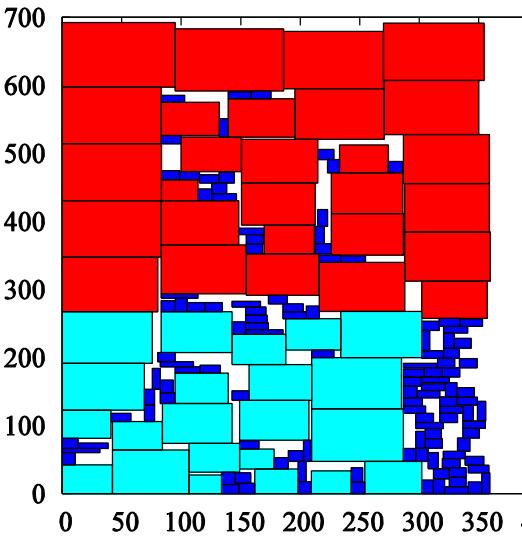


Future Work on Modern Floorplanning (2/4)

- . Floorplanning with multiple supply voltages (voltage islands): ICCAD-06, ICCAD-07



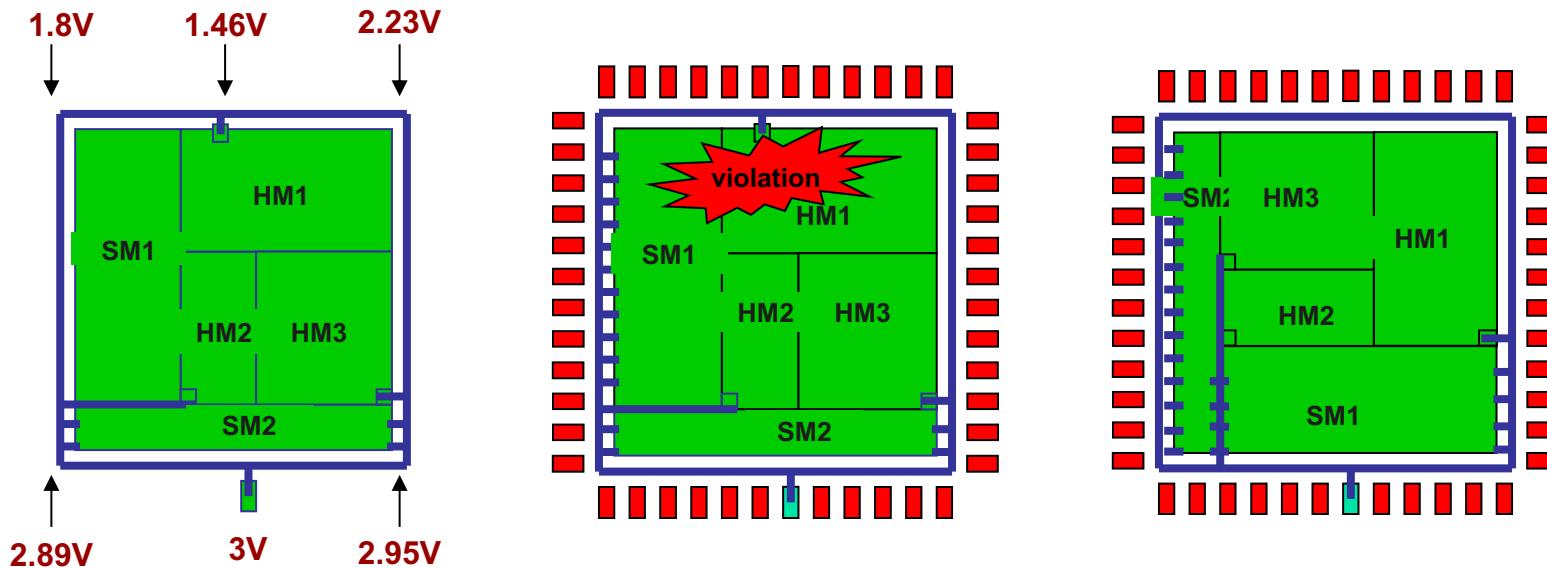
- VDDH power ring — VDDH power line
- VDDL power ring — VDDL power line



- VDDH block
- VDDL block
- Level shifter

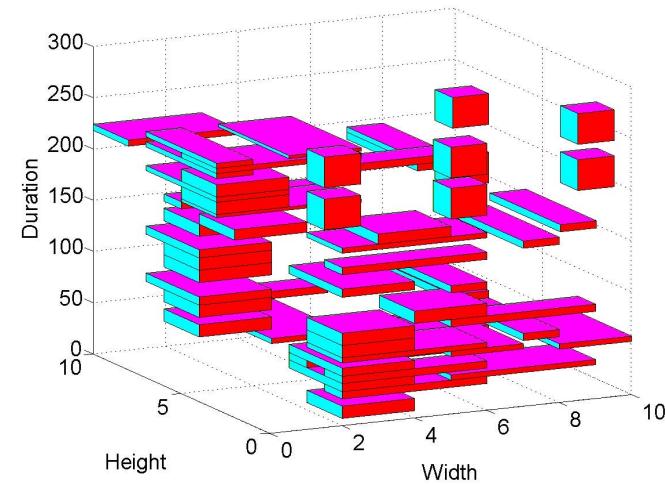
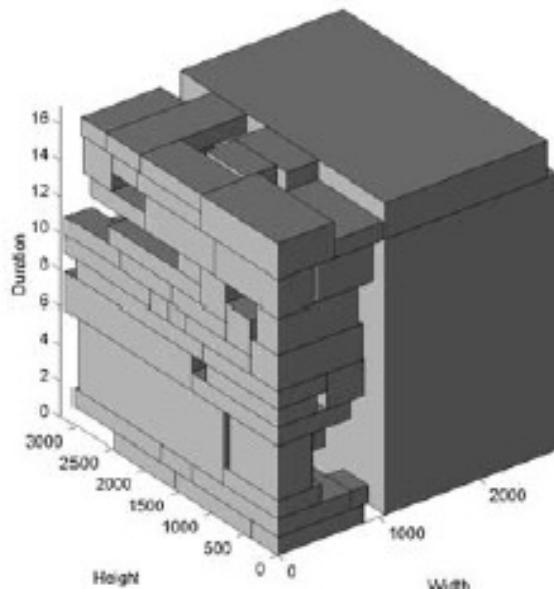
Future Work on Modern Floorplanning (3/4)

- . Floorplan co-synthesized with other circuit components
 - E.g., Power/ground networks for static/dynamic IR drop minimization (ISPD-06, ICCAD-09), buffer blocks for timing optimization, level shifters for multiple supply voltage designs (ICCAD-07)



