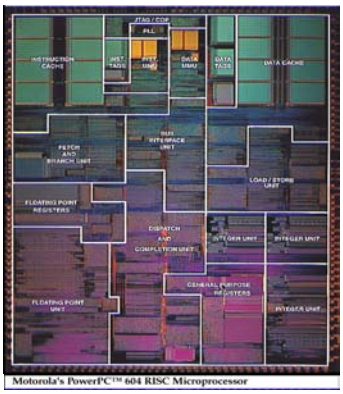
Unit 4: Floorplanning

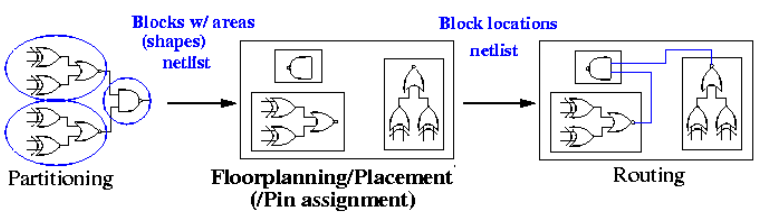
- Course contents:
  - Normalized polish expression for slicing floorplans
  - Sequence pair for general floorplans
  - B*-trees for compacted and large-scale floorplans
  - Comparisons on recently developed floorplan representations
  - ILP for general floorplans
- Readings
  - S&Y: Chapter 3
  - Sherwani: Chapter 6



Motorola's PowerPC™ 604 RISC Microprocessor

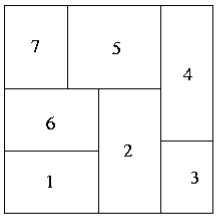
Floorplanning/Placement

- 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
  - Find **locations** for all blocks.
  - Consider shapes of soft block and pin locations of all the blocks.

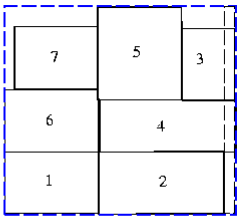


Floorplanning Problem

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

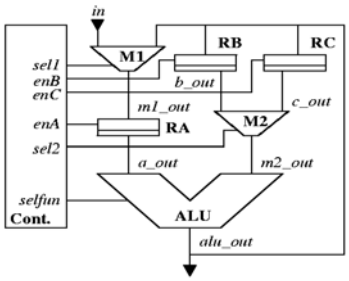


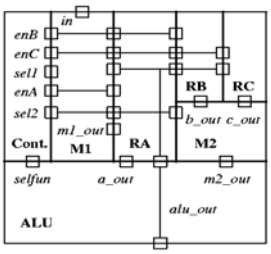
An optimal floorplan, in terms of area

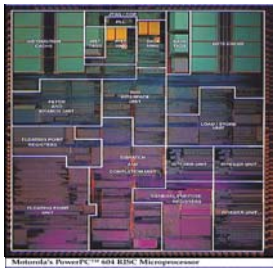


A non–optimal floorplan

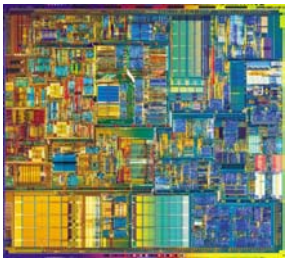
Floorplan Examples







PowerPC 604

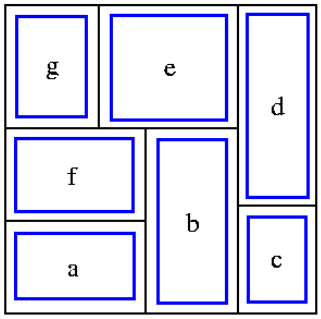


Pentium 4

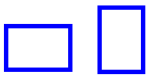
Early Layout Decision Methodology

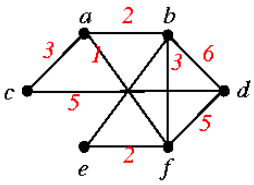
- An IC is a 2-D medium; consider the dimensions of blocks in early stages of the design helps to improve the quality.
- Floorplanning gives early feedback
  - Suggests valuable architectural modifications
  - Estimates the whole chip area
  - Estimates delay and congestion due to wiring
- Floorplanning fits very well in a *top-down* design strategy; the *step-wise refinement* strategy also propagated in software design.
- Floorplanning considers the *flexibility* in the shapes and terminal locations of blocks.

Floorplan Design



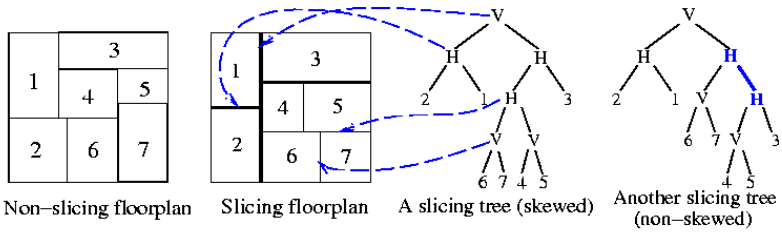
- *Modules:* 

- *Area: A=xy*
- *Aspect ratio: r <= y/x <= s*
- *Rotation:* 
- *Module connectivity*



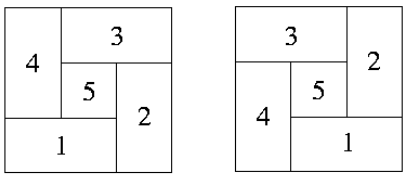
Slicing Floorplan Structure

- **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.

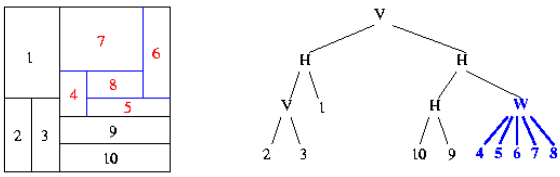


Floorplan Order

- **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.



The two possible wheels.

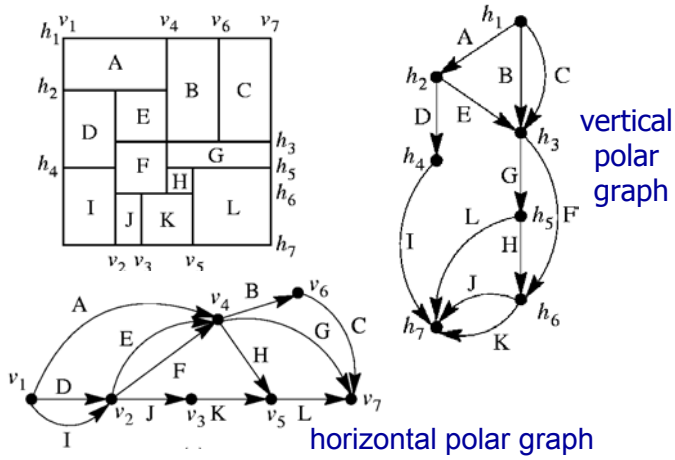


A floorplan of order 5

Corresponding floorplan tree

Polar Graphs for General Floorplans

- Ohtsuki, Suzigama, and Hawanishi, “An optimization technique for integrated circuit layout design,” ICCST-70.
- vertex: channel segment; edge (weight): block (dimension)
- **Question: How to manipulate the graphs?**



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
  - neighborhood structure
  - cost function
  - annealing schedule

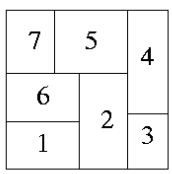
Solution Representation

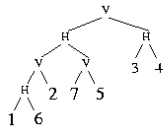
- An expression $E = e_1\, e_2 \ldots e_{2n-1}$, where $e_i \in \{1, 2, \ldots, n, H, V\}$, $1 \le i \le 2n-1$, is a **Polish expression** of length $2n$-1 iff
  1. every operand j, $1 \le j \le n$, appears exactly once in E;
  2. (**the balloting property**) for every subexpression $E_i = e_1 \ldots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

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

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

- Polish expression ↔ Postorder traversal.
- *ijH*: rectangle *i* on bottom of *j*; *ijV*: rectangle *i* on the left of *j*.



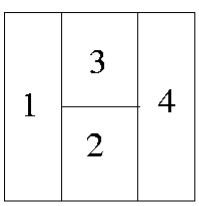


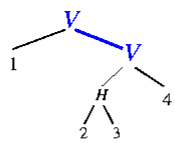
E = 16H2V75VH34HV

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

Postorder traversal of a tree!

