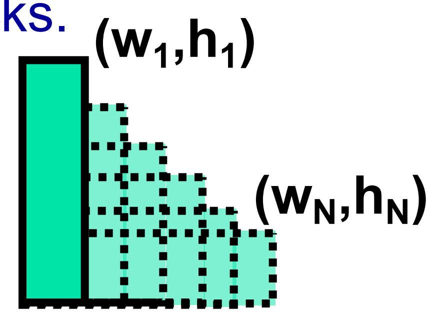
Chapter 3-2

VLSI Physical Design

何宗易
Tsung-Yi Ho
tyho@cs.nthu.edu.tw
http://theta.cs.nthu.edu.tw
Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan

Hierarchical Design

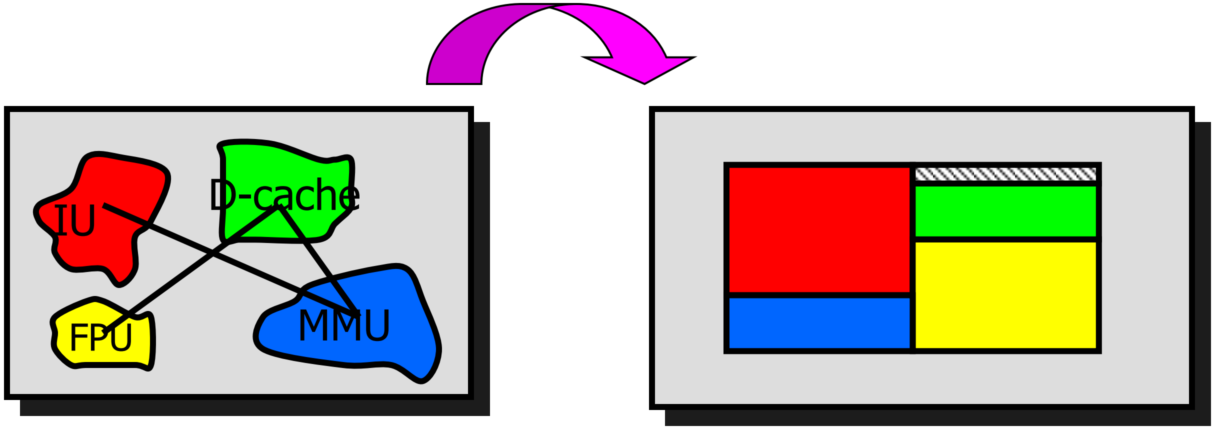
- Partitioning leads to
  - Blocks with well-defined **areas and shapes** (rigid/hard blocks).
  - Blocks with approximated areas and no particular shapes (flexible/soft blocks).
  - A **netlist** specifying connections between the blocks.
- Hierarchical design needs to
  - Put the blocks together.
  - Design each block.



- How to put the blocks together without knowing their shapes and the positions of the I/O pins?
- If we design the blocks first, those blocks may not be able to form a tight packing.

Floorplanning

- Problem
  - Given circuit modules and their connections, determine the *approximate* location and shape of circuit elements
- Approximate idea of
  - module areas
  - module connectivity
- Provides
  - area budgets
  - timing info (affect RTL?)

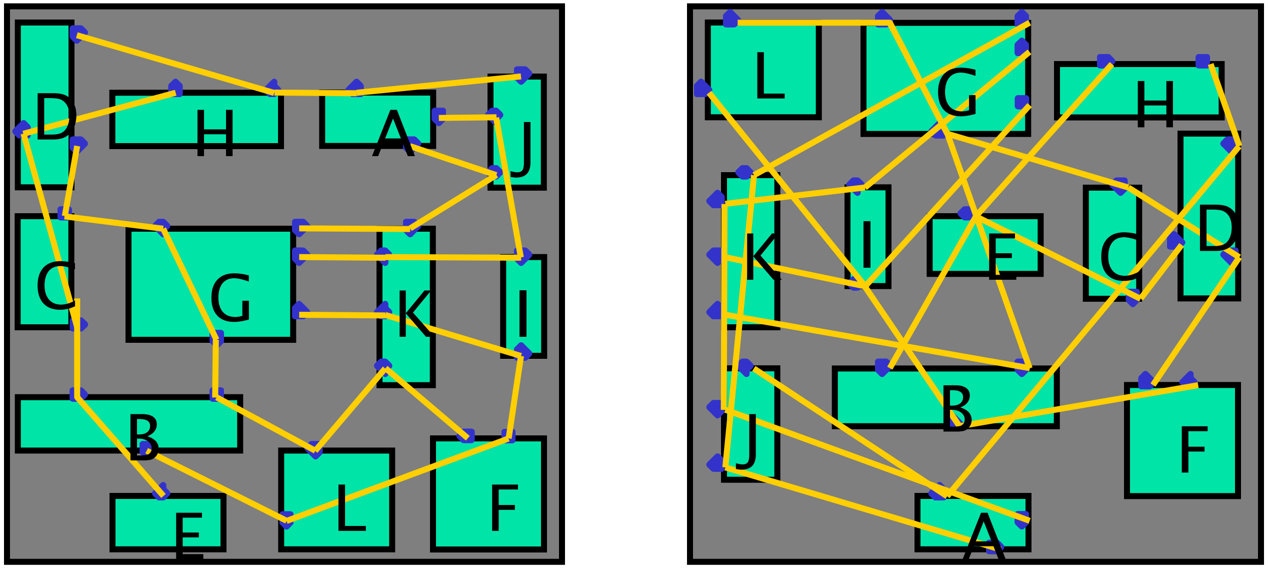


Floorplanning (cont.)

- Objectives:
  - Minimize area
  - Minimize total wire length
    - to make subsequent routing phase easy
      (short wire length roughly translates into routability)
  - Additional cost components:
    - Wire congestion (exact routability measure)
    - Wire delays
    - Power consumption
- Possible additional constraints:
  - Fixed location for some modules (pre-place and range constraints)
  - Fixed die, or range of die aspect ratio (fixed-outline)
  - Interconnect optimization (bus-driven)
  - Multiple dimensions (2.5D or 3D)
  - Analog designs (thermal or symmetric)

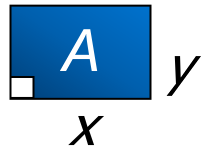
Floorplanning: Why Important?

- Early stage of physical design
  - Determines the location of large blocks
    ➔ detailed placement easier (divide and conquer!)
  - Estimates of area, delay, power
    ➔ important design decisions
  - Impact on subsequent design steps (e.g., routing, heat dissipation analysis and optimization)



Input of Floorplan Design

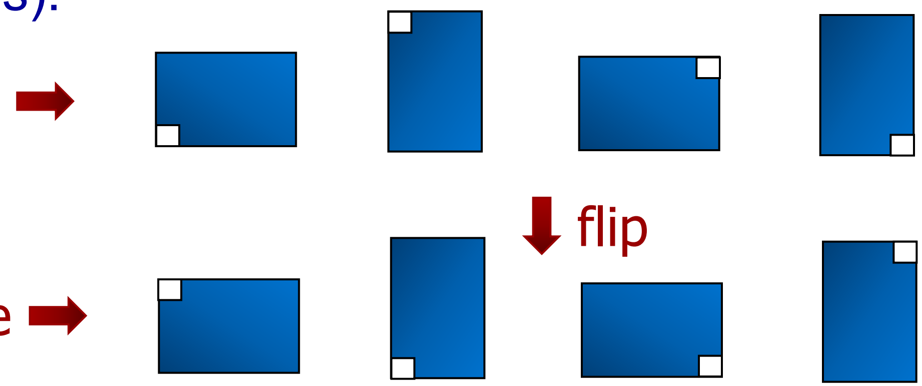
- Modules:
  - Area: $A = xy$



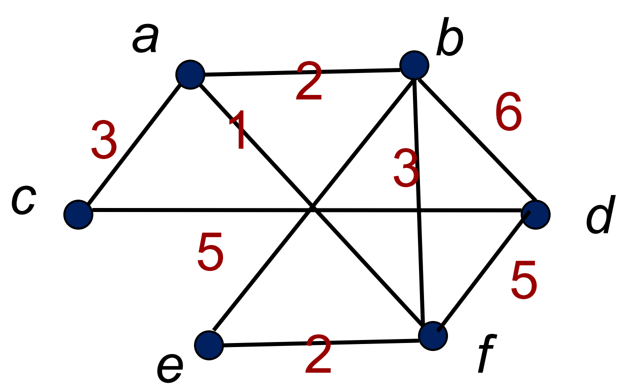
  - *Aspect ratio* (for soft block only) : $r \leq y/x \leq s$
  - *Rotation* (8 directions):

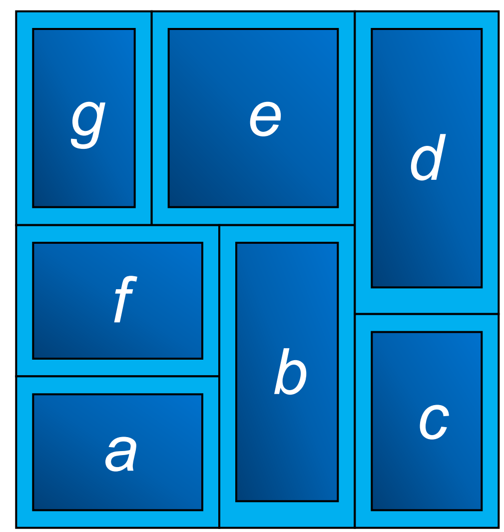
rotate clockwise

rotate counterclockwise



- *Module connectivity*





Bounds on Aspect Ratios

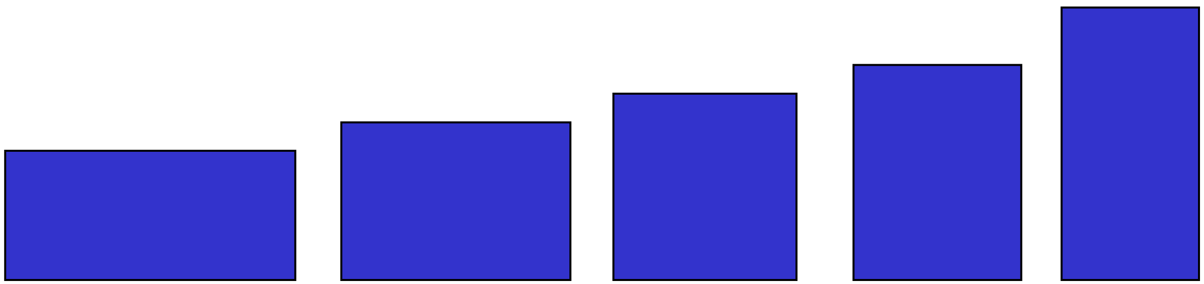
If there is no bound on the aspect ratios, can we pack everything tightly?