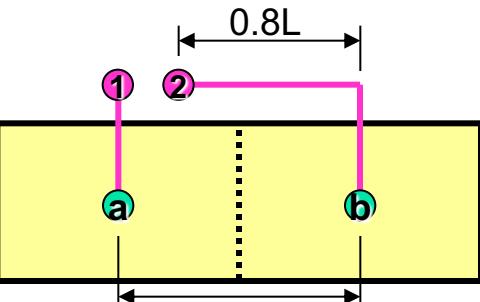
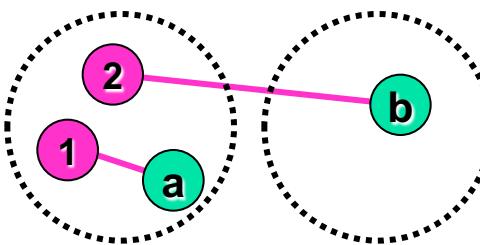
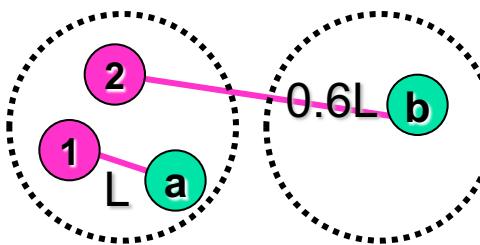
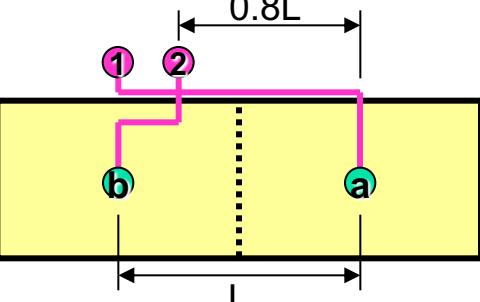
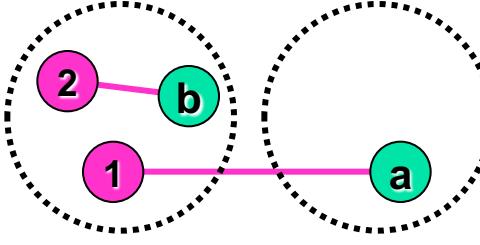
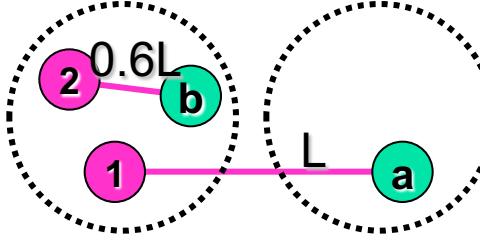
Future Work on Modern Floorplanning (4/4)

- . Floorplanning for reconfigurable computing & for digital microfluidic biochips (ICCAD-04, DAC-06, TCAD-07)
 - Scheduling is added into the picture
- . 2.5D/3D floorplanning for system-in-packages (SiP's) & 3D IC's (ASPDAC-10)
 - Layer partitioning needs to consider vertical vias for thermal & interconnection considerations



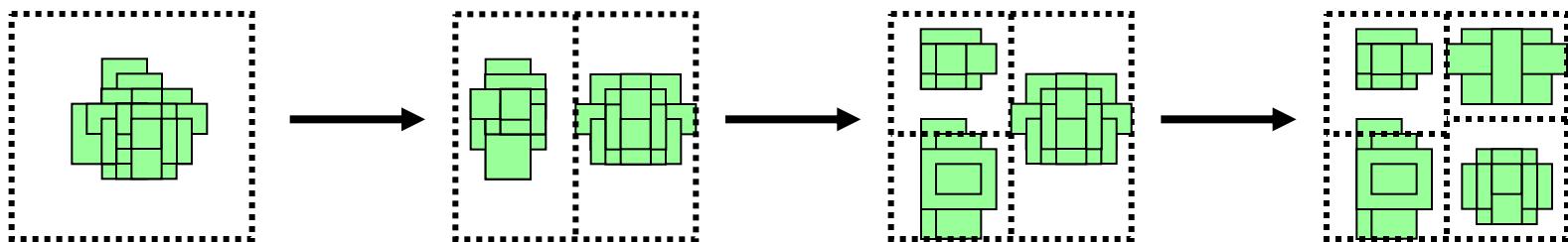
Appendix A:

Exact Net-Weight Partitioning

Physical Partitions	Traditional Terminal Propagation	Exact Net-Weight Modeling
 <p>$HPWL_x = 0.8L$</p>	 <p>Cutsize = 1</p>	 <p>Cut weight = 0.6L</p>
 <p>$HPWL_x = 1.2L$</p>	 <p>Cutsize = 1</p>	 <p>Cut weight = 1.0L</p>

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.

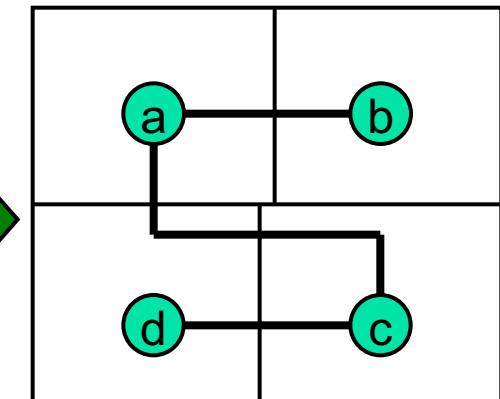
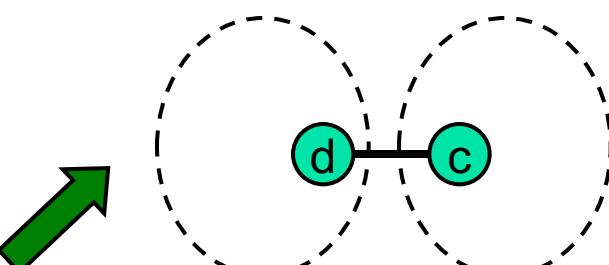
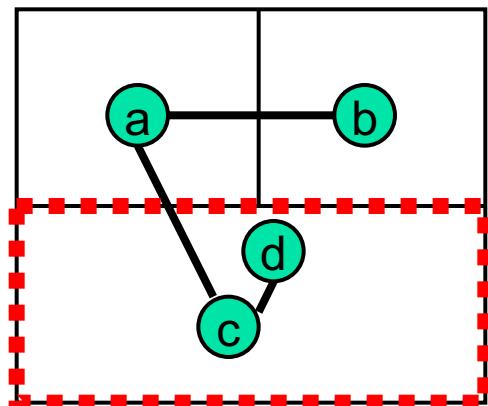


- The partitioning stage continues until the number of modules in each partition is smaller than a threshold, and the partitioned floorplan is obtained.

Terminal Propagation for Wirelength Optimization

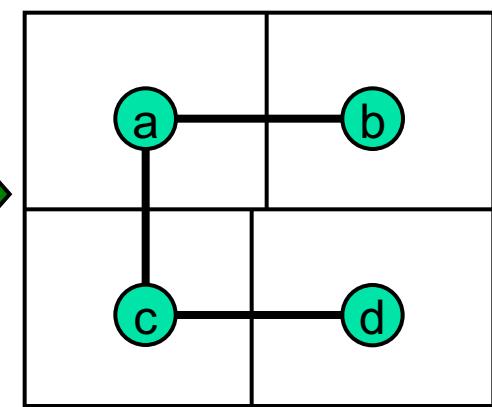
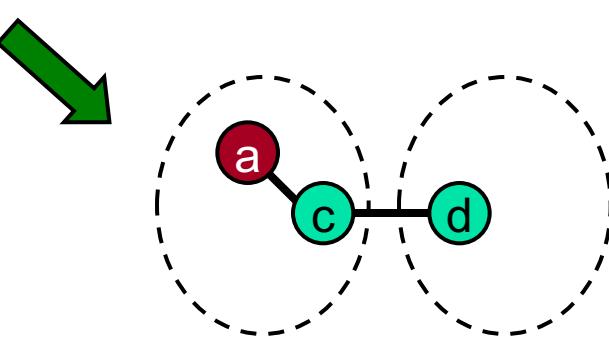
Need to consider the interconnections among subregions!