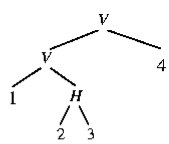
Redundant Representation





E = 123H4VV

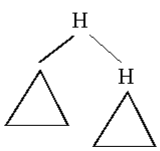
non−skewed!



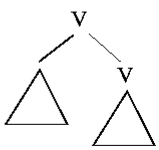
E = 123HV4V

skewed!

Non−skewed cases



..... HH .....

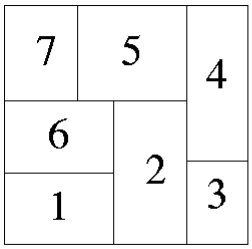


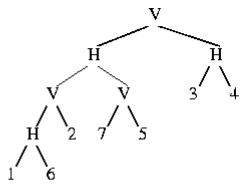
..... VV .....

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

Normalized Polish Expression

- A Polish expression $E = e_1\ e_2 \ldots e_{2n-1}$ is called **normalized** iff E has no consecutive operators of the same type (H or V).
- Given a **normalized** Polish expression, we can construct a **unique** rectangular slicing structure.



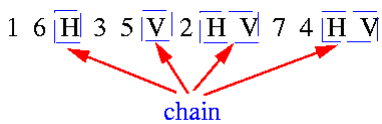


E = 16H2V75VH34HV

A normalized Polish expression

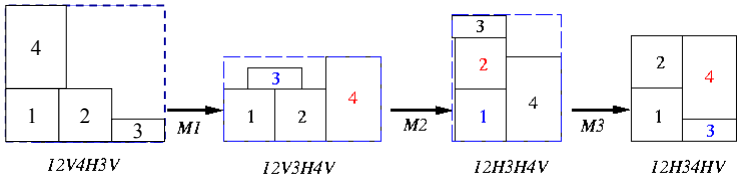
Neighborhood Structure

- **Chain:** *HVHVH* … or *VHVHV* …



- **Adjacent:** 1 and 6 are adjacent operands; 2 and 7 are adjacent operands; 5 and *V* are adjacent operand and operator.
- 3 types of moves:
  - *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.

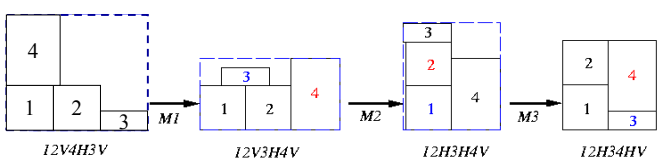
Effects of Perturbation



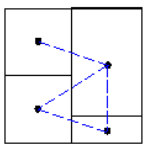
- **Question:** The balloting property holds during the moves?
  - *M1* and *M2* moves are OK.
  - **Check the *M3* moves!** Reject "illegal" *M3* moves.
- **Check *M3* moves:** Assume that *M3* swaps the operand e_i with the operator e_{i+1}, $1 \le i \le k$-1. Then, the swap will not violate the balloting property iff $2N_{i+1} < i$.
  - N_k: # of operators in the Polish expression $E = e_1\ e_2 \ldots e_k$, $1 \le k \le 2n$-1

Cost Function

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

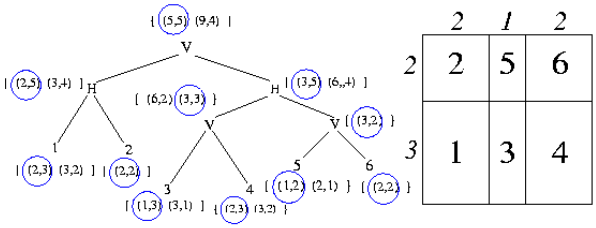


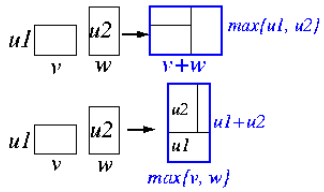
- $W = \sum_{ij} c_{ij}\, d_{ij}$.
  - c_{ij}: # of connections between blocks i and j.
  - d_{ij}: center-to-center distance between basic rectangles i and j.

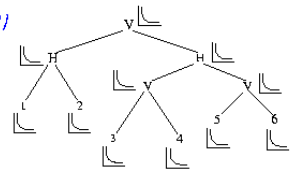


Area Computation for Hard Blocks

- Allow rotation



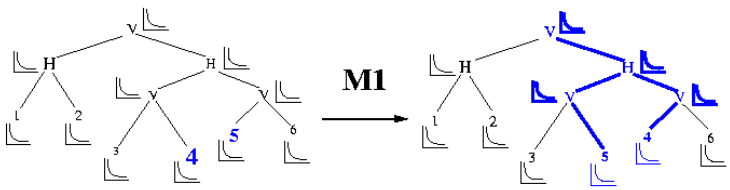




- Wiring cost?
  - Center-to-center interconnection length

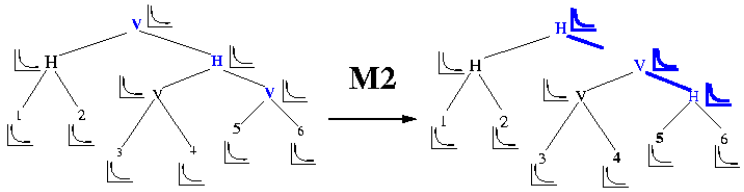
Incremental Computation of Cost Function

- 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.



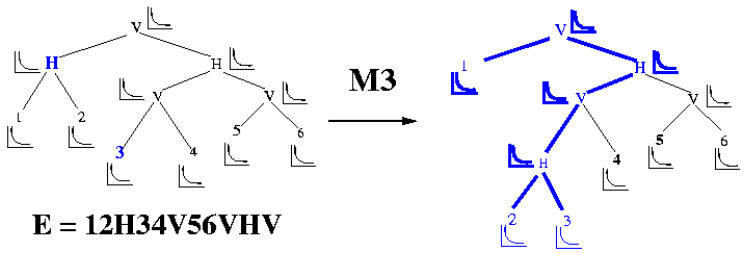
E = 12H34V56VHV E = 12H35V46VHV

Incremental Computation of Cost Function (cont'd)



E = 12H34V56VHV

E = 12H34V56HVH



E = 12H34V56VHV

E = 123H4V56VHV

Annealing Schedule

- Initial solution: 12*V*3*V* … *nV*.

1	2	3		n

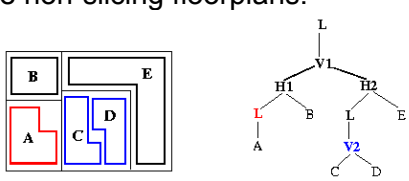
- $T_i = r^i T_0$, i = 1, 2, 3, …; r =0.85.
- At each temperature, try *kn* moves (k = 5-10).
- Terminate the annealing process if
  - # of accepted moves < 5%,
  - temperature is low enough, or
  - run out of time.

Algorithm: Wong-Liu (P, ε, r, k)

```
begin
E ← 12V3V4V … nV; /* initial solution */
Best ← E; T0 ← Δcost_avg / ln(P); M ← MT ← uphill ← 0; N = kn;
repeat
  MT ← uphill ← reject ← 0;
  repeat
    SelectMove(M);
    Case M of
    M1: Select two adjacent operands ei and ej; NE ← Swap(E, ei, ej);
    M2: Select a nonzero length chain C; NE ← Complement(E, C);
    M3: done ← FALSE;
      while not (done) do
          Select two adjacent operand ei and operator ei+1;
          if (ei-1 ≠ ei+1) and (2 Ni+1 < i) then done ← TRUE;
13'       Select two adjacent operator ei and operand ei+1;
14'       if (ei ≠ ei+2) then done ← TRUE;
      NE ← Swap(E, ei, ei+1);
    MT ← MT+1; Δcost ← cost(NE) − cost(E);
    if (Δcost ≤ 0) or (Random < e^(−Δcost/T))
     then
        if (Δcost > 0) then uphill ← uphill + 1;
        E ← NE;
        if cost(E) < cost(best) then best ← E;
     else reject ← reject + 1;
   until (uphill > N) or (MT > 2N);
   T ← rT; /* reduce temperature */
until (reject/MT > 0.95) or (T < ε) or OutOfTime;
end
```

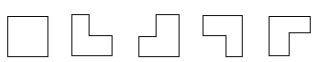

Extension to L-Shaped Modules

- 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 floorplans.