- Sure!

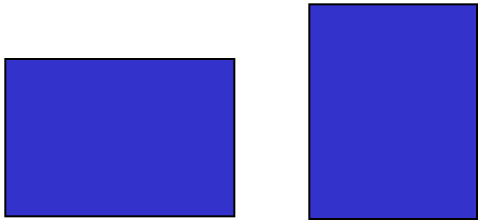
But we don't want to layout blocks as long strips, so we require $r_i \leq h_i/w_i \leq s_i$ for each i.

Bounds on Aspect Ratios

- We can also allow several shapes for each block:

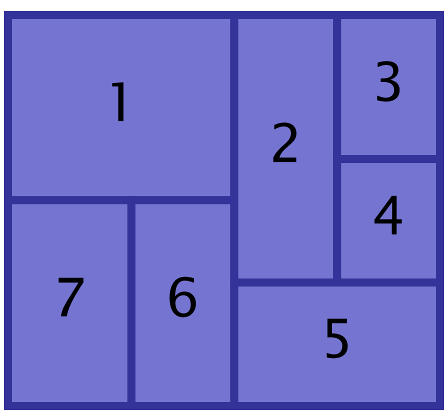


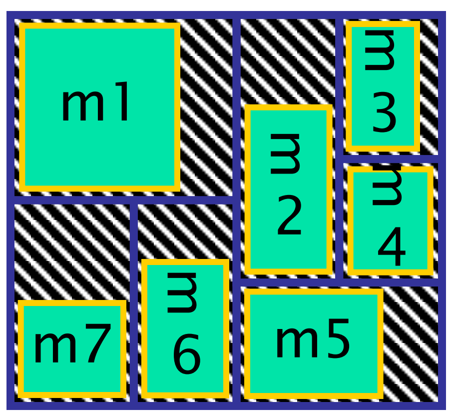
- For hard blocks, the orientations can be changed:



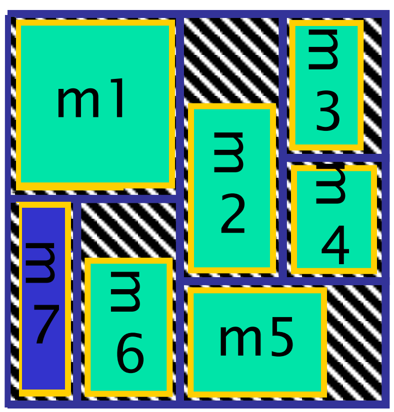
Area Utilization, Hard and Soft Modules

- The hierarchy tree and floorplan define "place holders" for modules
- Area utilization
  - Depends on how nicely the rigid modules' shapes are matched
  - Soft modules can take different shapes to "fill in" empty slots ➔ floorplan sizing





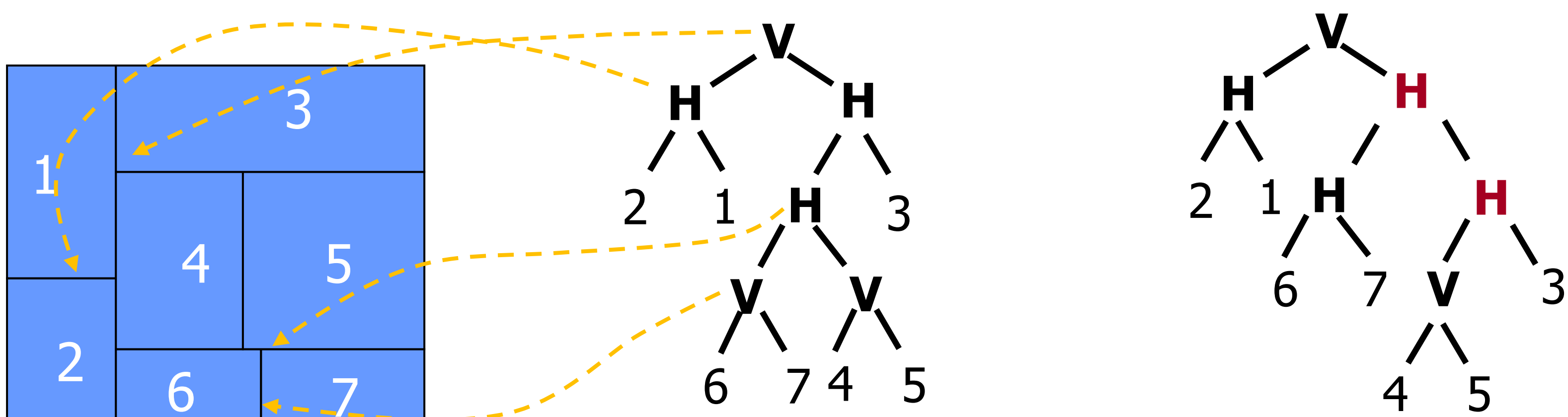
Area = 20x22 = 440



Area = 20x19 = 380

Slicing Tree for Slicing Floorpaln

- **Slicing tree:** A binary tree with *n* leaves and *n*-1 nodes, 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.



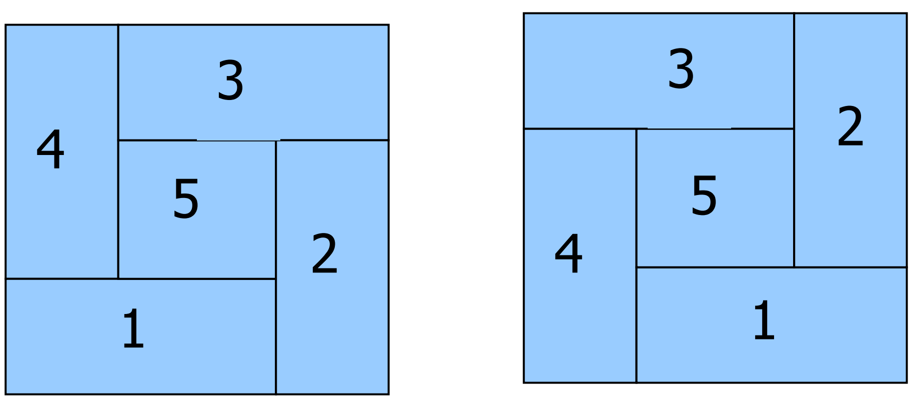
A slicing floorplan

A slicing tree (**skewed**)

A slicing tree (**non-skewed**)

Non-Slicing Floorplan

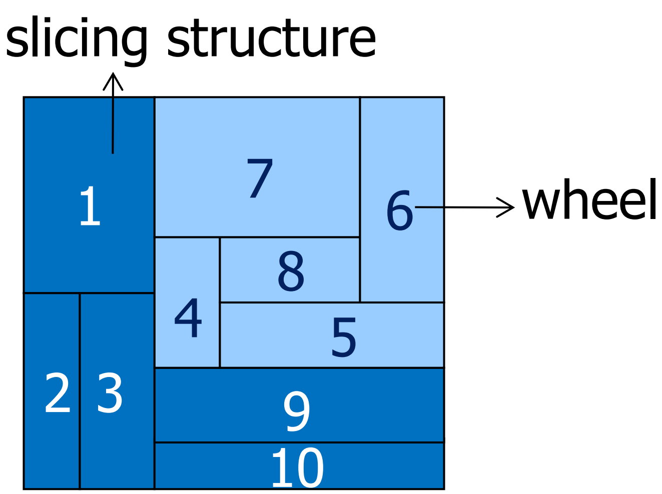
- **Non-Slicing Floorplan:** a floorplan that is not slicing one
- **Wheel:** the smallest non-slicing floorplans (Wang and Wong, TCAD, Aug. 92).
  - There are only two possible wheels as shown in the following.



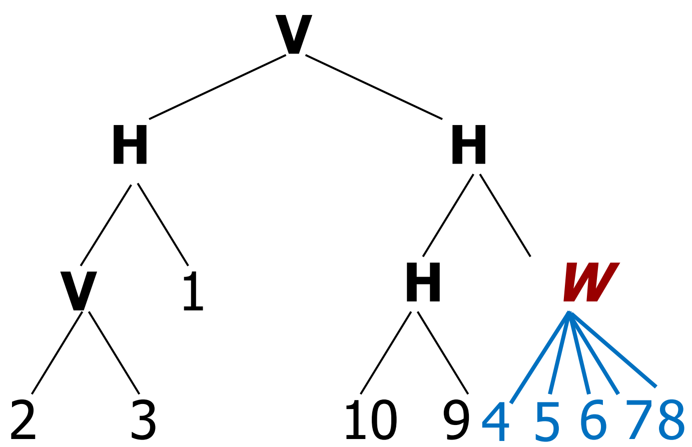
Two possible wheels

Hierarchical Floorplan of Order Five

- **Hierarchical Floorplan of Order Five:** a floorplan that can be obtained by by recursively subdividing each rectangle into either *two* parts (*slicing structure*), or either *five* parts (*wheel*)
- **Floorplan tree:** A tree representing the hierarchy of partitioning. Each leaf represents a basic rectangle and each node a composite rectangle.