A possible solution **without**
using terminal propagation



**With terminal propagation
(Node a is a fixed node)**

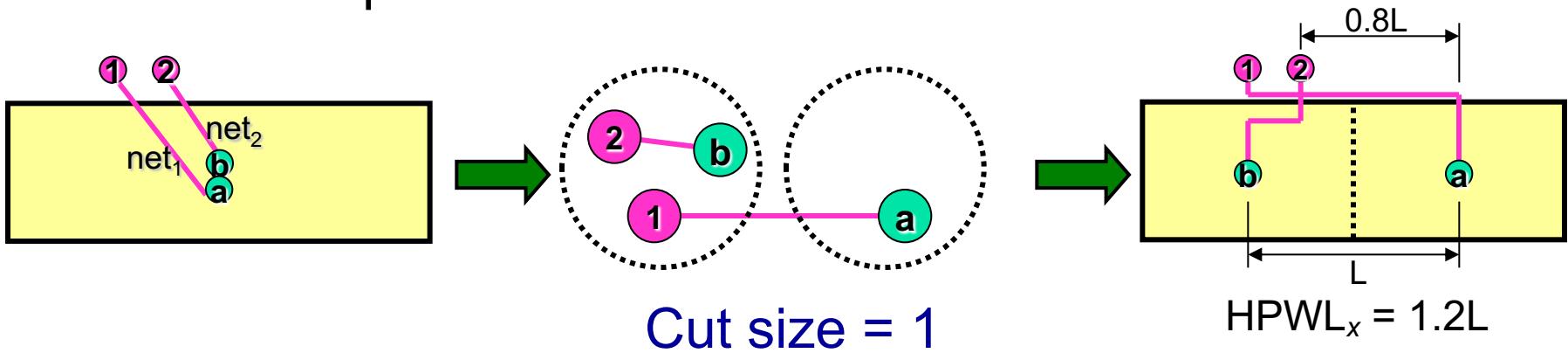
Courtesy of Prof. Y.-W. Chang and H.-M. Chen



Obtain a better result

Traditional Terminal Propagation

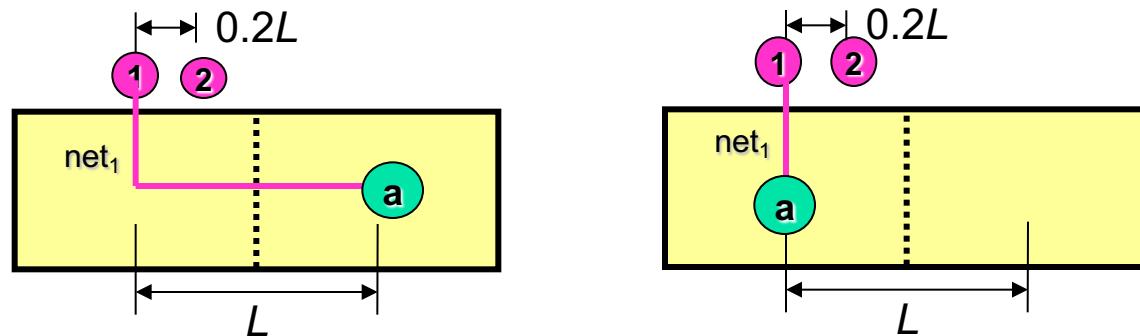
- For traditional terminal propagation, the min-cut is *not* equivalent to finding the minimum HPWL exactly
- An example that traditional terminal propagation leads to a sub-optimal solution:



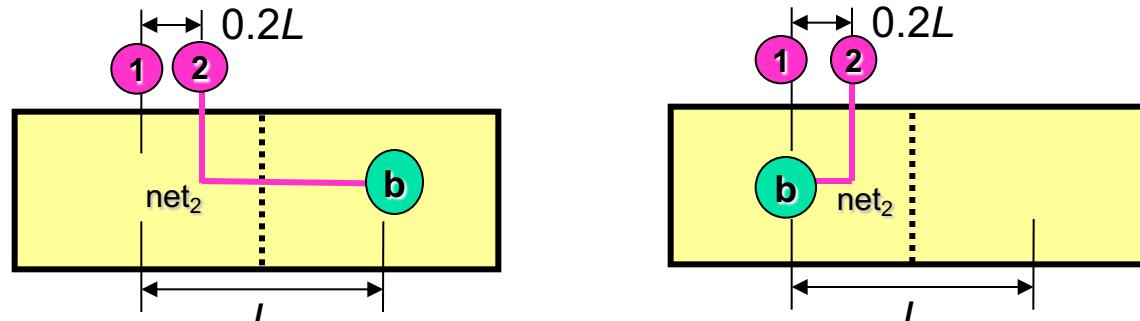
- Problem: hyperedge weight is a **constant** value!
 - Shall map the min-cut cost to HPWL change
 - Shall assign the hyperedge weight as the value of the HPWL contribution if the hyperedge is cut

Net Weight Assignment

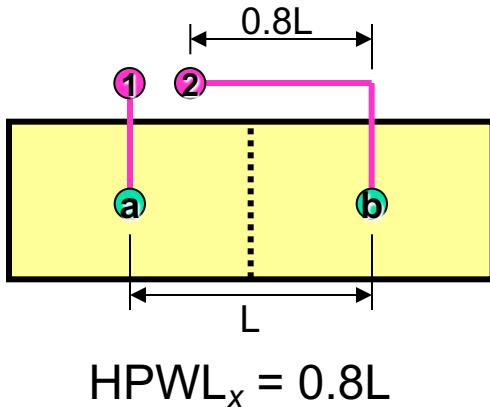
- . net₁ connects a movable node **a** and a fixed node **1**.
$$\begin{aligned}\text{Weight(net}_1\text{)} &= \text{Length(net}_1 \text{ is cut)} - \text{Length(net}_1 \text{ is not cut)} \\ &= L - 0L = L\end{aligned}$$



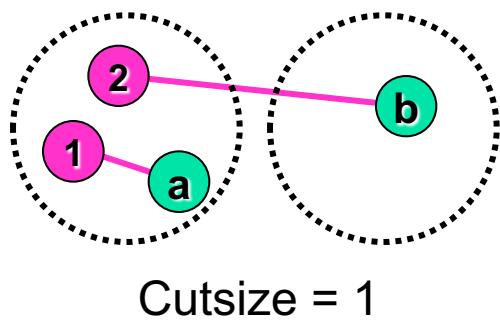
- . net₂ connects a movable node **b** and a fixed node **2**.
$$\begin{aligned}\text{Weight(net}_2\text{)} &= \text{Length(net}_2 \text{ is cut)} - \text{Length(net}_2 \text{ is not cut)} \\ &= 0.8L - 0.2L = 0.6L\end{aligned}$$



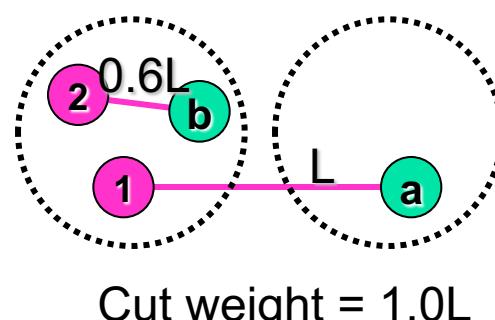
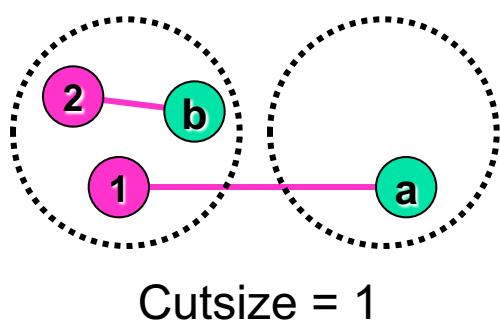
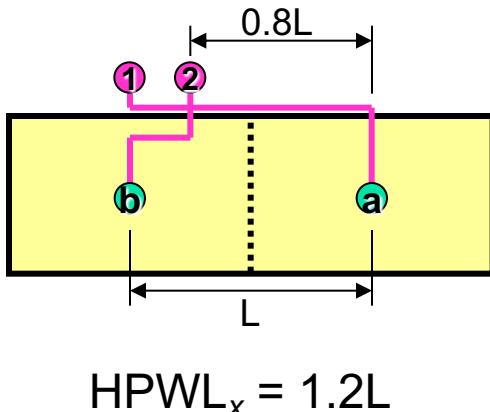
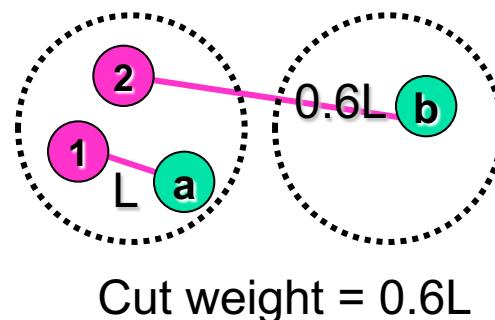
Physical Partitions