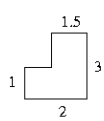


E = A L B H1 C D V2 L E H2 V1 L

Orientation:



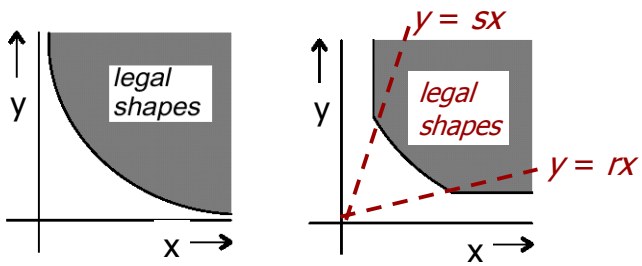
Representation:



(2, 1.5, 3, 1)

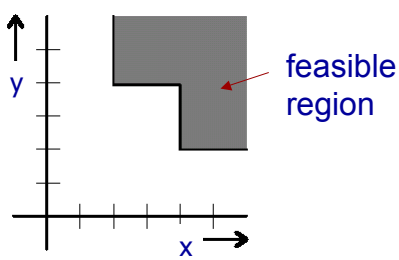
Shape Curve for Floorplan Sizing

- 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: $r \le y/x \le s$.



Shape Curve

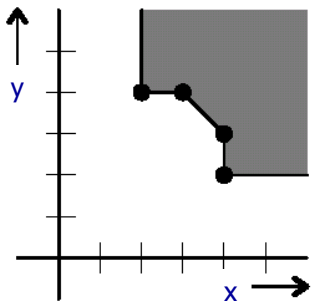
- Since a basic block is built from discrete transistors, it is not realistic to assume that the shape function follows the hyperbola continuously.
- In an extreme case, a block is rigid/hard: it can only be rotated and mirrored during floorplanning or placement.



The shape curve of a 2 × 4 hard block.

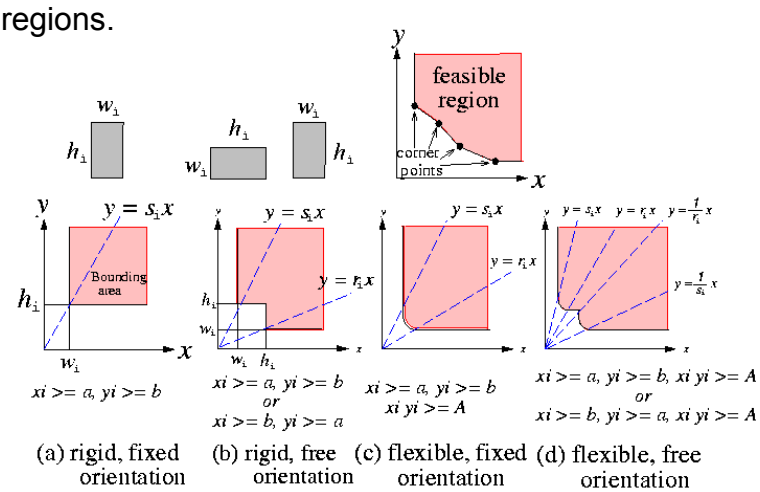
Shape Curve (cont'd)

- 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.



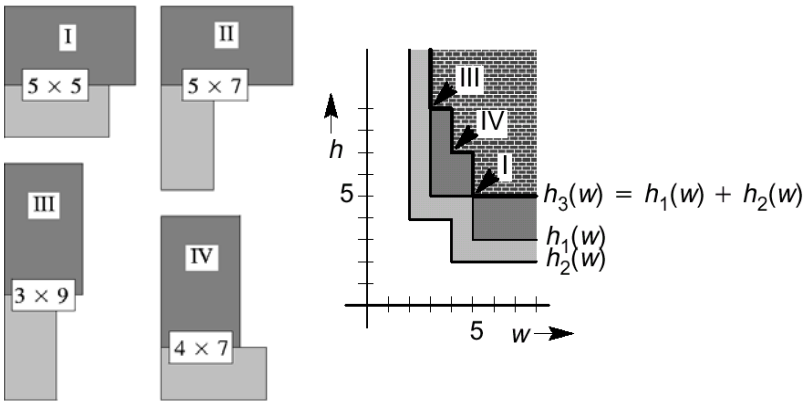
Feasible Implementations

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



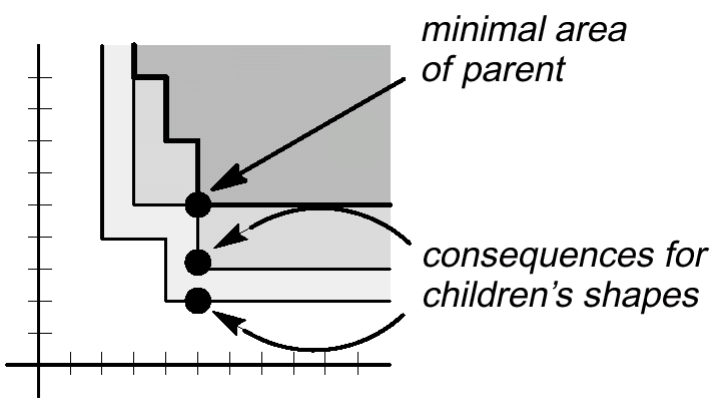
Vertical Abutment

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



Deriving Shapes of Children

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



Slicing Floorplan Sizing

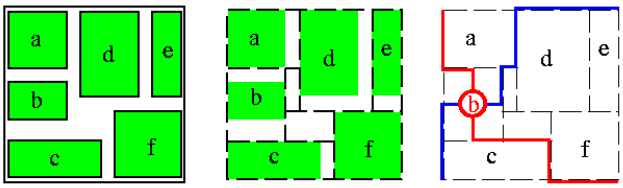
- The shape functions of all leaf blocks are given as piecewise linear functions.
- Traverse the slicing tree to compute the shape functions of all composite blocks (bottom-up composition).
- Choose the desired shape of the top-level block
  - Only the corner points of the function need to be evaluated 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

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-2002)
  5. The relationship between any two blocks is defined in the representation (topological representation).
- Slicing floorplan is **not** P-admissible. Why?
- A P*-admissible floorplan representation: **Sequence Pair.**

Sequence Pair (SP)

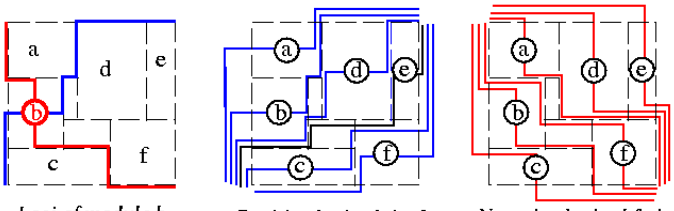
- Murata, Fujiyoshi, Nakatake, Kajitani, "Rectangle-Packing Based Module Placement," ICCAD-95.
- Represent a packing by a pair of module-name sequences (e.g., (*abdecf*, *cbfade*)).
  - Solution space: $(n!)^2$
- Correspond all pairs of the sequences to a P-admissible solution space.
- Search in the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



A floorplan Loci of module b

Relative Module Positions

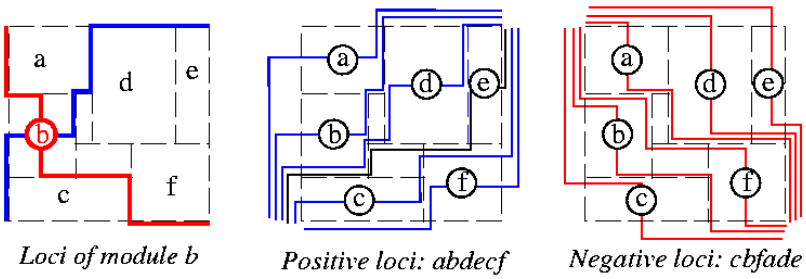
- A floorplan is a partition of a chip into **rooms**, each containing at most one block.
- **Locus** (right-up, left-down, up-left, down-right)
  1. Take a non-empty room.
  2. Start at the center of the room, walk in two alternating directions to hit the sides of rooms.
  3. Continue until to reach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