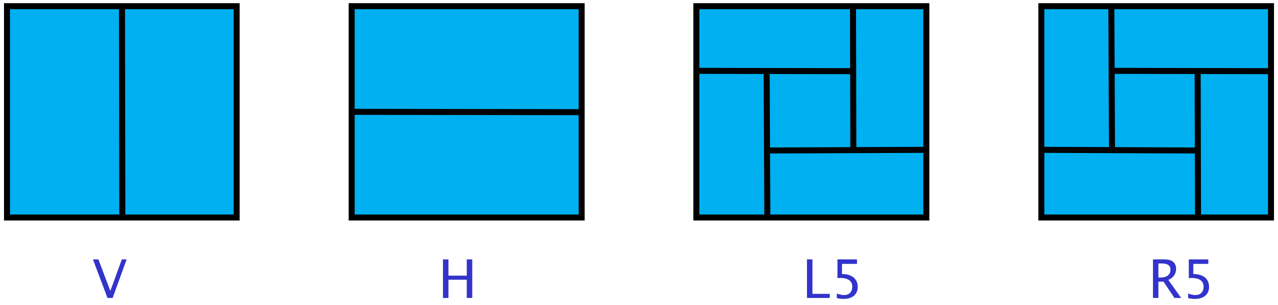
(a) A hierarchical floorplan of order 5



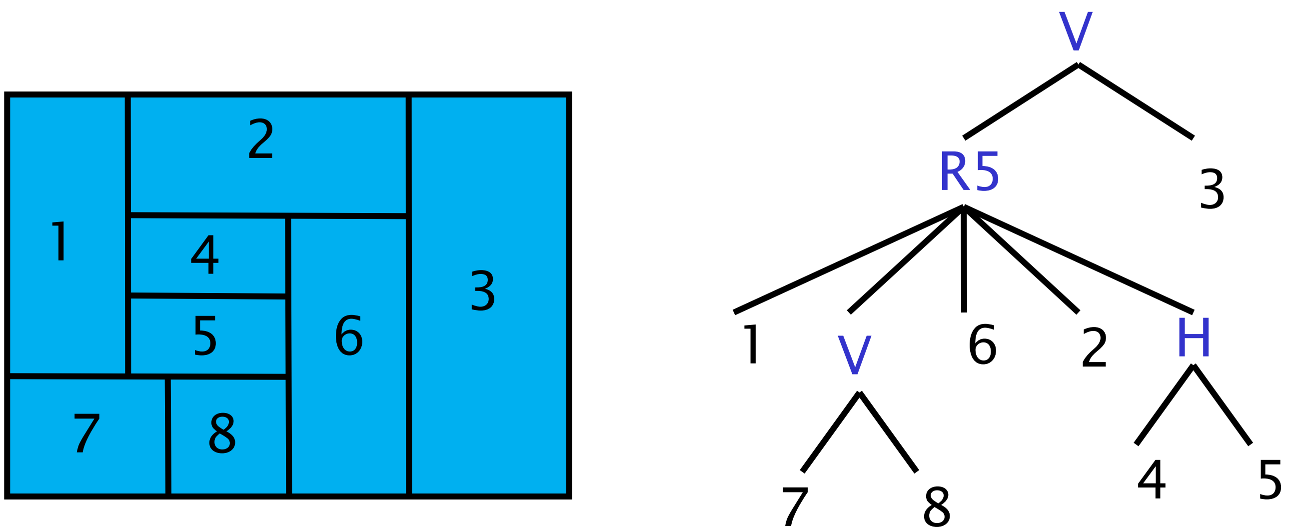
(b) Corresponding floorplan tree.

Non-slicing Floorplan Example

- Hierarchical floorplan of order 5
  - Templates



  - Floorplan and tree example



Floorplanning Algorithms

- Components
  - "Placeholder" representation
    - Usually in the form of a tree
    - Slicing class: Polish expression [Otten]
    - Non-slicing class: B* tree, Sequence Pair, BSG, etc.
    - Just defines the *relative position* of modules
  - Perturbation
    - Going from one floorplan to another
    - Usually done using Simulated Annealing
  - Floorplan sizing
    - Definition: Given a floorplan tree, choose the best shape for each module to minimize area
    - Slicing: polynomial, bottom-up algorithm
    - Non-slicing: NP! Use mathematical programming (exact solution)
  - Cost function
    - Area, wire-length, ...

Solution Representation

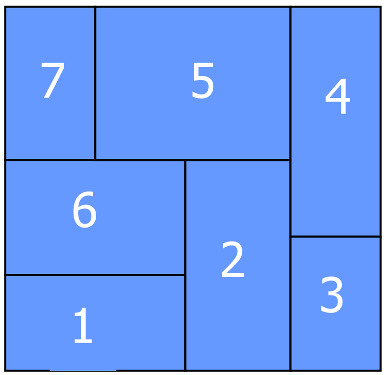
- R.H.J.M. Otten, "Automatic floorplan design," DAC-82.
- Wong & Liu, "A new algorithm for floorplan design," DAC-86.

- An expression $E = e_1 \, e_2 \ldots \, e_{2n-1}$, where $e_i \in \{1, 2, \ldots, n, \mathbf{H}, \mathbf{V}\}$, $1 \le i \le 2n-1$, 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**

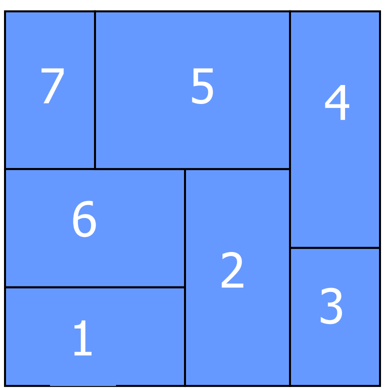
# of operands = 4 (includes 1 6 … 3 5)
# of operators = 2 (includes … **H** ….**V**)



Solution Representation (cont'd)

- Polish expression ($E = e_1\ e_2\ldots\ e_{2n\text{-}1}$) is equivalent to the postorder traversal of a slicing tree since
    1. ij**H**: rectangle i on bottom of j.
    2. ij**V**: rectangle i on left of j.

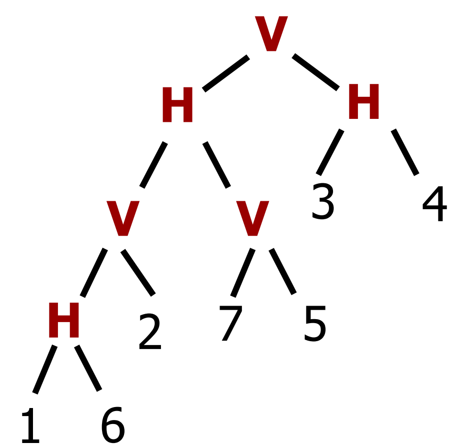
Poster traversal of a tree!



$E =$ 16**H**2**V**75**VH**34**HV**

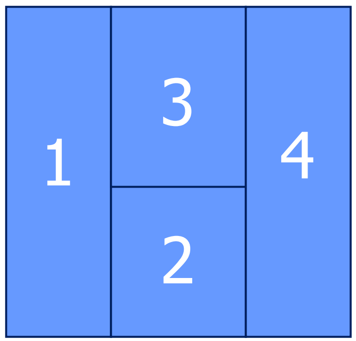
1 is bellow 6

7 is on left of 5

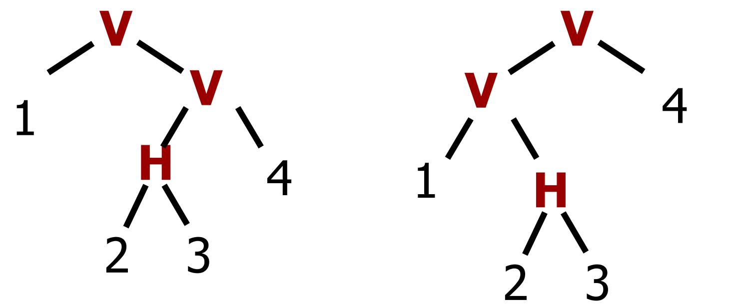


Redundant Representation

- One floorplan can be represented by more than one representations.



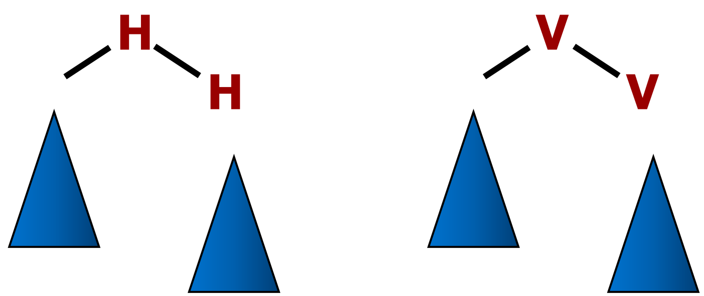
(a) A floorplan.



E = 123**H**4**VV** non-skewed!

E = 123**HV**4**V** skewed!

(b) The corresponding representations.



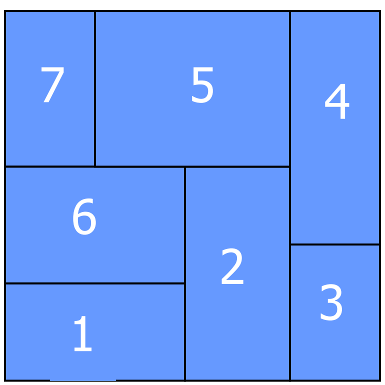
.....HH....   .....VV....

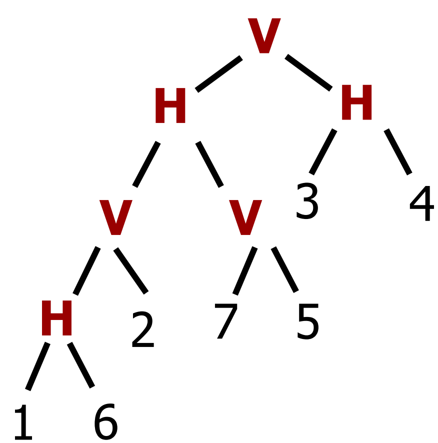
(c) Non-skewed slicing trees and the corresponding polish expressions.