Traditional Terminal Propagation



Exact Net-Weight Modeling



Cut weight is proportional to the HPWL!

$$HPWL = Cut\ weight + 0.2L$$

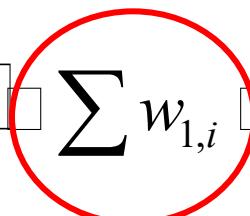
(0.2L is the lower bound for the net2 HPWL)

Relationship Between HPWL and Cut Weight

- Theorem: $HPWL_i = w_{1,i} + n_{cut,i}$
 - $n_{cut,i}$: cut weight for net i
 - $w_{1,i}$: the HPWL lower bound for net i

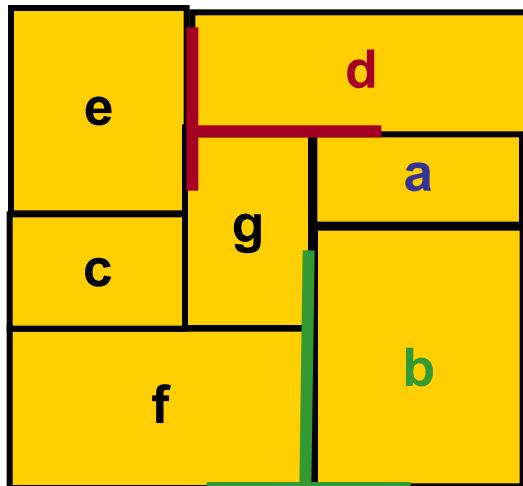
Constant!

- Then, we have

$$\min \left[\sum HPWL_i \right] \leq \min \left[\sum (w_{1,i} + n_{cut,i}) \right] \leq \sum w_{1,i} + \min \left[\sum n_{cut,i} \right]$$


Finding the minimum HPWL is equivalent to finding the min-cut!!

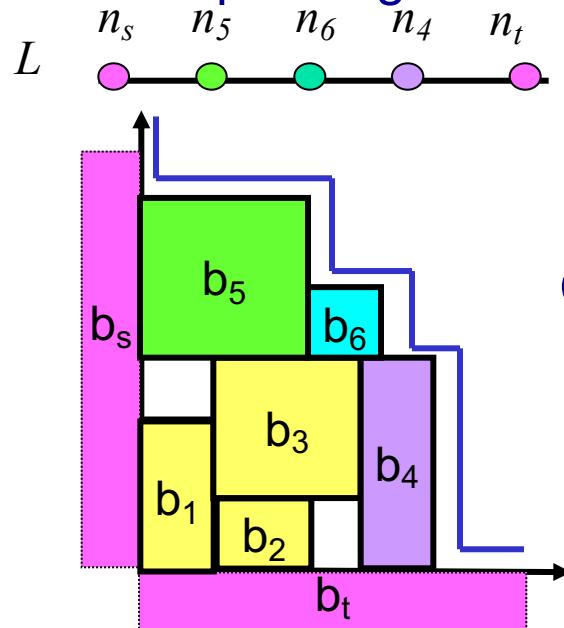
Appendix B: Other Floorplan Representations



$S = (\text{fcegbad})$, $L = (001\textcolor{blue}{1}00)$, $T = (0010\textcolor{green}{1}00\textcolor{red}{1}0)$

Corner Sequence (CS)

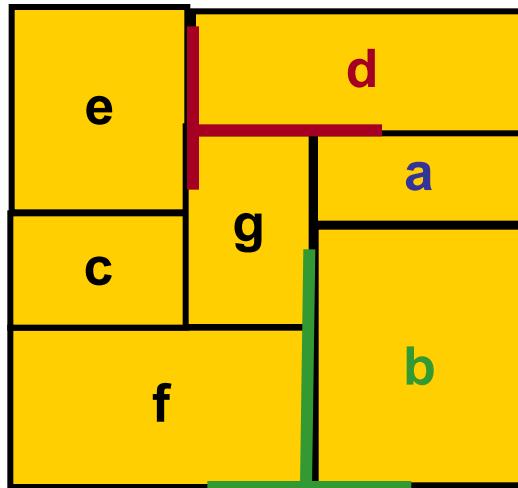
- Lin, Chang, Lin, “Corner sequence: A P-admissible floorplan representation with linear-time packing scheme,” IEEE TVLSI 2003.
- Sequence of modules and their corresponding corners CS = $\langle (S_1, D_1), (S_2, D_2), \dots, (S_m, D_m) \rangle$
 - S_i : a module
 - D_i : the corresponding bend for packing S_i



$$\text{CS} = \langle (b_1, [b_s, b_t]), (b_2, [b_1, b_t]), (b_3, [b_1, b_2]), (b_4, [b_3, b_t]), (b_5, [b_s, b_3]), (b_6, [b_5, b_4]) \rangle$$

Corner Block List (CBL)

- Hong, et. al., “Corner block list: An effective and efficient topological representation of non-slicing floorplan,” ICCAD-2K.
- Each room contains one and only one block (**mosaic** floorplan).
- CBL = (S, L, T):
 - S: sequence of corner modules.
 - L: List of module orientations (0: vertical T-junction; 1: horizontal one).
 - T: list of T-junction information (# of T-junctions with the corner block).



$$S = (\text{fcegbad}), L = (001100), T = (001010010)$$

Appendix C:

Bus-Driven Floorplanning

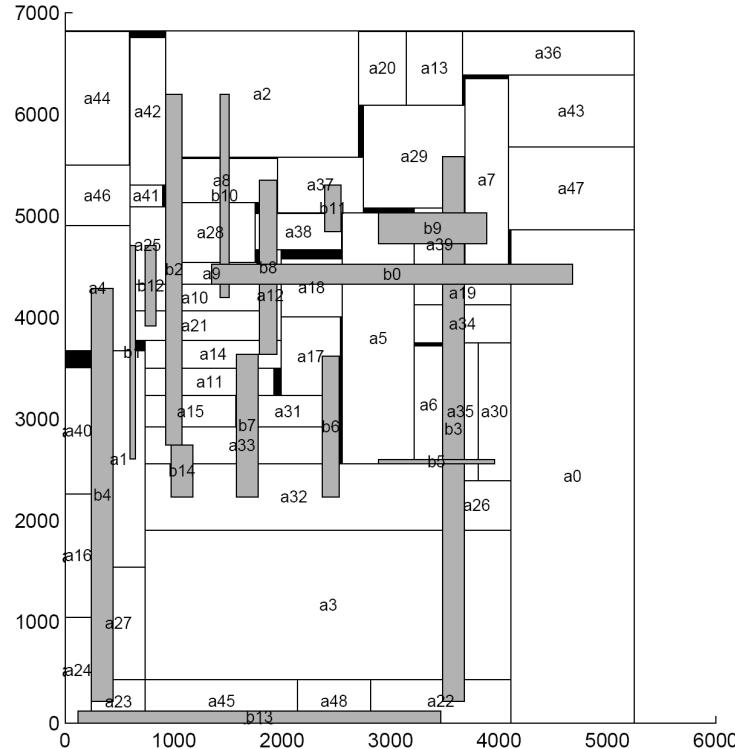
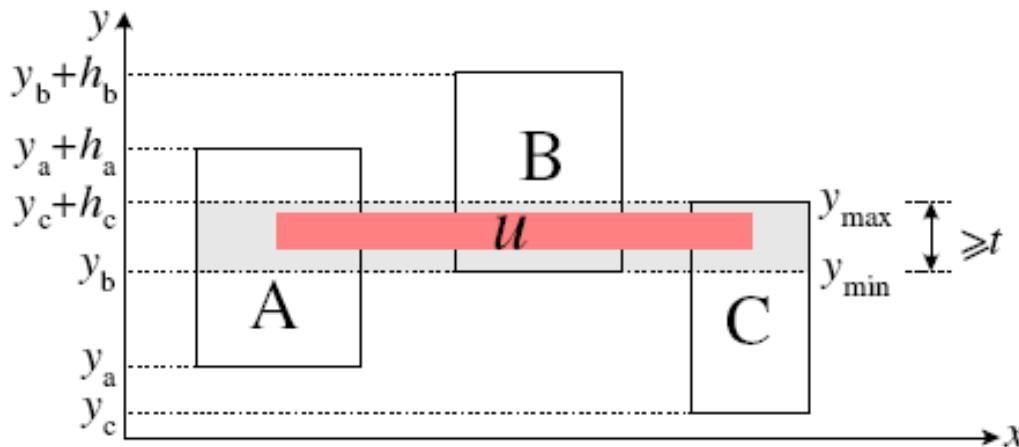