Geometrical Information

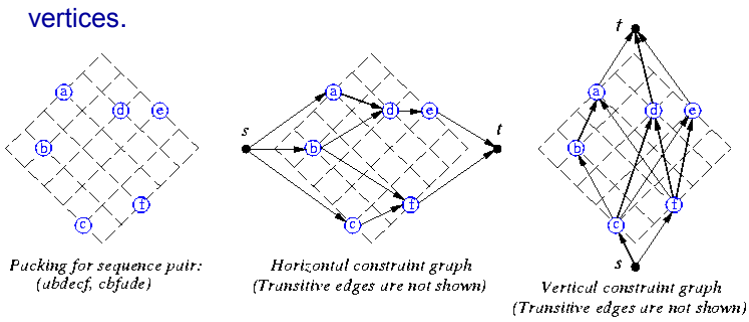
- No pair of positive (negative) loci cross each other, i.e., **loci are linearly ordered**.
- SP uses two sequences (Γ_+, Γ_-) to represent a floorplan.
  - **H-constraint:** (..a..b.., ..a..b..) iff a is on the left of b
  - **V-constraint:** (..a..b..,..b..a..) iff b is below a



(Γ_+, Γ_-) = (abdecf, cbfade)

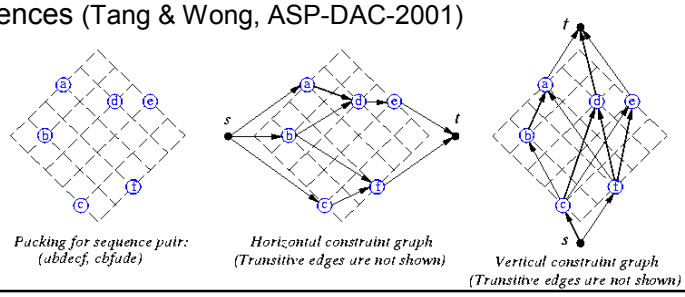
(Γ_+, Γ_-)-Packing

- 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.



Cost Evaluation

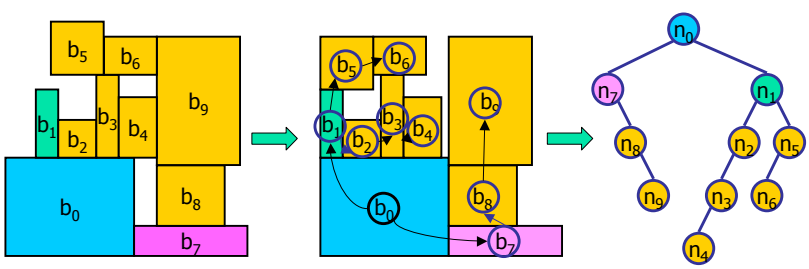
- **Optimal** (Γ_+, Γ_-)-Packing can be obtained in $O(n^2)$ time by applying a longest path algorithm on a vertex-weighted directed acyclic graph.
  - G_H and G_V are independent.
  - The X and Y coordinates of each module are the minimum values of the longest path length between s and the corresponding 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, ASP-DAC-2001)



Packing for sequence pair: (abdecf, cbfade) — *Horizontal constraint graph (Transitive edges are not shown)* — *Vertical constraint graph (Transitive edges are not shown)*

B*-Tree: Compacted Floorplan Representation

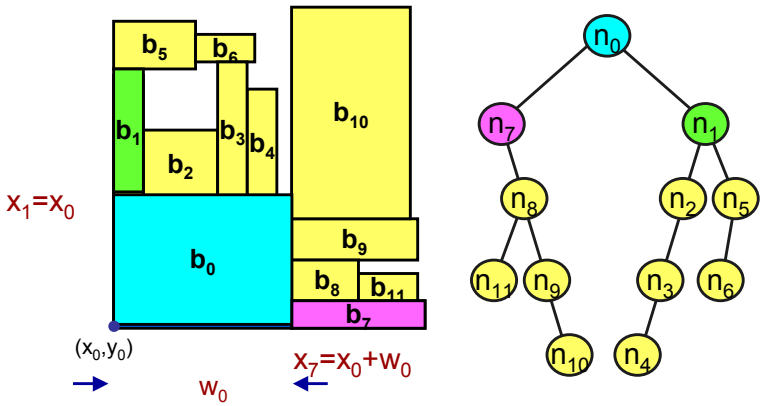
- Chang et. al., "B-tree: A new representation for non-slicing floorplans," DAC-2k.
  - Compact modules to left and bottom.
  - Construct an ordered binary tree (B*-tree).
    - Left child: the lowest, adjacent block on the right ($x_j = x_i + w_i$).
    - Right child: the first block above, with the same x-coordinate ($x_j = x_i$).



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

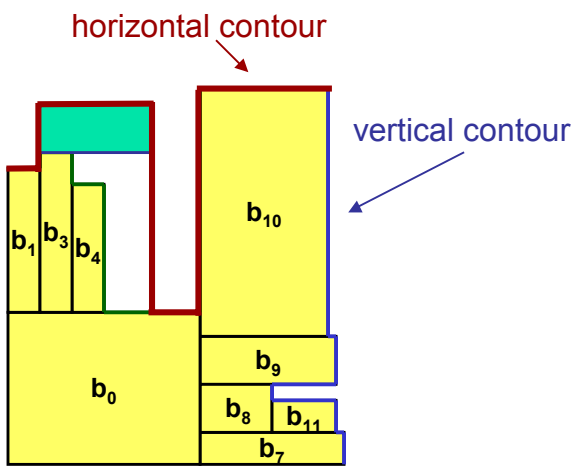
B*-tree Packing

- x-coordinates can be determined by the tree structure.
  - Left child: the lowest, adjacent block on the right ($x_j = x_i + w_i$).
  - Right child: the first block above, with the same *x*-coordinate ($x_j = x_i$).
- y-coordinates?



Computing y-coordinates

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

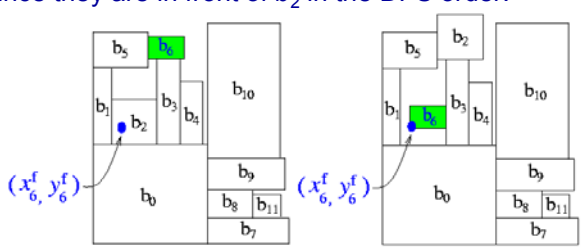


Pros and Cons

- Advantages
  - Binary tree based, efficient and easy.
  - Flexible to deal with hard, preplaced, soft, and rectilinear modules.
  - Transformation between a tree and its placement takes only linear time (v.s. $O(n^2)$ or $O(n\lg\lg n)$ for sequence pair).
  - Operate only on one B*-tree (v.s. two O-trees).
  - Can evaluate area cost incrementally.
  - Smaller solution space: only $O(n!\ 4^n/n^{1.5})$ combinations (v.s. $O((n!)^2)$ for sequence pair).
  - Directly corresponds to multilevel framework for large-scale floorplan designs.
- Disadvantages
  - Representation may change after packing.
  - Less flexible than sequence pair in representation
    - Can represent only compacted placement.
  - B*-tree is not P*-admissible

Coping with Pre-placed Modules

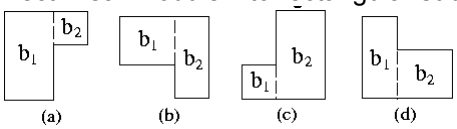
- If there are modules ahead or lower than b_i so that b_i cannot be placed at its fixed position (x^f_i, y^f_i), exchange b_i with the module in $D_i = \{b_j \mid (x_j, y_j) \le (x^f_i, y^f_i)\}$ that is closest to (x^f_i, y^f_i).
- Incremental area cost update is possible.
  - E.g., the positions of b_0, b_7, b_8, b_{11}, b_9, b_{10}, and b_1 (before b_2 in the DFS order of T) remain unchanged after the exchange since they are in front of b_2 in the DFS order.



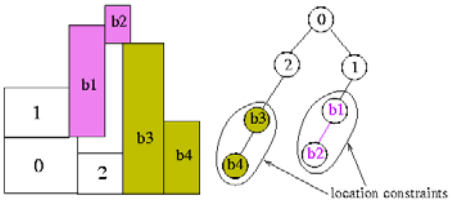
b_6 is a preplaced module

Coping with Rectilinear Modules

- Wu, Chang, Chang, "Rectilinear block placement using B*-trees," ACM TODAES, 2003 (ICCD-01).
- Partition a rectilinear module into rectangular sub-modules.