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

Normalized Polish Expression

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

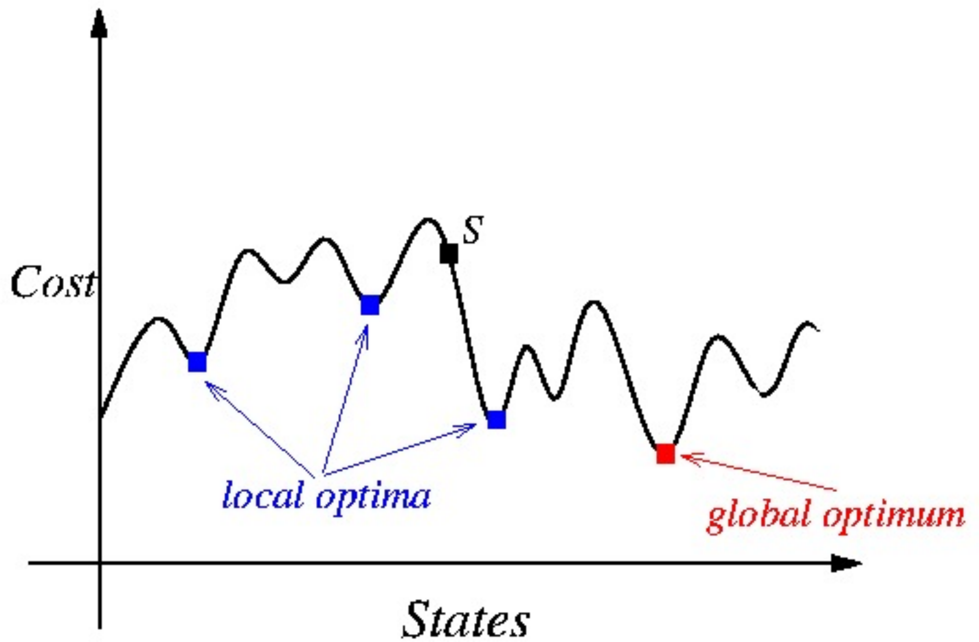




E = 16**H**2**V**75**VH**34**HV**

A normalized Polish expression

Simulated Annealing Revisit



- Basic Ingredients for Simulated Annealing:
  - **Solution state**
  - **Neighborhood structure**
  - **Cost function**
  - **Annealing schedule**

Neighborhood Structure

- **Chain: HVHVH** … or **VHVHV** …

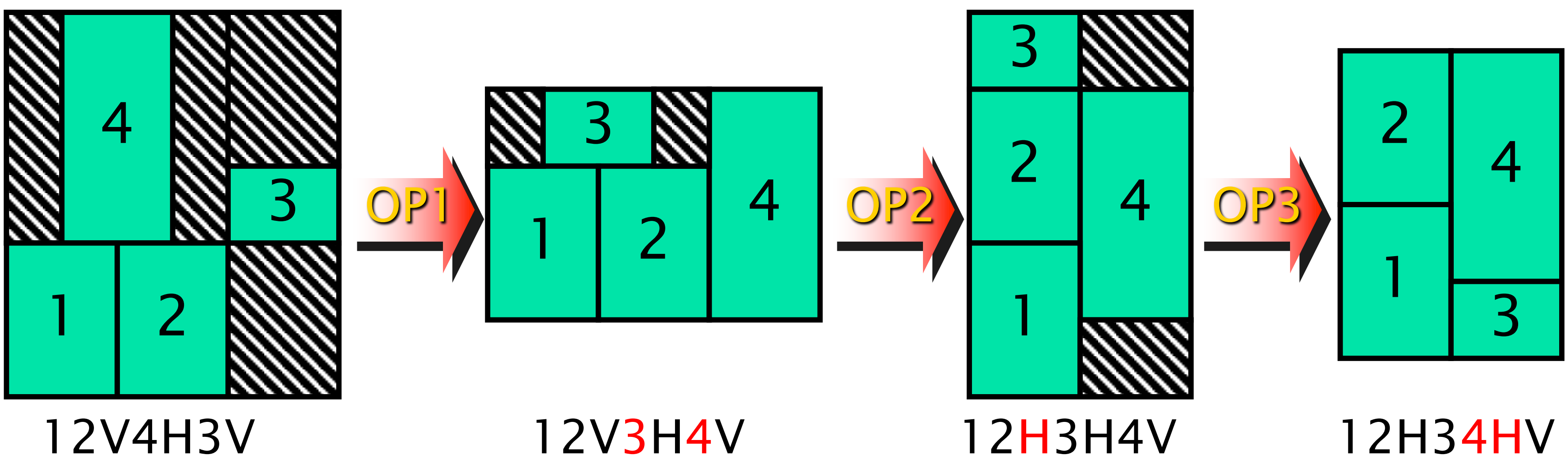
1 6 **H** 3 5 **V** 2 **H** **V** 7 4 **H** **V**

chain

- **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:
  - *OP1* (**Operand Swap**): Swap two adjacent operands.
  - *OP2* (**Chain Invert**): Complement some chain (**V** = **H**, **H** = ***V***).
  - *OP3* (**Operator/Operand Swap**): Swap two adjacent operand and operator.

Example

- *OP1* (**Operand Swap**): Swap two adjacent operands.
- *OP2* (**Chain Invert**): Complement some chain (**V** = **H**, **H** = ***V***).
- *OP3* (**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 *(the sequence of operands and operators maintains unchanged!!!)* .
  - **Check the *M3* moves!** Reject "illegal" *M3* moves.

12H**3H**4V $\xrightarrow{M3}$ 12H**H3**4V

# of operand = 2
# of operator = 2
violate the balloting property

- **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 $E = e_1\ e_2 \ldots e_k$, $1 \leq k \leq 2n$-1

In the above example, *M3* swaps e_4 and e_5 (i = 4, N_5= 2).

Because $2N_5 <\ 4$ is violated, we cannot apply M3 on e_4 and e_5.

Cost Function

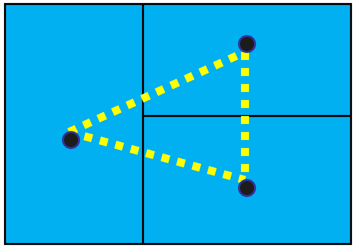
A commonly used objective function is a weighted sum of area and wirelength:

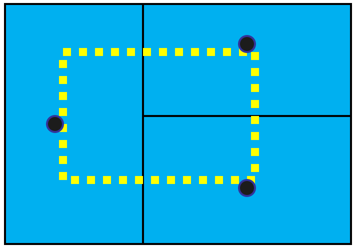
$$\text{cost} = \alpha A + \beta W$$

where A is the total area of the packing, W is the total wirelength, and α and β are constants.

Wirelength Estimation

- Exact wirelength of each net is not known until routing is done.
- In floorplanning, even pin positions are not known yet.
- Some possible wirelength estimations:
  - Center-to-center estimation
  - Half-perimeter estimation

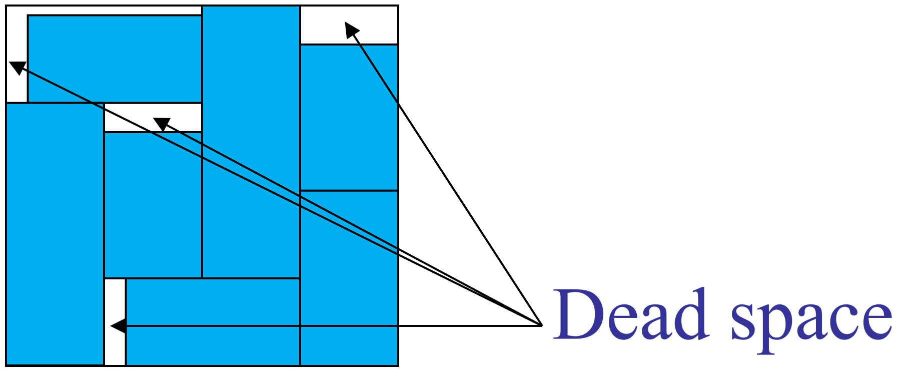




- $\boldsymbol{W} = \sum_{ij} c_{ij} d_{ij}$.
  - c_{ij}: # of connections between blocks i and j.
  - d_{ij}: center-to-center distance between basic rectangles i and j.

Dead space (White space)

- Dead space is the space that is wasted:

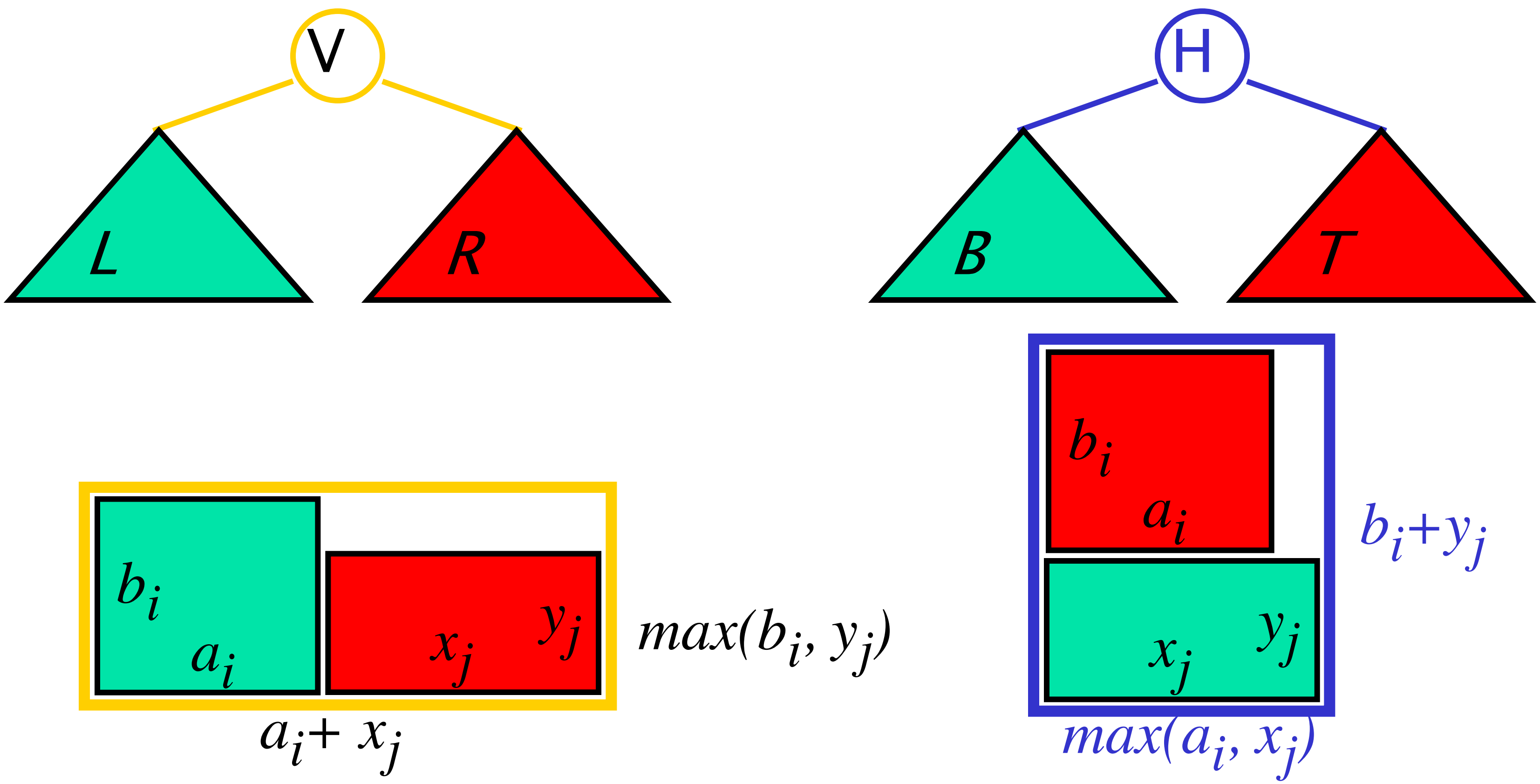


- Minimizing area is the same as minimizing deadspace.
- Dead space percentage is computed as

$$(A - \Sigma_i A_i) / A \times 100\%$$

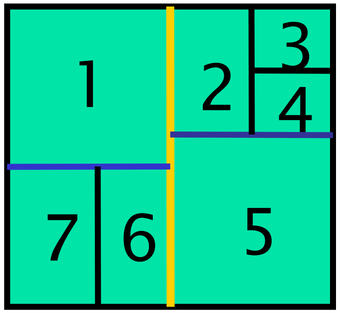
Floorplan Sizing for Slicing Floorplans

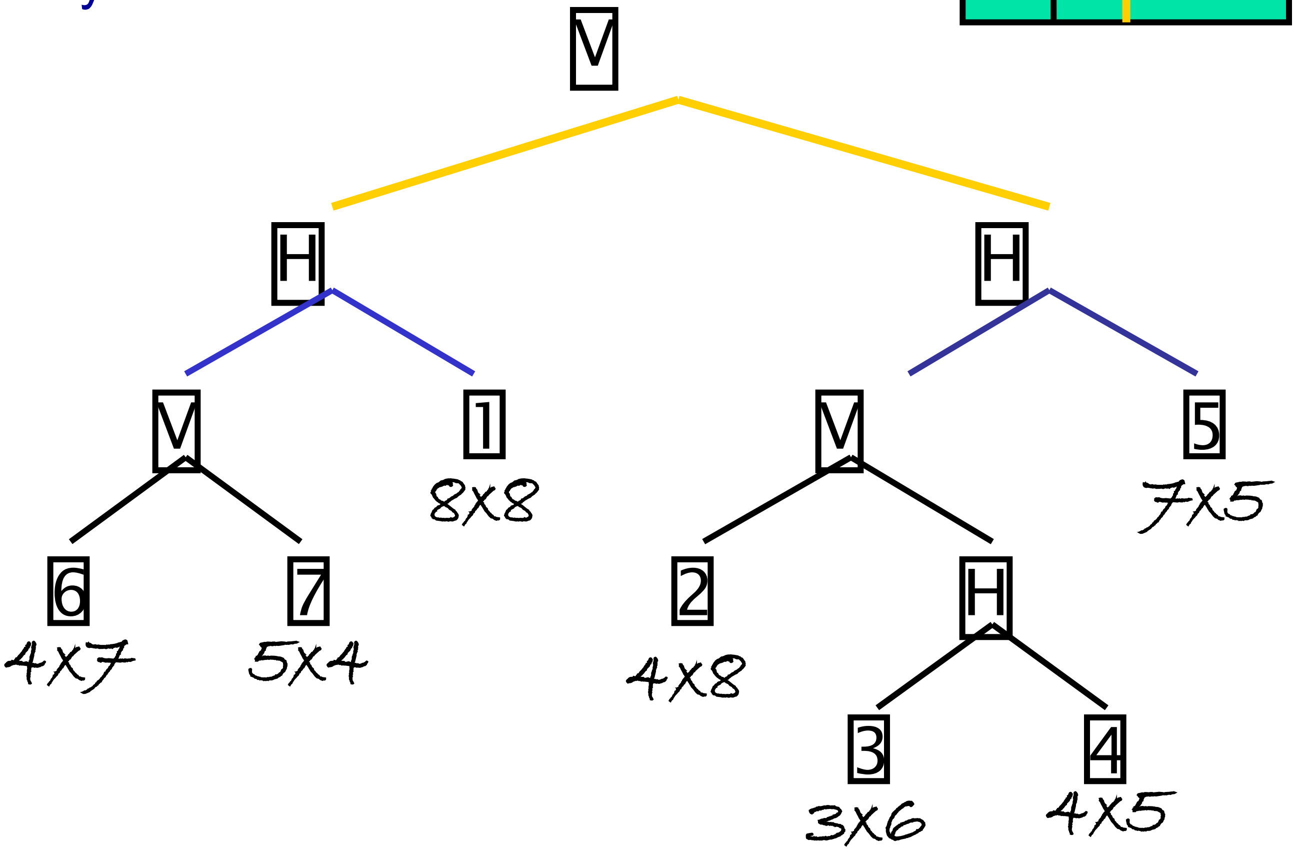
- Bottom-up process
- Has to be done per floorplan perturbation
- Requires O(n) time.
  - n is the total number of shapes of all the modules

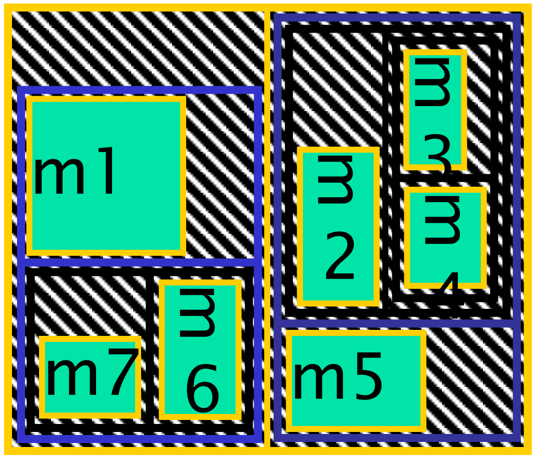


Sizing Slicing Floorplans

- Simple case:
  - All modules are hard macros
  - No rotation allowed
    ➔ one shape only

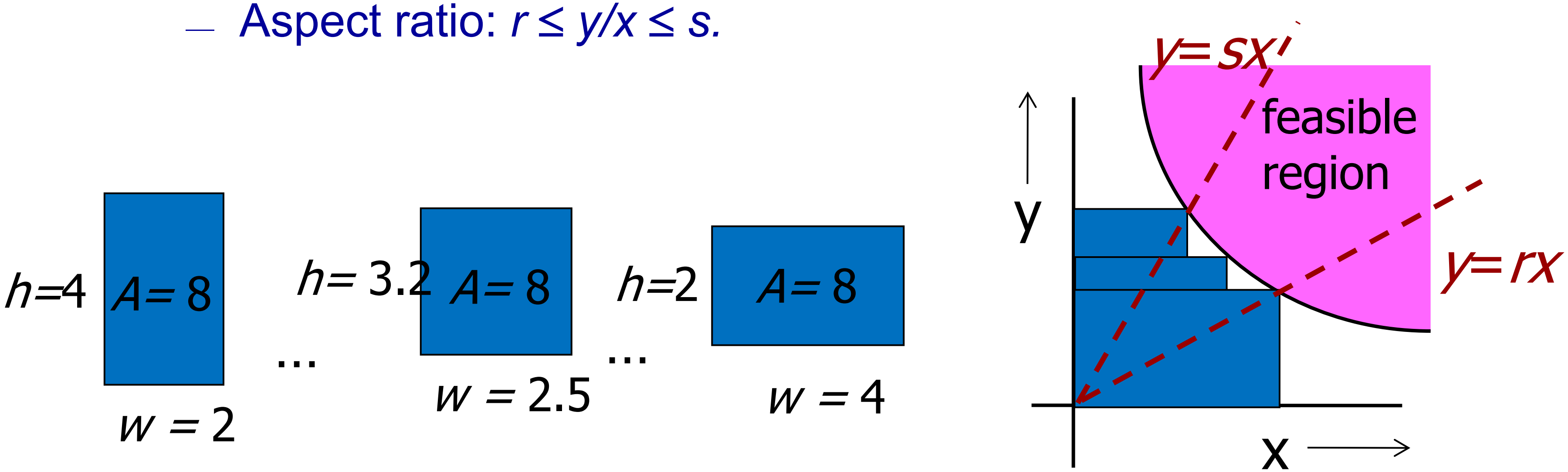






Bounding curve for the soft block

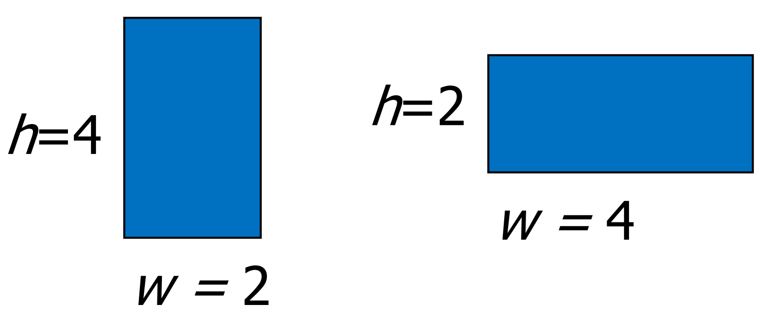
- A soft (flexible) blocks *b* can have different aspect ratios, but is with a fixed area *A*.
- The shape function of *b* is a hyperbola: *xy* **=** *A*, or *y* **=***A*/x, for width *x* and height *y*.
- Very thin blocks are often not interesting and feasible to design
  - Add two straight lines for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