Fig. 10. The resulting packing of ami49-3 with soft block adjustment. There are 49 blocks and 15 buses. The buses are $\{0, 5, 9, 12, 18\}$, $\{1, 10, 21, 25\}$, $\{2, 28, 33\}$, $\{3, 19, 22, 26, 29, 34\}$, $\{4, 23, 27\}$, $\{5, 35, 30, 6\}$, $\{32, 31, 17\}$, $\{11, 14, 15, 32, 33\}$, $\{12, 8, 14\}$, $\{5, 7, 39\}$, $\{2, 8, 9, 10\}$, $\{37, 38\}$, $\{10, 21, 25\}$, $\{22, 23, 24\}$, $\{32, 33\}$.

Bus-Driven Floorplanning

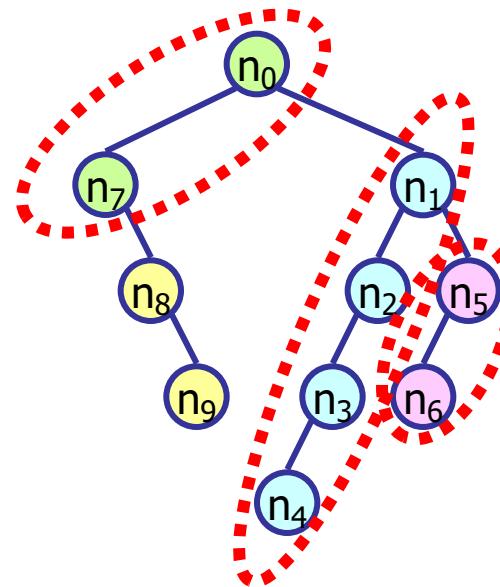
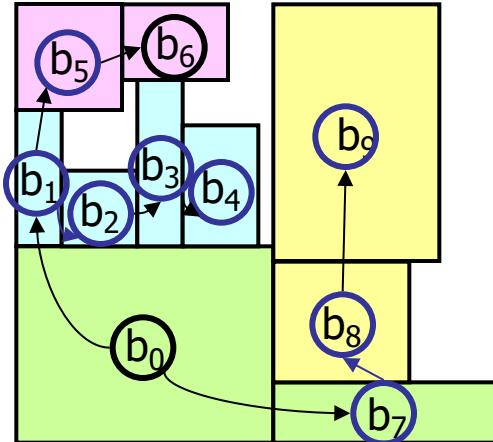
- . Given n rectangular blocks $B = \{ b_i \mid i = 1, \dots, n \}$ and m buses $U = \{ u_i \mid i = 1, \dots, m \}$, each bus u_i has the width t_i and goes through a set of blocks B_i .
 - Decide the positions of blocks and buses so that bus u_i goes through all of its blocks.
 - Minimize the **chip/bus area**.
 - No overlap between any two blocks or between any two horizontal (vertical) buses.



- A feasible horizontal bus $u = \langle H, t, \{ A, B, C \} \rangle$.
- $y_{\max} = y_c + h_c$
- $y_{\min} = y_b$
- $y_{\max} - y_{\min} \geq t$

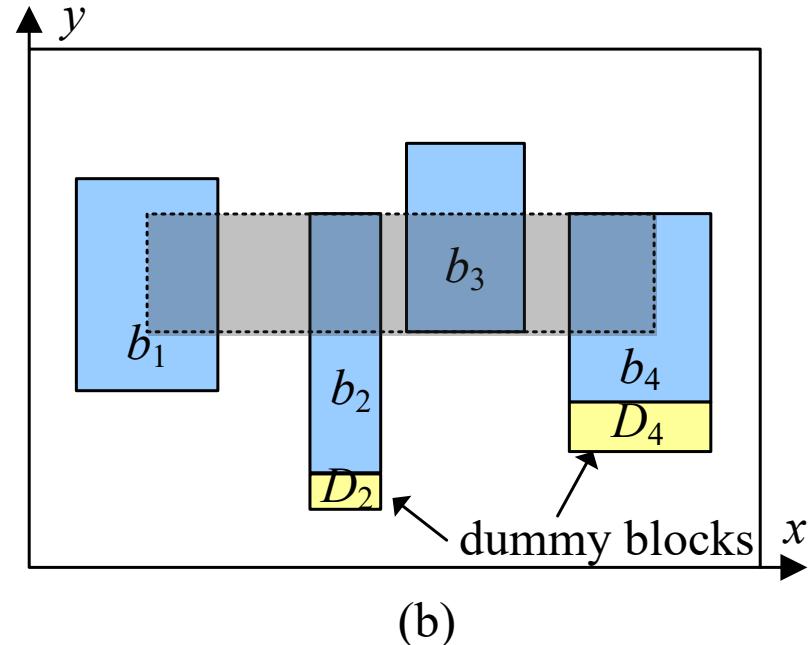
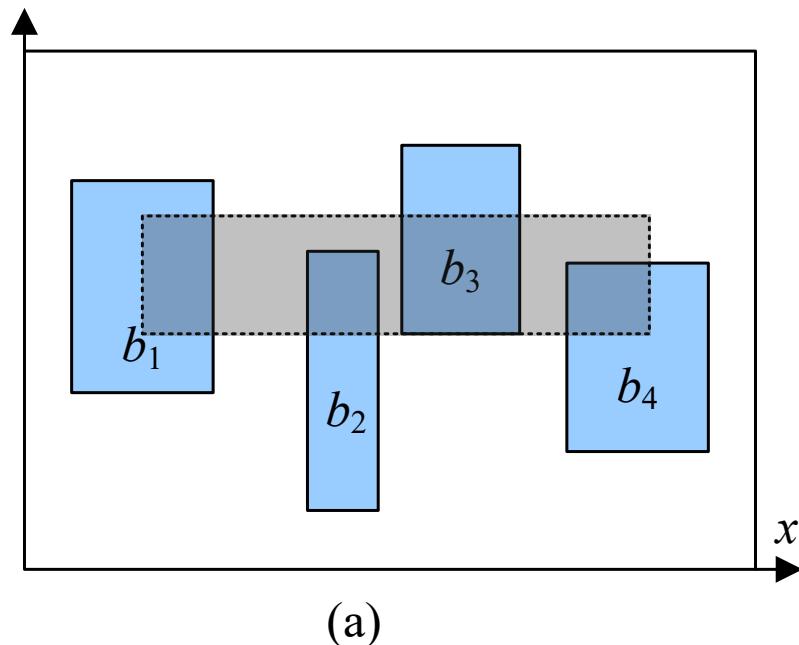
B*-tree Properties for Horizontal Buses

- . Left child
 - The lowest, adjacent block on the right ($x_j = x_i + w_i$)
- 1. In a B*-tree, the nodes in a *left-skewed sub-tree* may satisfy a **horizontal bus** constraint.