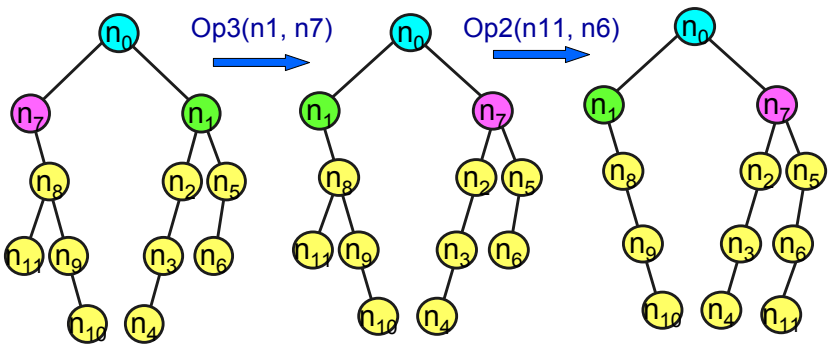


- Keep **location constraints** for the sub-modules.
  - E.g., Keep the right sub-module as the left child in the B*-tree.
- Align sub-modules, if necessary.
- Treat the sub-modules of a module as a whole during processing.



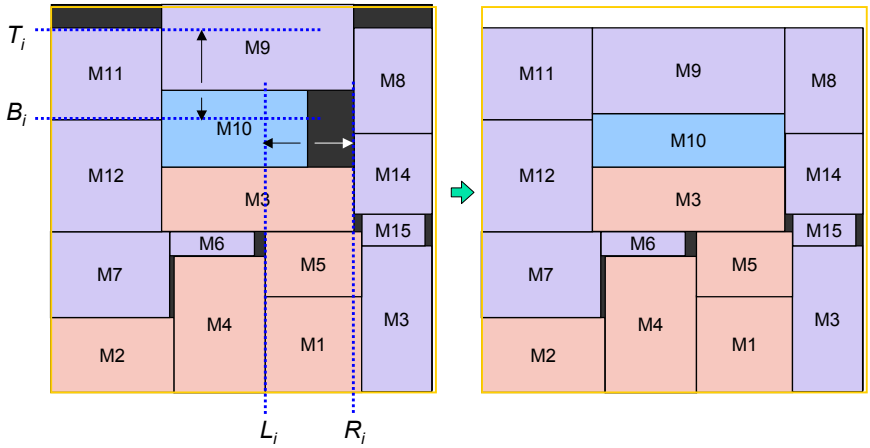
B*-Tree Perturbation

- Op1: rotate a macro
- Op2: delete & insert
- Op3: swap 2 nodes
- Op4: resize a soft macro



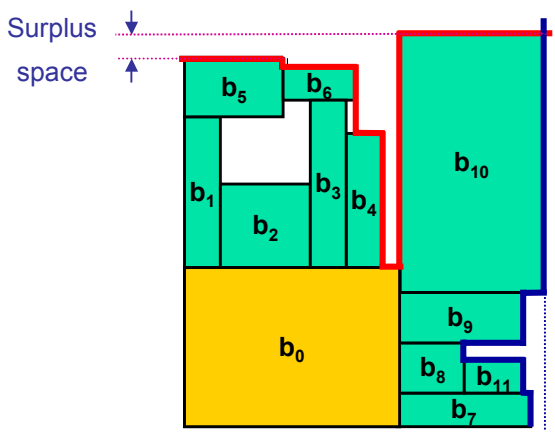
Simple Floorplan Sizing

- Key: Line up with adjacent modules
- Advantage: fast and reasonably effective



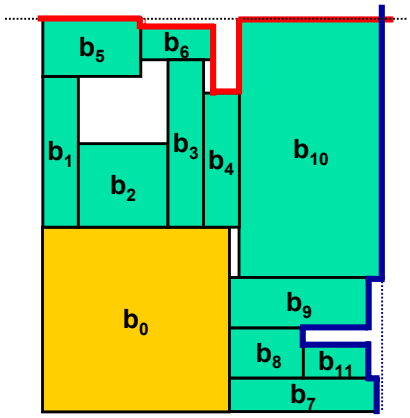
More Sophisticated Sizing (1/4)

- Step1: Change the shape of the inserted soft module.
- Step2: Change the shapes of other soft modules.



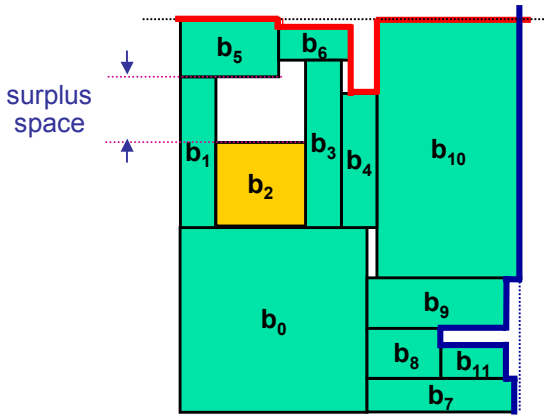
More Sophisticated Sizing (2/4)

- Step1: Change the shape of the inserted soft module
- Step2: Change the shape of other soft modules



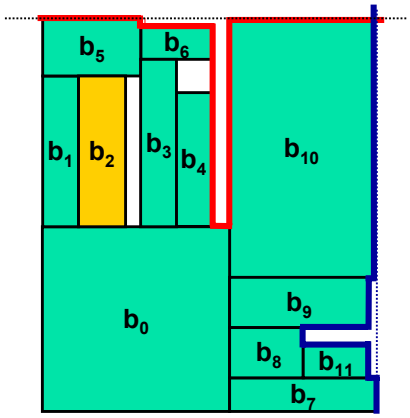
More Sophisticated Sizing (3/4)

- Step1: Change the shape of the inserted soft module
- Step2: Change the shapes of other soft modules

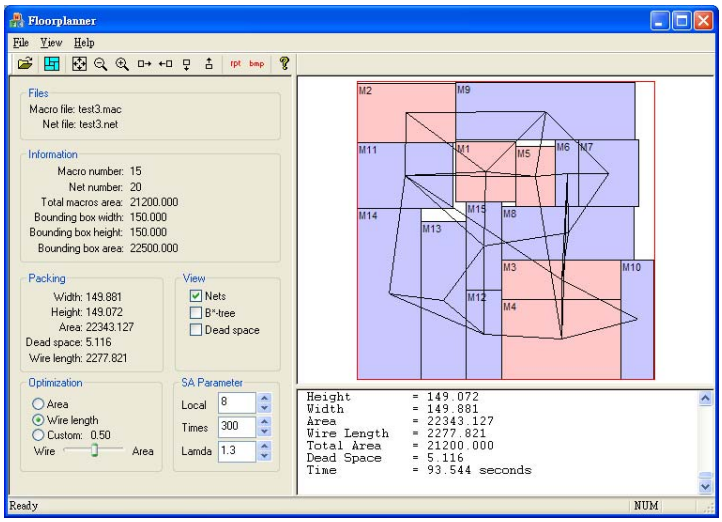


More Sophisticated Sizing (4/4)

- Step1: Change the shape of the inserted soft module
- Step2: Change the shape of other soft modules



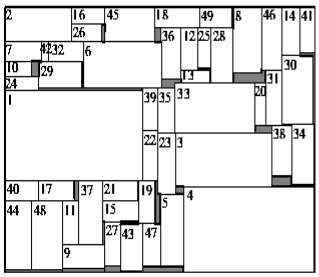
B*-tree Based Floorplanner



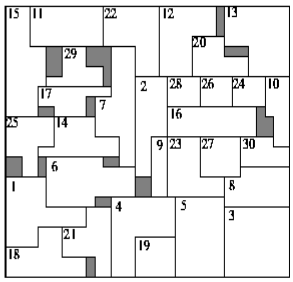
Courtesy T.-C. Chen

Example Results

- ami49: Area = 36.74 mm^2; dead space = 3.53%; CPU time = 0.25 min on SUN Ultra 60 (optimum = 35.445 mm^2).
- Multilevel B*-tree: Area = 36.46 mm^2, dead space = 2.78%; CPU time = 0.4 min (best area so far).