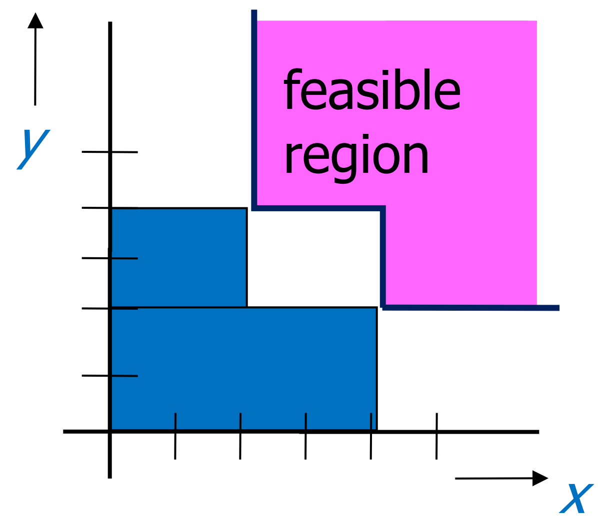
(a) shapes of a soft block b (b) bonding curves of the block

Bounding Curve for the Hard Block

- 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.



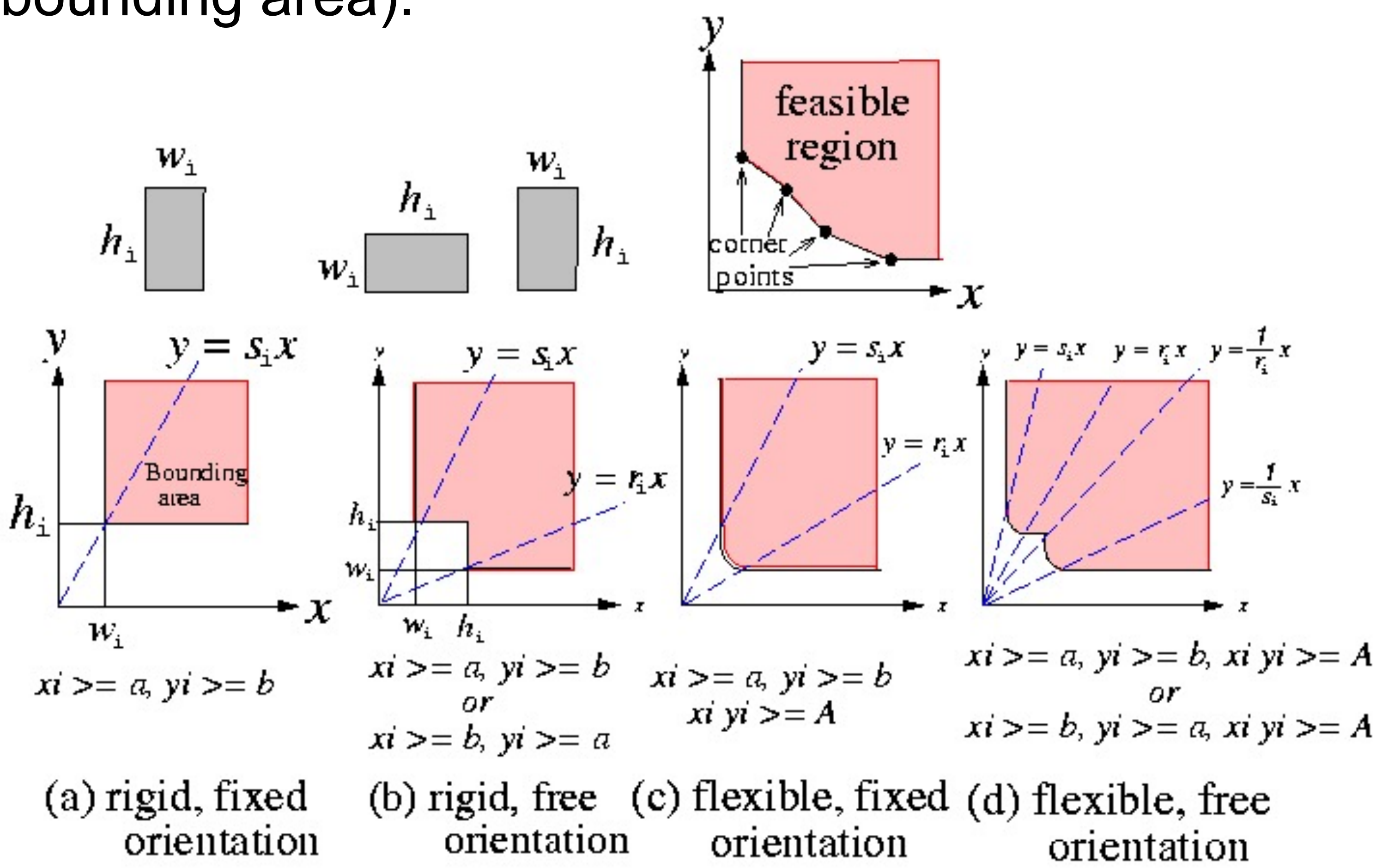
(a) shapes of a hard block b



(a) bonding curves of the block

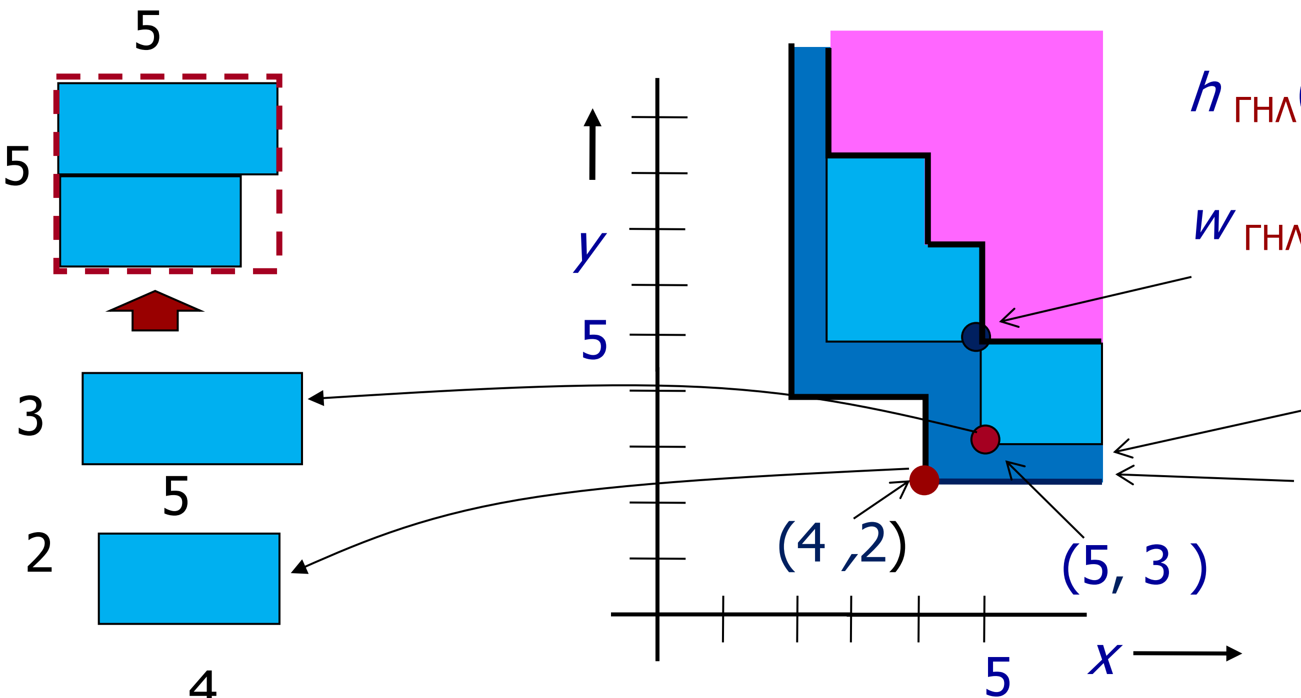
Bounding Curves for Various Modules

- Bonding curves correspond to different kinds of constraints where the shaded areas are feasible regions (bounding area).