Dummy Blocks for Bus Alignment

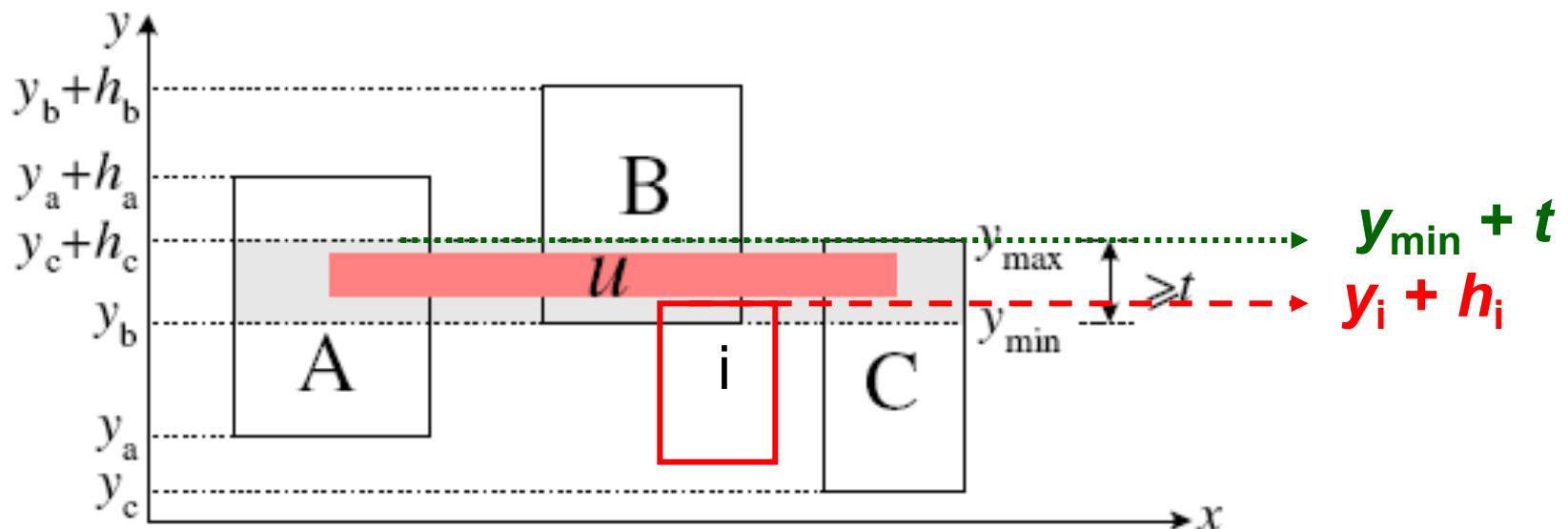
2. Inserting **dummy blocks** of appropriate heights, we can guarantee a horizontal bus with blocks whose corresponding B*-tree nodes are in a left-skewed subtree



Dummy-Block Height Determination

- The height of the dummy block D_i :

$$\Delta_i = \begin{cases} (y_{min} + t) - (y_i + h_i) & \text{if } (y_{min} + t) > (y_i + h_i) \\ 0 & \text{otherwise} \end{cases}$$

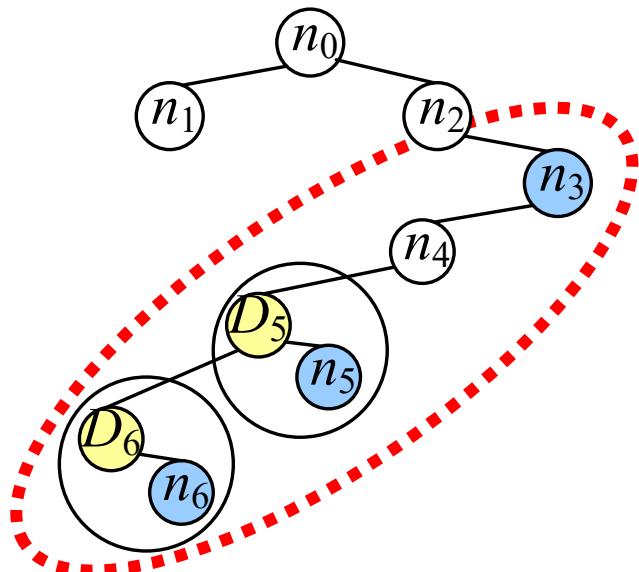


Example Dummy Blocks

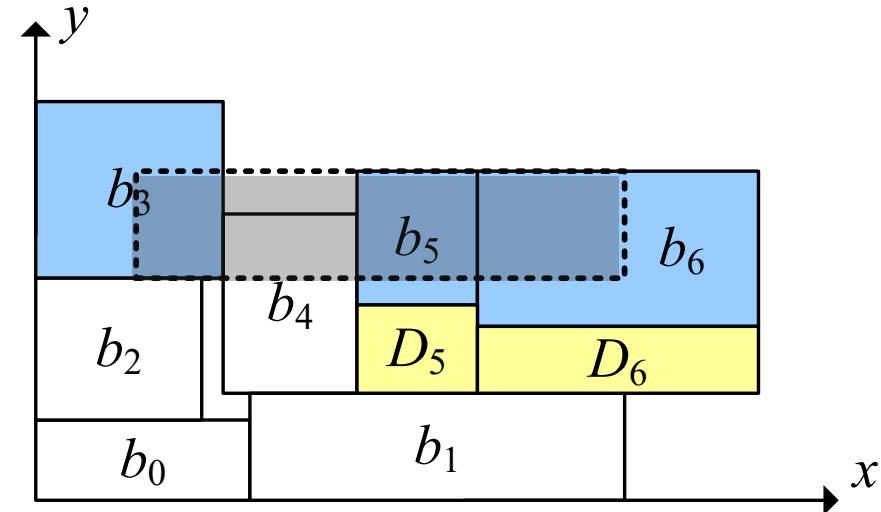
- The height of the dummy block D_i :

$$\Delta_i = \begin{cases} (y_{min} + t) - (y_i + h_i) & \text{if } (y_{min} + t) > (y_i + h_i) \\ 0 & \text{otherwise} \end{cases}$$

- An example of inserting dummy blocks to satisfy a horizontal bus:



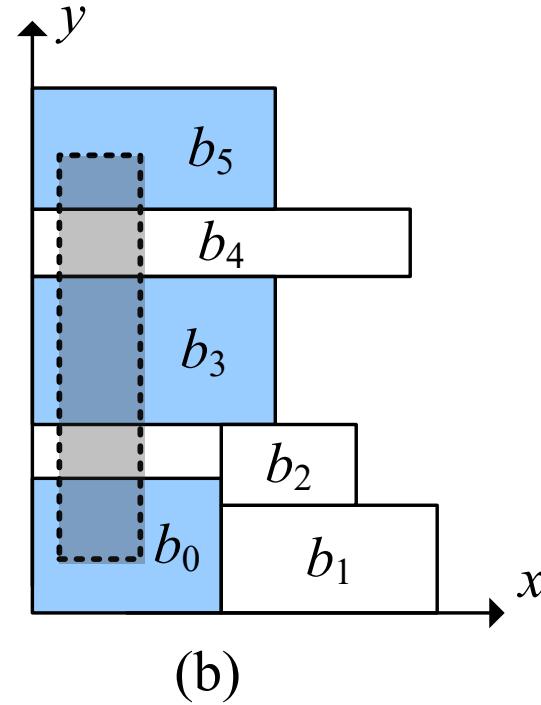
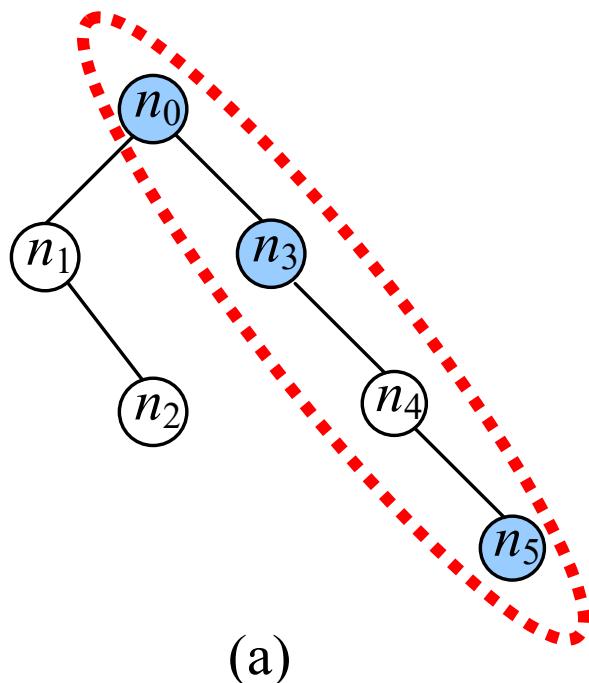
(a)