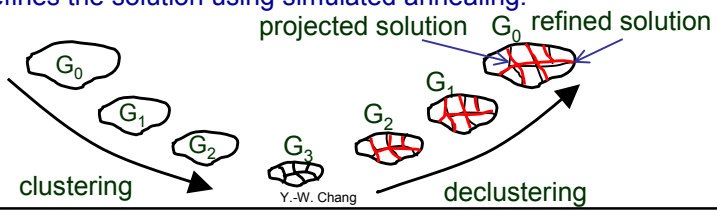
ami49



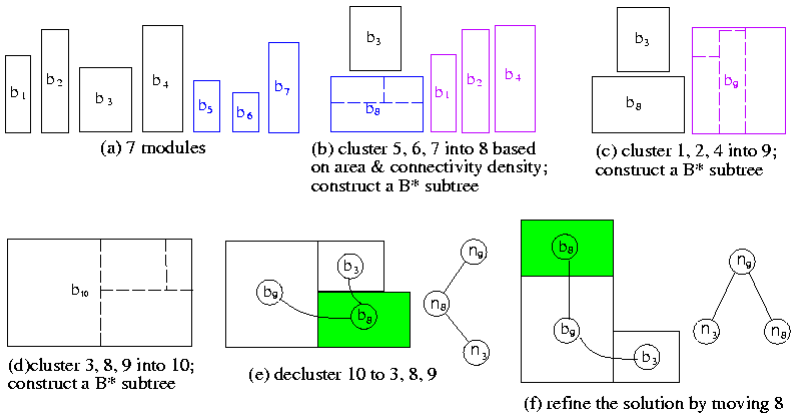
Rectangular, L–, and T–shaped modules

Multilevel B*-trees

- Lee, Hsu, Chang, Yang, "Multilevel floorplanning/placement for large-scale modules using B*-trees," DAC-2003.
- 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.



Multilevel B*-tree Example



(a) 7 modules

(b) cluster 5, 6, 7 into 8 based on area & connectivity density; construct a B* subtree

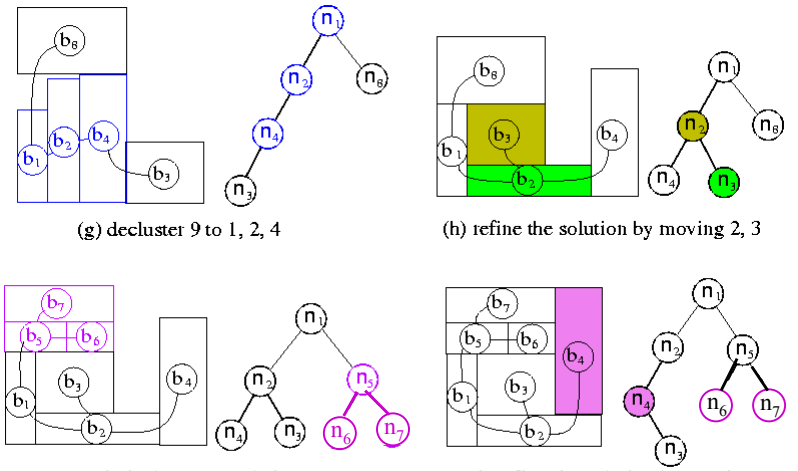
(c) cluster 1, 2, 4 into 9; construct a B* subtree

(d)cluster 3, 8, 9 into 10; construct a B* subtree

(e) decluster 10 to 3, 8, 9

(f) refine the solution by moving 8

Multilevel B*-tree Example (cont'd)



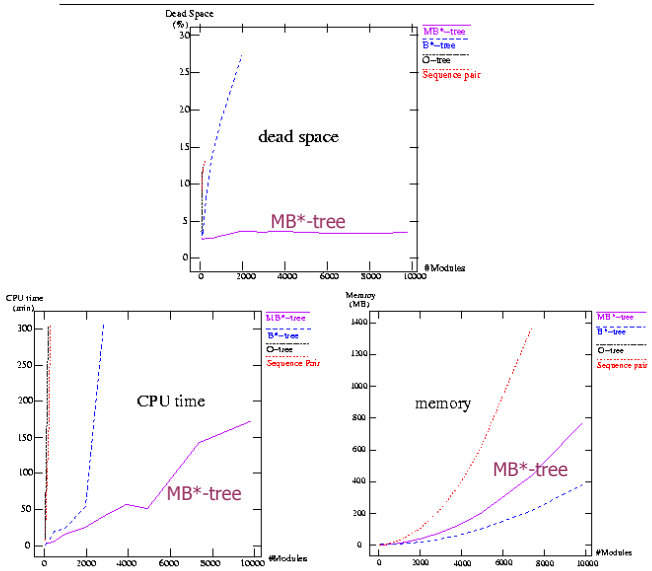
(g) decluster 9 to 1, 2, 4

(h) refine the solution by moving 2, 3

(i) decluster 8 to 5, 6, 7

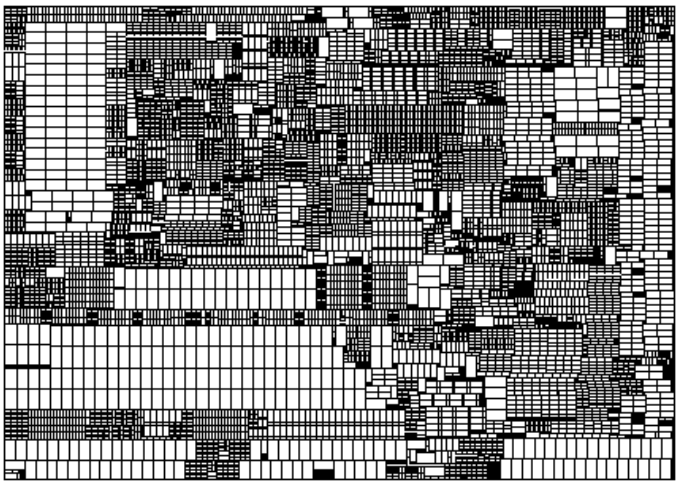
(j) refine the solution by moving 4

Comparison among Representations



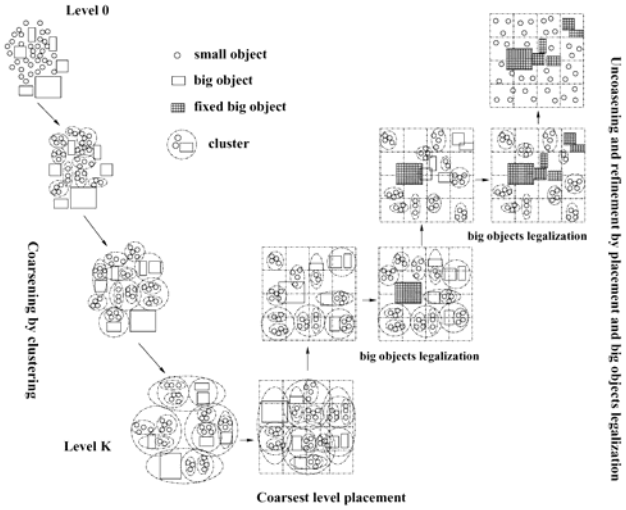
Layout of ami49_200

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



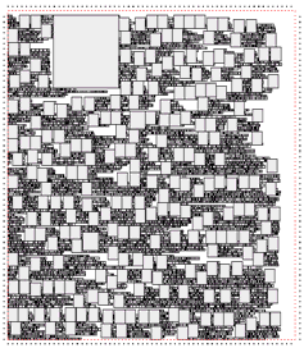
Multi-level, Multi-stage Framework

- How to handle mixed-size cell/macro placement?
  - Key: Cluster cells/macros when their sizes are comparable.

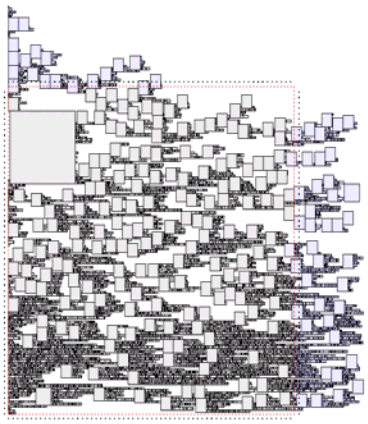


Results on Multi-level, Multi-stage Placement

- Benchmark: ISPD98 ibm01 with 12,752 cells, 247 macros
  - A_{max}/A_{min} = 8416



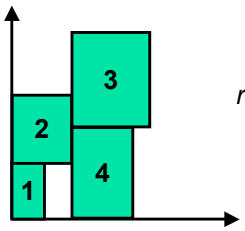
Multilevel, multistage flow: 0.78 min

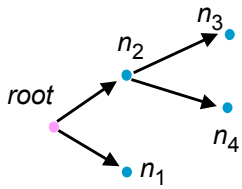


Multilevel flow: 60 min

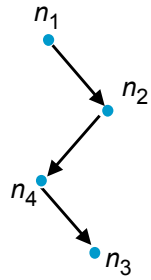
Problems with Tree-based Representations

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

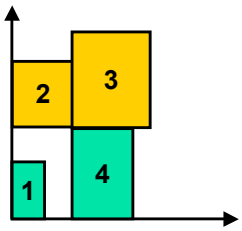




horizontal O-tree



B*-tree

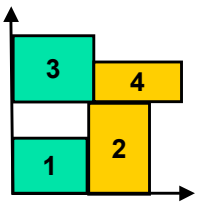


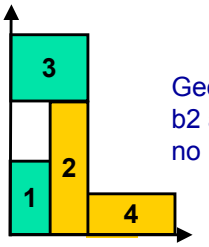
O-tree??