Composition of Bounding Curves

- The resulting bounding curve after applying operations on curves **Γ** and **Λ** for rigid blocks are defined as follows:
    1. Horizontal cut operation: $\Gamma H \Lambda = \{ (w, h_1 + h_2) \mid (w_1, h_1) \in \Gamma$ and $(w_2, h_2) \in \Lambda$ and $w = \max \{w_1, w_2\} \}$
    2. Vertical cut operation: $\Gamma V \Lambda = \{ (w_1 + w_2, h) \mid (w_1, h_1) \in \Gamma$ and $(w_1, h_2) \in \Lambda$ and $h = \max \{h_1, h_2\}\}$



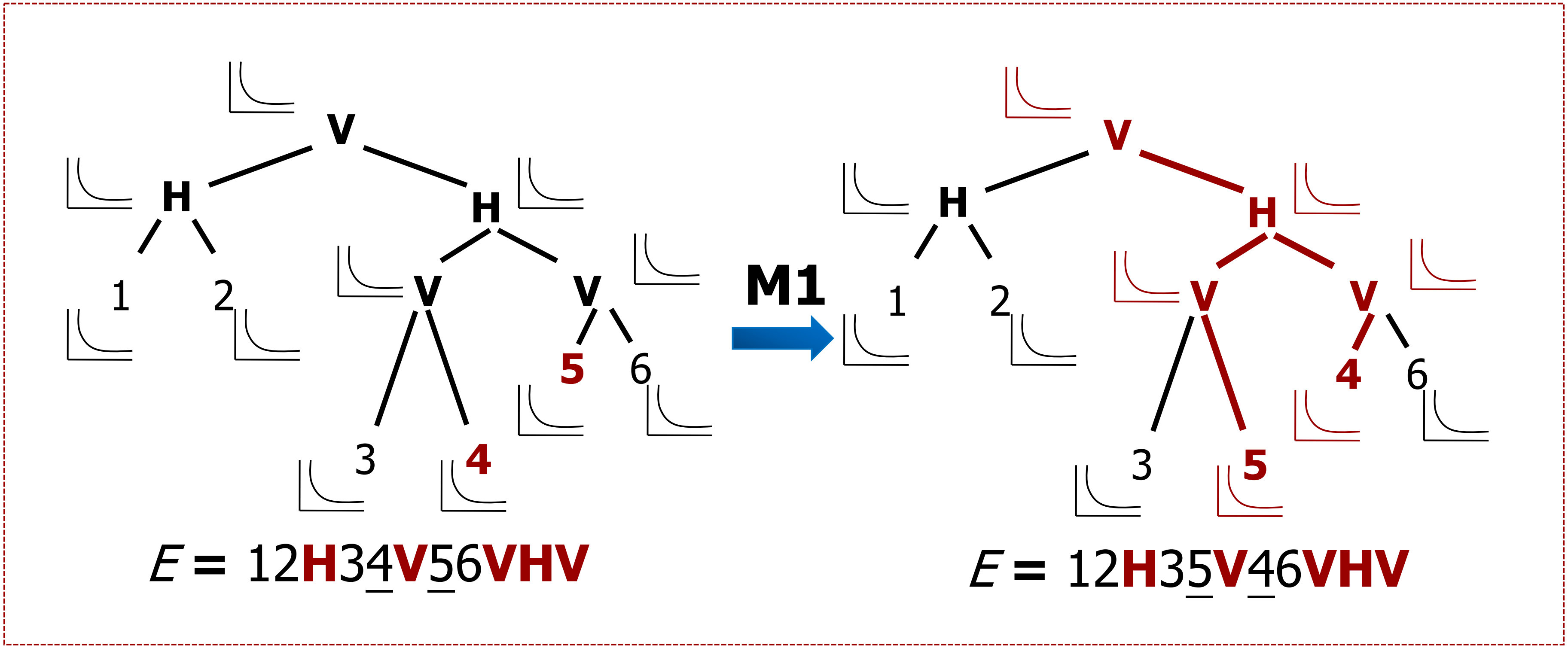
$h_{\Gamma H \Lambda}(w) = h_\Gamma + h_\Lambda$
$=3+2= 5$

$w_{\Gamma H \Lambda}(w) = \max \{w_\Gamma, w_\Lambda\}$
$= \max \{5, 4\} = 5$

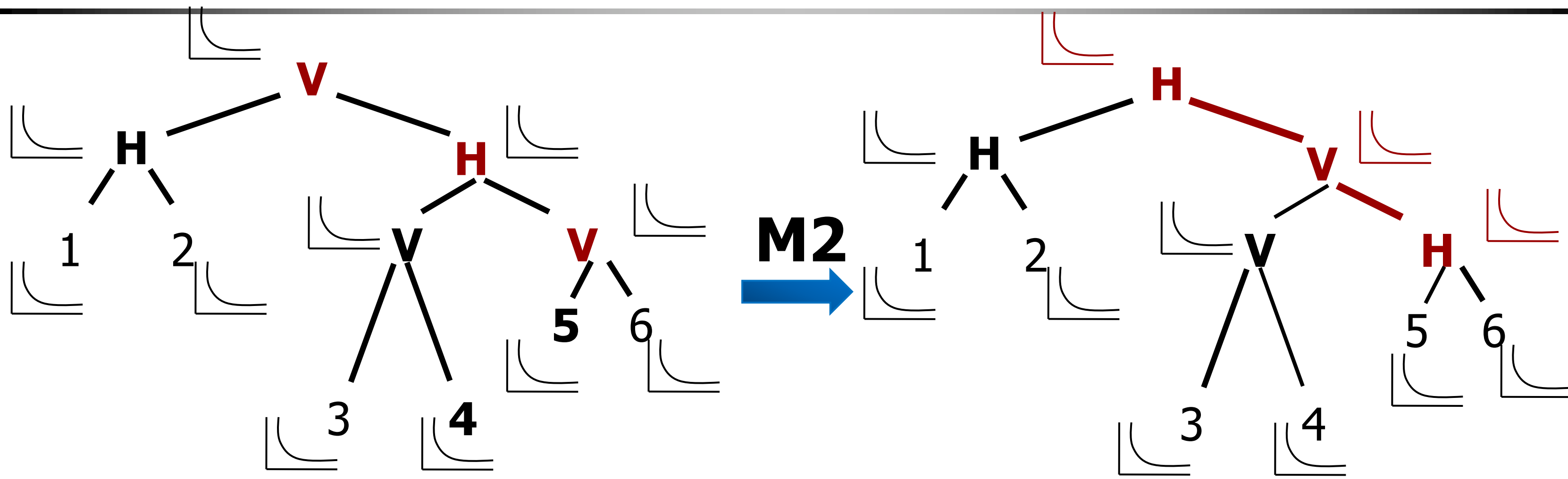
Horizontal abutment of two bonding curves Γ and Λ (i.e., Γ**H**Λ)

Incremental Computation of Cost Function

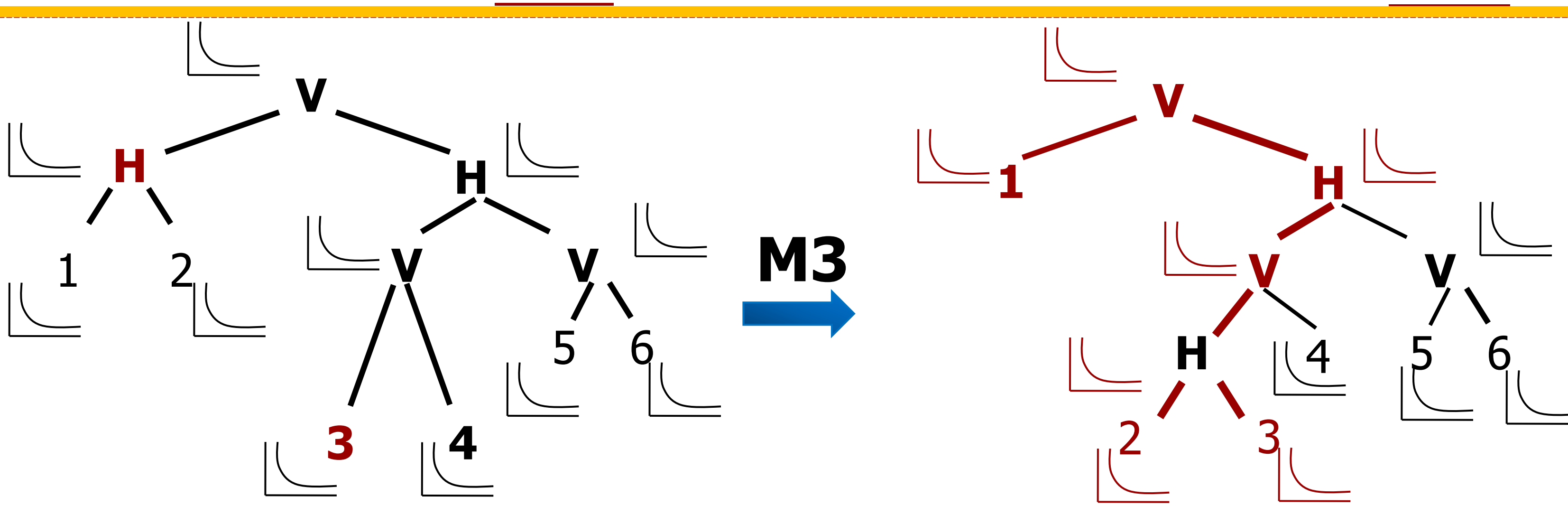
- Each move leads to only a minor modification of the Polish expression.
- At most **two paths** of the slicing tree need to be updated for each move.



Incremental Computation of Cost Function



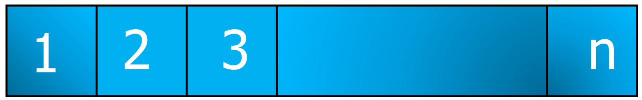
$E = 12H34V56\underline{VHV}$ $\quad$ $E = 12H34V56\underline{HVH}$



$E = 12\underline{H3}4V56VHV$ $\quad$ $E = 12\underline{3H}4V56VHV$

Annealing Schedule

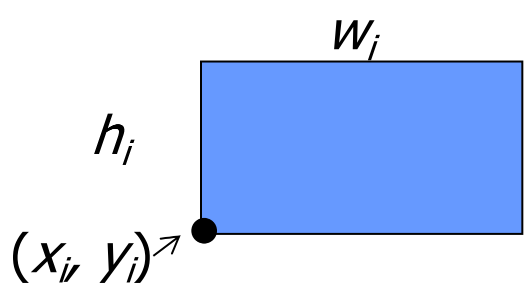
- Initial solution: $12V3V \ldots nV$.



- $T_i = r^i T_0$, $i = 1, 2, 3, \ldots$; $r = 0.85$.
- 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.

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 = $h_i * w_i$
Aspect ratio = w_i/h_i

Nonoverlap Constraints

- Two modules M_i and M_j are nonoverlap, if at least one of the following linear constraints is satisfied :

if M_i to the left of M_j:	$x_i + w_i \leq x_j$
if M_i below M_j:	$y_i + h_i \leq y_j$
if M_i to the right of M_j:	$x_i - w_j \geq x_j$
if M_i above M_j:	$y_i - h_j \geq x_j$

- 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 for the second equation listed bellow):

	p_{ij}	q_{ij}
$x_i + w_i \leq x_j + W(p_{ij} + q_{ij})$	0	0
$y_i + h_i \leq y_j + H(1 + p_{ij} - q_{ij})$	0	1
$x_i - w_j \geq x_j - W(1 - p_{ij} + q_{ij})$	1	0
$y_i - h_j \geq x_j - H(2 - p_{ij} - q_{ij})$	1	1

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. $x_i \ge 0$, $y_i \ge 0$, $1 \le \mathrm{i} \le n$;
  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

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

- 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: glpk, lp_solve, cplex, etc.