(b)

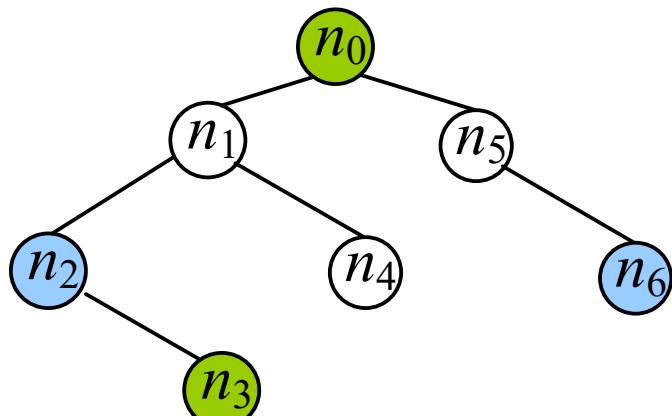
B*-tree Properties for Vertical Buses

- Right child
 - The first block above, with the same x-coordinate ($x_j = x_i$).
3. In a B*-tree, the nodes in a *right-skewed sub-tree* can guarantee the feasibility of a vertical bus.

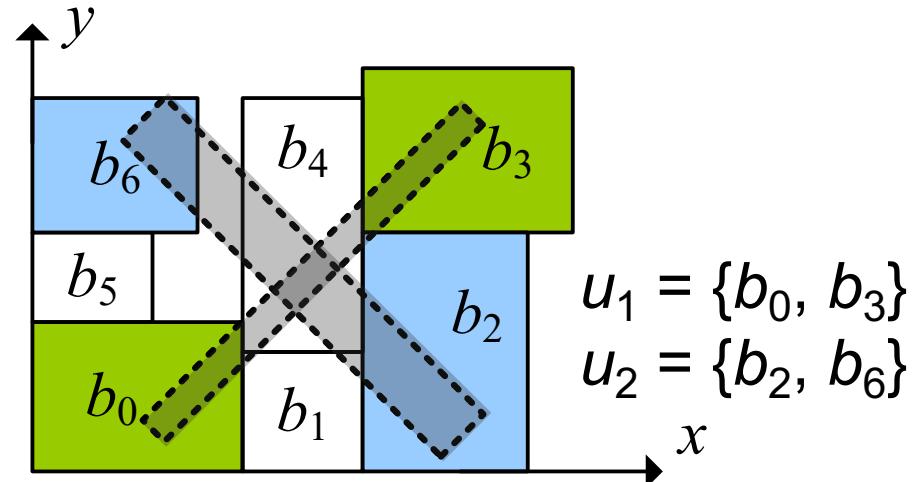


Infeasible Twisted-Bus Structure

- Consider two buses simultaneously, we cannot always fix the horizontal bus constraint by inserting dummy blocks
- Should discard such a tree configuration



(a)

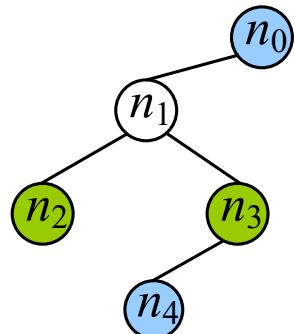


(b)

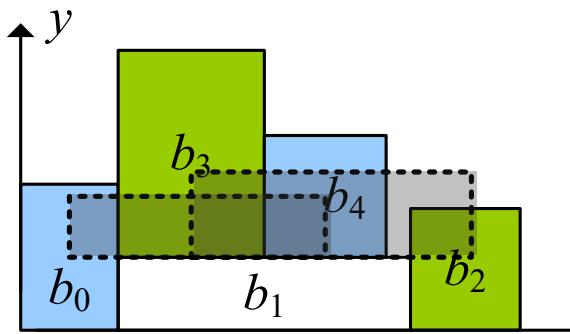
n_3 is in the right-skewed subtree of n_2 ,
and n_6 is also in the right-skewed subtree of n_0

Bus Overlapping

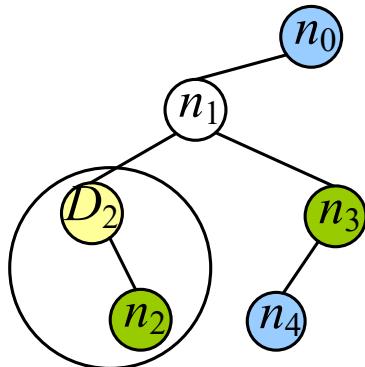
- Use dummy blocks to avoid bus overlapping while considering multiple buses.



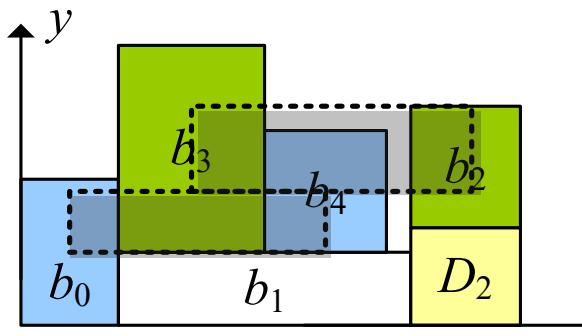
(a)



(b)



(c)



(d)

$$u_1 = \{b_0, b_4\}$$
$$u_2 = \{b_2, b_3\}$$

$$u_1 = \{b_0, b_4\}$$
$$u_2 = \{b_2, b_3\}$$

Cost Function for BDF

- . Use simulated annealing to search for a desired solution.
 - Cost function:

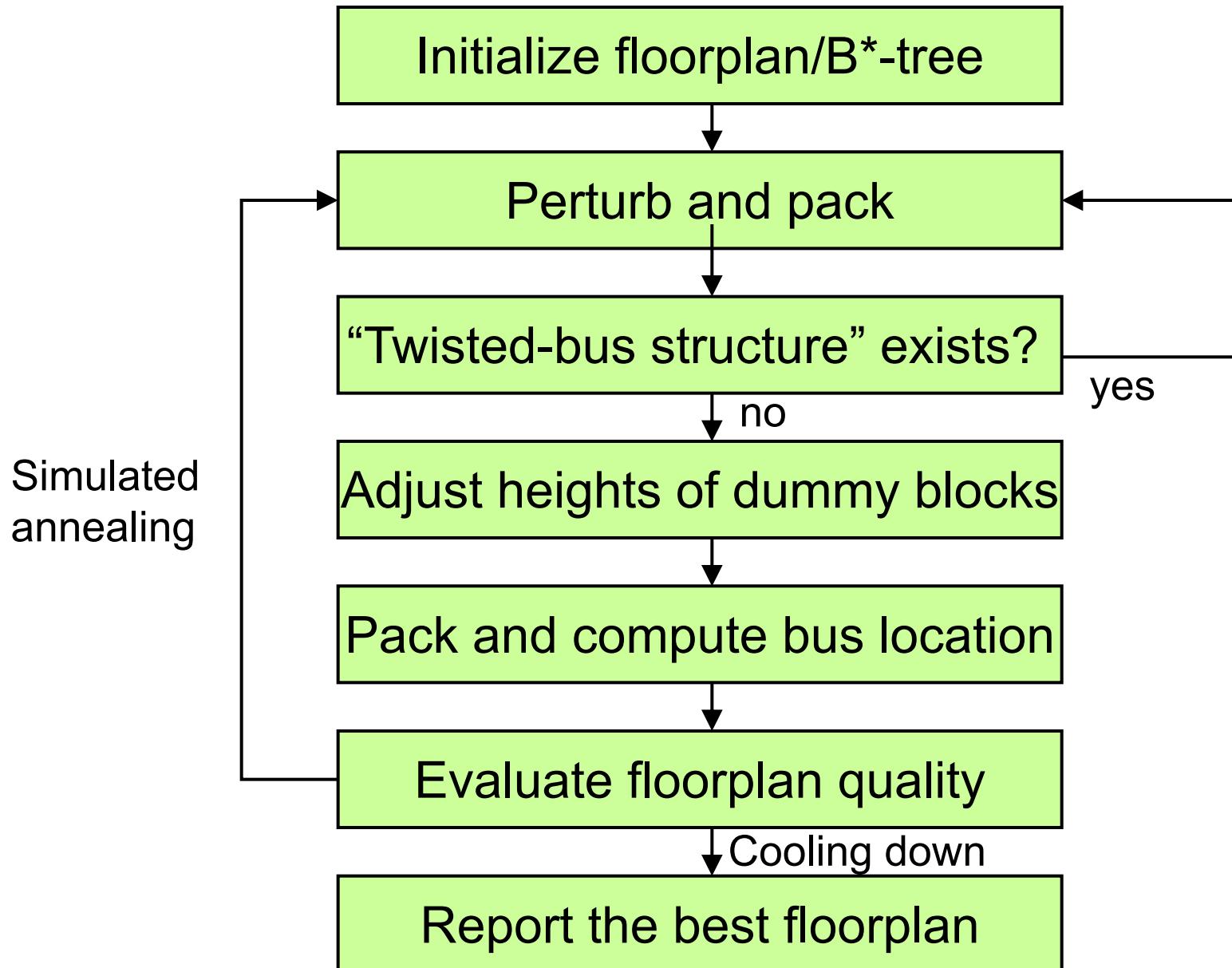
$$\Psi(F, U) \square \alpha A \square \beta B \square \gamma M$$

A chip area

B bus area

M number of unassigned buses

The BDF Algorithm



Bus-Driven Floorplanning Results

- MCNC benchmark on Pentium 4 2.8GHz. Obtain 20% (55%) less dead space on average for hard (soft) blocks.

Block type			Hard Macro Blocks				Soft Macro Blocks			
Circuits	Block #	Bus #	SP: Xiang et al. (ICCAD 2003)		B*-tree: Chen & Chang (ISPD-05)		Xiang et al. (ICCAD 2003)		Chen & Chang (ISPD-05)	
			Time (sec)	Dead space	Time (sec)	Dead space	Time (sec)	Dead space	Time (sec)	Dead space
apte	9	5	11	4.11%	8	1.59%	12	0.72%	3	0.02%
xerox	10	6	12	3.88%	5	3.85%	13	0.95%	6	0.10%
hp	11	14	28	5.02%	20	4.47%	28	0.62%	11	0.03%
ami33-1	33	8	61	6.02%	19	5.69%	62	0.94%	35	0.33%
ami33-2	33	18	81	6.10%	22	3.87%	86	1.27%	35	0.73%
ami49-1	49	9	98	5.42%	28	5.34%	101	0.85%	65	0.51%
ami49-2	49	12	278	6.09%	43	5.45%	281	0.84%	90	0.67%
ami49-3	49	15	265	7.40%	66	4.74%	268	1.09%	109	0.92%
Average			104	5.51%	26	4.38%	106	0.91%	47	0.41%

*SP: Hua Xiang, Xiaoping Tang, and Martin D.F. Wong, *ICCAD 2003*.

The platform of SP is Intel Xeon 2.4GHz

Example Bus-Driven Floorplan

- MCNC ami49-3 with soft block adjustment.
- It has 49 modules and 15 buses.

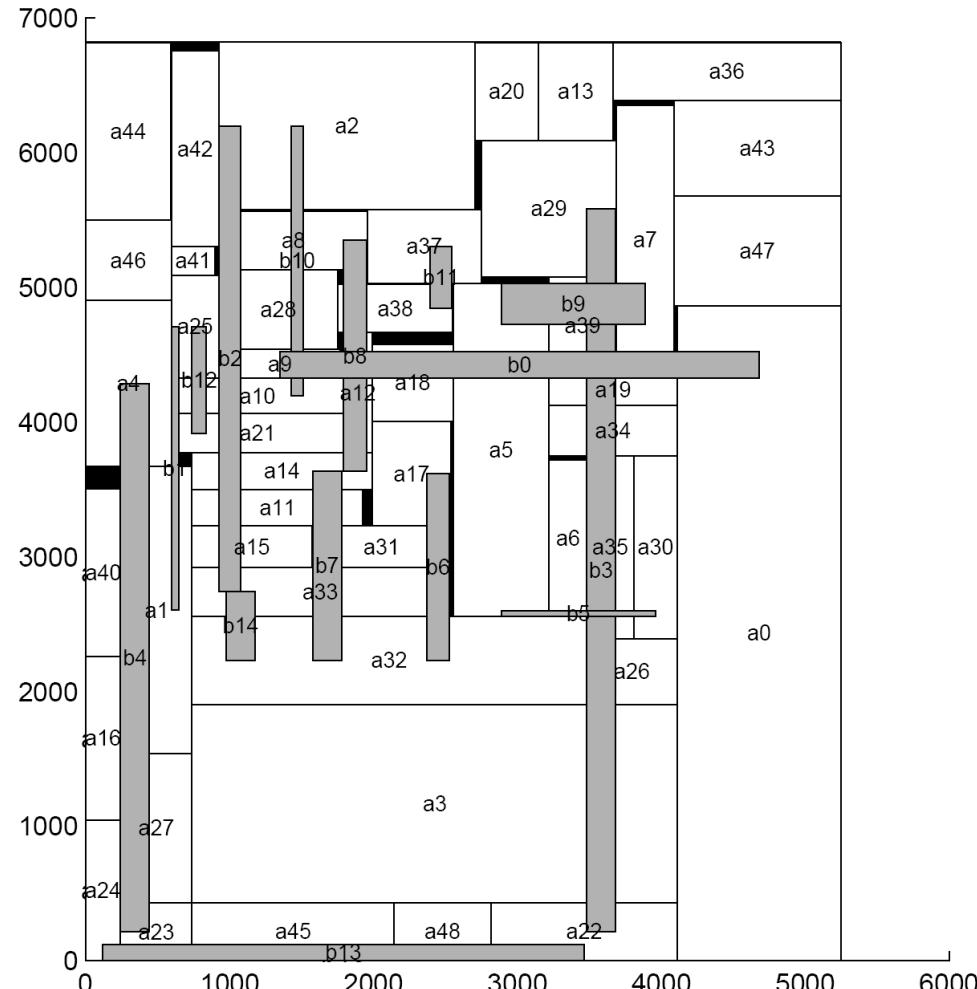


Fig. 10. The resulting packing of ami49-3 with soft block adjustment. There are 49 blocks and 15 buses. The buses are {0, 5, 9, 12, 18}, {1, 10, 21, 25}, {2, 28, 33}, {3, 19, 22, 26, 29, 34}, {4, 23, 27}, {5, 35, 30, 6}, {32, 31, 17}, {11, 14, 15, 32, 33}, {12, 8, 14}, {5, 7, 39}, {2, 8, 9, 10}, {37, 38}, {10, 21, 25}, {22, 23, 24}, {32, 33}.