B*-tree??

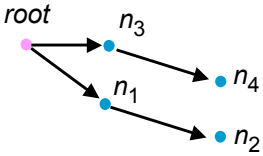
Problems with Tree-based Representations (cont'd)

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

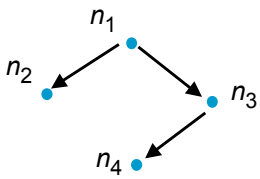




Geometric relation between b2 and b4 is changed, b2 is no longer a boundary module



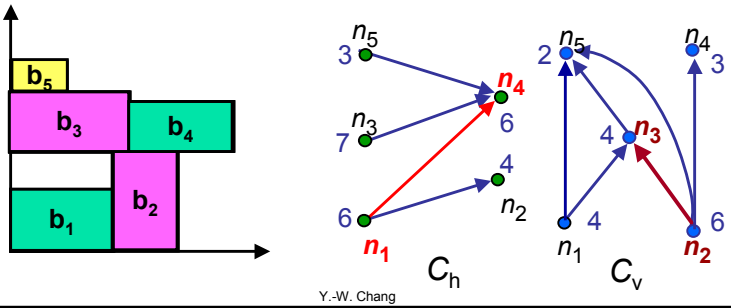
Horizontal O-tree



B*-tree

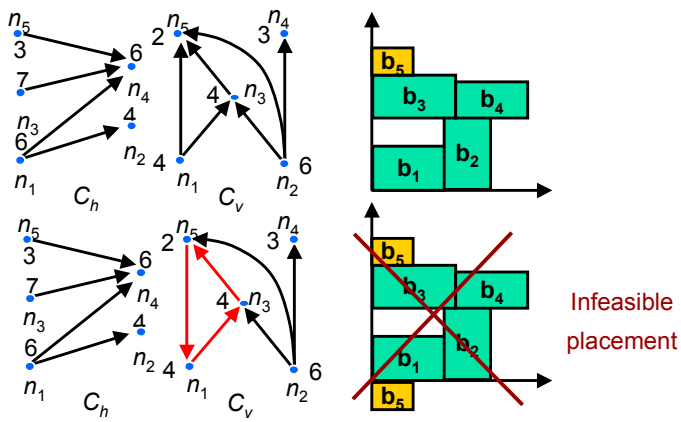
Transitive Closure Graph (TCG)

- Lin & Chang, "TCG: A transitive closure graph representation for non-slicing floorplans," DAC-2001 (TVLSI-2004)
- TCG = (C_h, C_v): pair of vertical and horizontal constraint graphs.
  - 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).



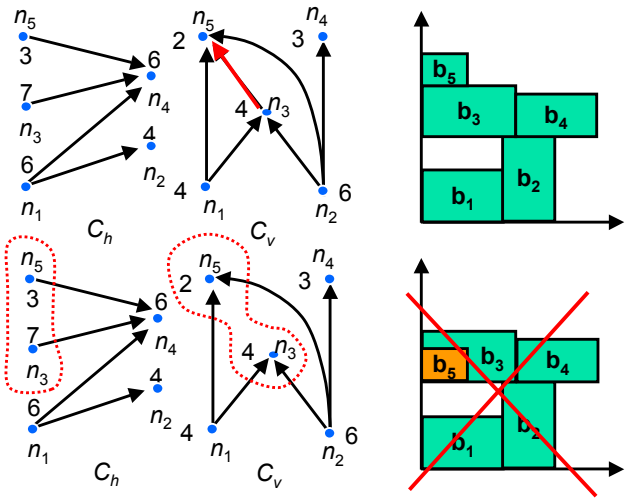
Feasibility of TCG

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



Feasibility of TCG

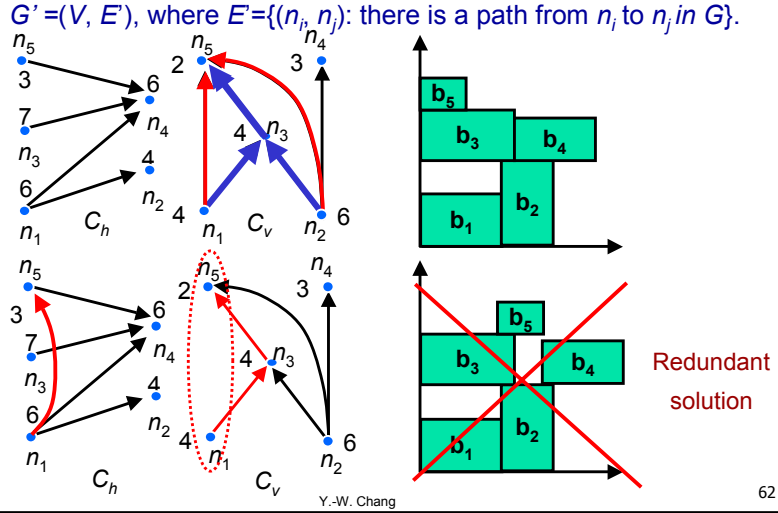
2. Each pair of nodes must be connected by exactly one edge either in C_h or in C_v to prevent an overlap among modules.



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.

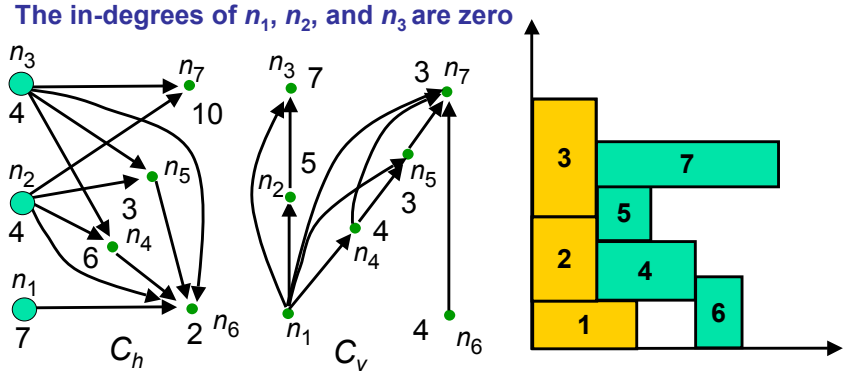
– The transitive closure of a directed graph G=(V, E) is the graph G' =(V, E'), where E'={(n_i, n_j): there is a path from n_i to n_j *in G*}.



Boundary Modules in TCG

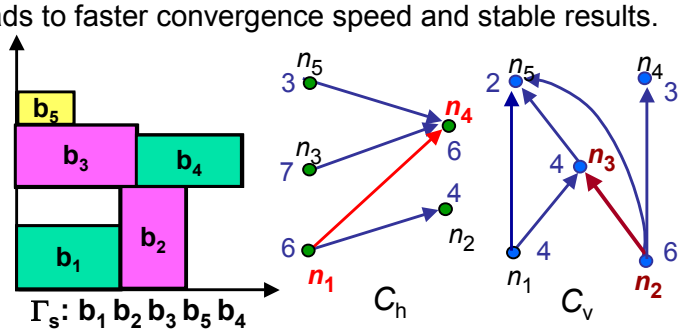
- b_i is a left (right) boundary module if n_i's in-degree (out-degree) in C_h is zero.
- b_i is a bottom (top) boundary module if n_i's in-degree (out-degree) in C_v is zero.

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



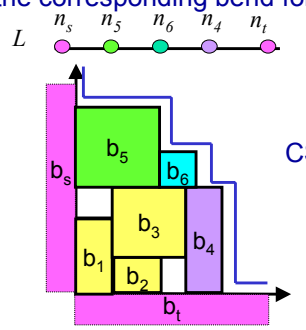
TCG-S

- 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 stable results.



Corner Sequence (CS)

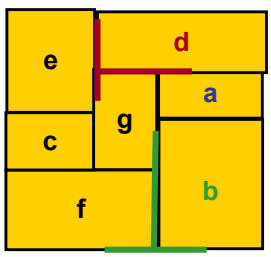
- Lin, Chang, Lin, "Corner sequence: A P-admissible floorplan representation with linear-time packing scheme," IEEE TVLSI 2003.
- Sequence of modules and their corresponding corners CS = <(S_1, D_1), (S_2, D_2), …, (S_m, D_m) >
  - S_i: a module
  - D_i: the corresponding bend for packing S_i



CS = <(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]) >

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.



S = (fcegbad), L = (001100), T = (001010010)

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	$O(n!2^{2n}/n^{1.5})$	$O(n)$	⊗	3
B*-tree	$O(n!2^{2n}/n^{1.5})$	$O(n)$	⊗	3
CS	$O(n!)^2$	$O(n)$	⊗	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*-admoissible); 3 (compacted; P-admissible); 2 (mosaic); 1 (slicing)

Floorplan Quality Comparison

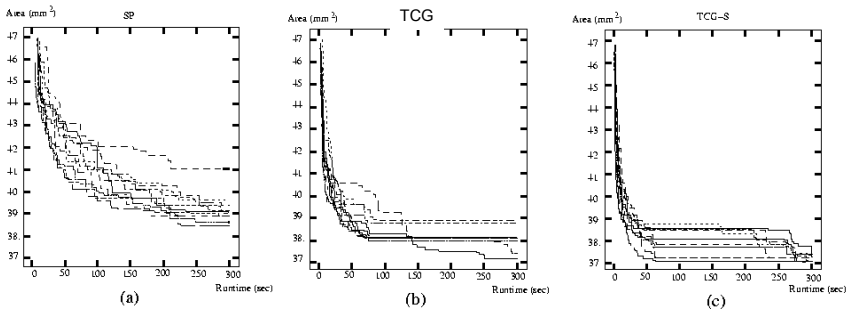
- Floorplan areas obtained by popular representations
  - Minimum areas are in red

MCNC Circuits	SP ICCAD 1995	Q-Seq DATE 2002	O-tree DAC 1999	CBL ICCAD 2001	TCG DAC 2001 (Ours)	TCG-S DAC 2002 (Ours)	CS TVLSI 2003 (Ours)	B*-tree DAC 2000 (Ours)
apte	48.12	**46.92**	47.1	NA	**46.92**	**46.92**	**46.92**	**46.92**
xerox	20.69	19.93	20.1	20.96	19.83	**19.796**	19.83	**19.796**
hp	9.93	9.03	9.21	NA	**8.947**	**8.947**	**8.947**	**8.947**
ami33	1.22	1.194	1.25	1.20	1.20	1.185	1.18	**1.168**
ami49	38.84	36.75	37.6	38.58	36.77	36.4	**36.28**	36.4

- Best run-time performance: B*-tree
  - B*-tree-v1.0, TCG, and TCG-S: available at http://cc.ee.ntu.edu.tw/~ywchang/research.html

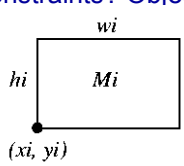
Convergence Speed & Stability

- **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.



Floorplanning by Mathematical Programming

- Sutanthavibul, Shragowitz, and Rosen, "An analytical approach to floorplan design and optimization," 27th DAC, 1990.
- 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.
  - $a_i \le w_i/h_i \le b_i$: 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?



*Area = hi * wi*
Aspect ratio = wi / hi

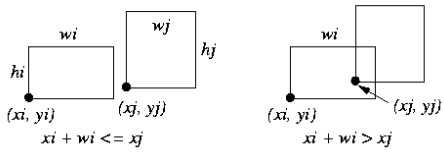
Nonoverlap Constraints

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

		p_{ij}	q_{ij}
M_i to the left of M_j:	$x_i + w_i \le x_j$	0	0
M_i below M_j:	$y_i + h_i \le y_j$	0	1
M_i to the right of M_j:	$x_i - w_j \ge x_j$	1	0
M_i above M_j:	$y_i - h_j \ge y_j$	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., $p_{ij} = 0$, $q_{ij} = 1 \Rightarrow y_i + h_i \le y_j$ is satisfied

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



Cost Function & Constraints

- Minimize *Area* = *xy*, **nonlinear!** (*x, y*: width and height of the resulting floorplan)
- How to fix?
  - Fix the width *W* and minimize the height *y*!
- Four types of constraints:
  1. no two modules overlap ($\forall\ i, j$: $1 \le i < j \le n$);
  2. each module is enclosed within a rectangle of width *W* and height *H* ($x_i + w_i \le W$, $y_i + h_i \le H$, $1 \le i \le n$);
  3. $x_i \ge 0$, $y_i \ge 0$, $1 \le i \le n$;
  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.

$$\min \quad y$$

subject to

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

- 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, lp_solve, etc.

Mixed ILP for Floorplanning (cont'd)

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

$$\min \quad y$$

subject to

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

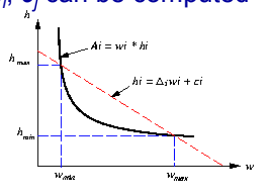
- For each module *i* with free orientation, associate a 0-1 variable r_i:
  - $r_i = 0$: 0° rotation for module *i*.
  - $r_i = 1$: 90° rotation for module *i*.
- $M = \max\{W, H\}$.

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}).

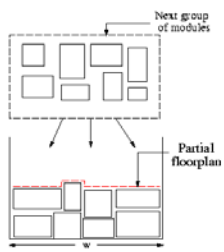
$$h_i = \Delta_i w_i + c_i \qquad /*\ y = mx + c\ */$$
$$\Delta_i = \frac{h_{max} - h_{min}}{w_{min} - w_{max}} \qquad /*\ slope\ */$$
$$c_i = h_{max} - \Delta_i w_{min} \qquad /*\ c = y_0 - mx_0\ */$$

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



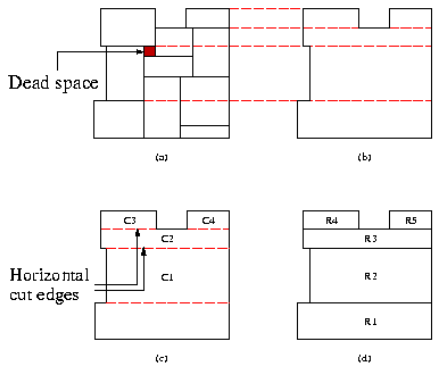
Reducing the Size of the Mixed ILP

- 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.



Summary: Floorplanning

- 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 Polished expression (DAC-86)
  - 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), MB*-tree (DAC-03), CS (TVLSI, 2003)
  - General: SP (ICCAD-95), BSG (ICCAD-96), TCG (DAC-01), TCG-S (DAC-02).
- P*-admissible representations: all 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

Summary: Floorplanning (cont'd)

- Since each representation has its pros and cons, so maybe we can
  - Integrate two or more representations to get a better one (e.g., TCG-S, DAC-02)
  - Apply different representations at different stages
- Other issues
  - Soft module: shape curve (NPE, DAC-86), (Integer) linear programming (DAC-90, DAC-2k), stretching range (B*-tree, DAC-2k), Lagrangian relaxation (SP, ISPD-2k)
  - 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)
  - Rectilinear module: TCAD-2K (SP), ICCAD-98 (SP), ISPD-98 (SP), ISPD-01 (SP), DATE-02 (TCG), TVLSI-02 (TCG), ICCD-2K (B*-tree), ACM TODAES-03 (B*-tree), ISPD-01 (O-tree)
  - Range constraint: ISPD-99 (NPE), ASPDAC-01 (SP), DAC-02 (TCG-S)
  - Boundary constraint: ASPDAC-01 (SP), DAC-02 (TCG-S), IEE Proc.-02 (B*-tree)
  - Bus-driven constraint: ICCAD-2003

Summary: Floorplanning (cont'd)

- Large-scale module floorplanning/placement (MB*-tree, DAC-03)
- Mixed sized cell/block floorplanning/placement (ISPD-02, ASPDAC-03, ICCAD-03)
- Performance-driven floorplanning
  - Buffer planning (ICCAD-99, ISPD-2K, DAC-01, ASPDAC-03)
  - Wire planning (ICCAD-99)
  - Noise-aware floorplanning (ASPDAC-03)
  - Power supply planning (ASPDAC-01, 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://cc.ee.ntu.edu.tw/~ywchang/research.html.