Mixed ILP for Floorplanning (cont'd)

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

$$\min \quad y$$

subject to

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

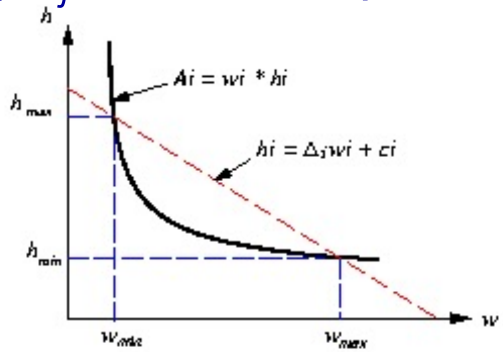
- For each module i with free orientation, associate a 0-1 variable r_i:
  - $r_i = 0$: 0^o rotation for module i.
  - $r_i = 1$: 90^o rotation for module i.
- $M = \max\{W, H\}$.

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}).

$$
\begin{aligned}
h_i &= \Delta_i w_i + c_i && /*\ \ y = mx + c\ \ */ \\
\Delta_i &= \frac{h_{max} - h_{min}}{w_{min} - w_{max}} && /*\ \ slope\ \ */ \\
c_i &= h_{max} - \Delta_i w_{min} && /*\ \ c = y_0 - m x_0\ \ */
\end{aligned}
$$

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



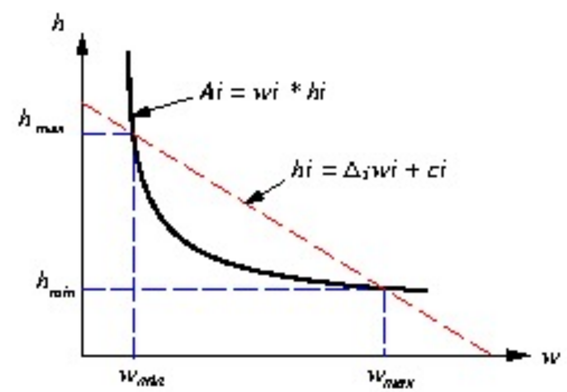
Flexible/Soft Modules

- Assumptions: w_i, h_i are unknown; area lower bound: A_i.
- Module size constraints: $w_i\, h_i \geq A_i$.
- Hence, $h_i = A_i / w_i = f(w_i)$
- Apply Taylor's series expansion for the above equation:

$$f(w_i) = h_i = A_i / w_{i,max} + A_i(w_{i,max} - w_i)/w^2_{i,max} + O(w_i - w_{i,max})$$

Let $h_{i,0} = A_i / w_{i,max}$, $\Delta_i = w_{i,\,max} - w_i$, and $\lambda_i = A_i / w^2_{i,max}$

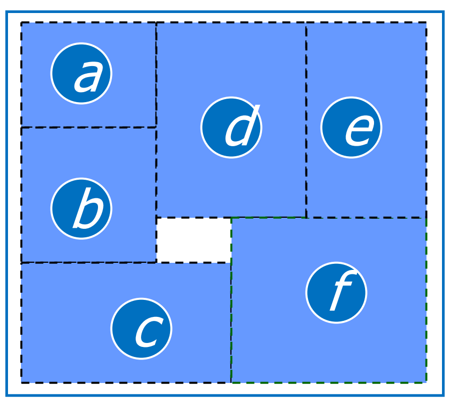
Then $h_i = h_{i,0} + \lambda_i\, \Delta_i$



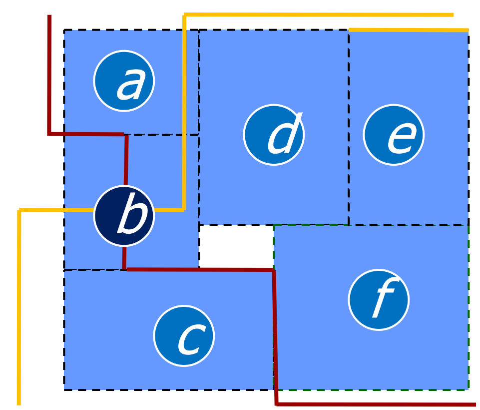
$$
\begin{array}{llll}
x_i + w_i \leq x_j + W(p_{ij} + q_{ij}) & \rightarrow & x_i + w_{i,max} - \Delta_i & \leq x_j + W(p_{ij}+q_{ij}) \\
y_i + h_i \leq y_j + H(1 + p_{ij} - q_{ij}) & \rightarrow & y_i + h_{i,0} + \lambda_i\, \Delta_i & \leq y_j + H(l + p_{ij} - q_{ij}) \\
x_i - w_j \geq x_j - W(1 - p_{ij} + q_{ij}) & \rightarrow & x_i - w_{j,max} + \Delta_j & \geq x_j - W(1 - p_{ij} + q_{ij}) \\
y_i - h_j \geq x_j - H(2 - p_{ij} - q_{ij}) & \rightarrow & y_i - h_{i,0} - \lambda_j\, \Delta_j & \geq y_j - H(2 - p_{ij} - q_{ij})
\end{array}
$$

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 (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 floorpaln

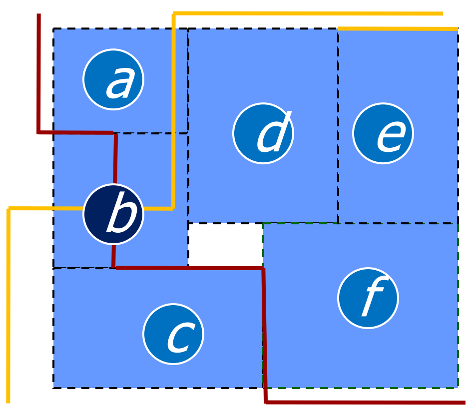


Loci of module *b*

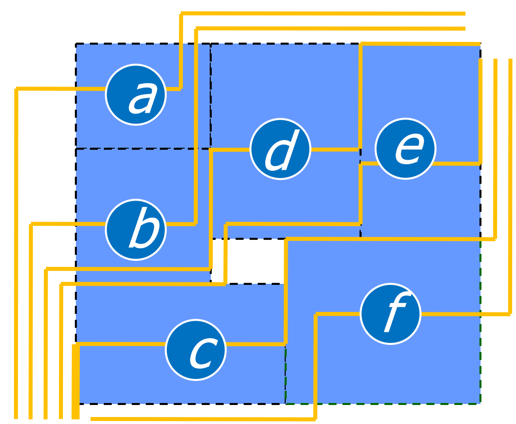
Relative Module Positions

- 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.

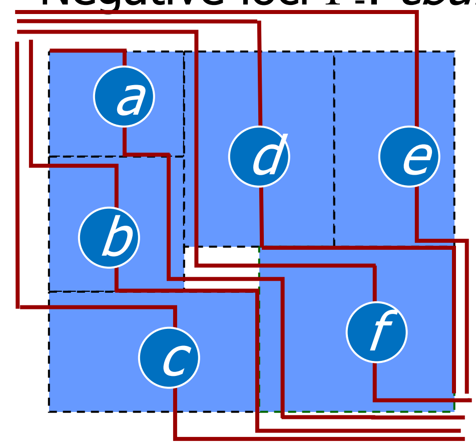
Negative loci Γ_-: *cbafde*



Loci of module: *b*

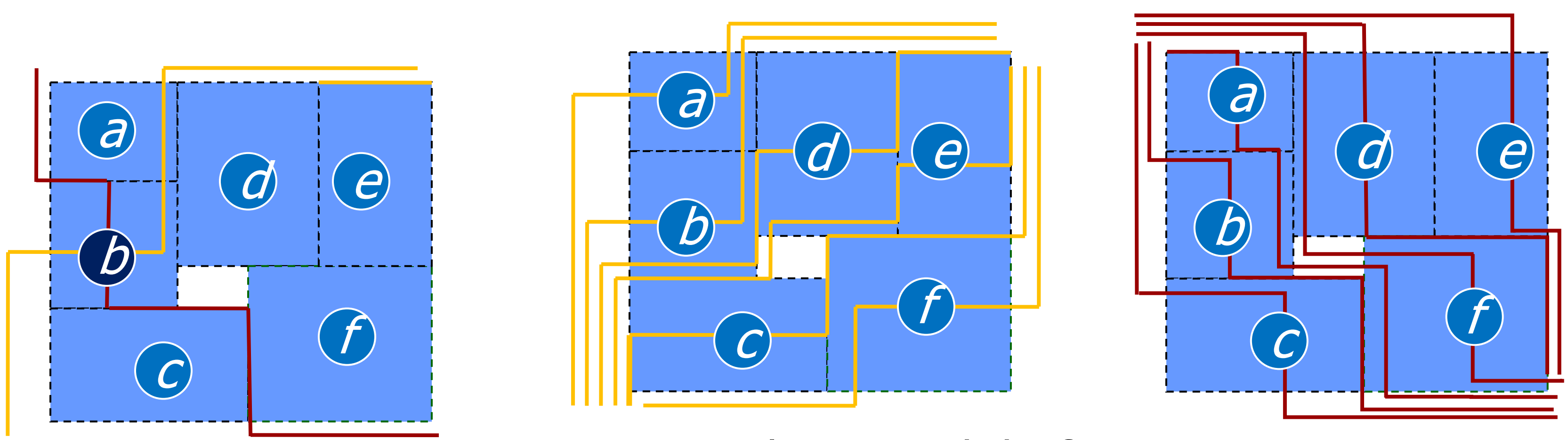


Positive loci Γ_+: *abdecf*



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*

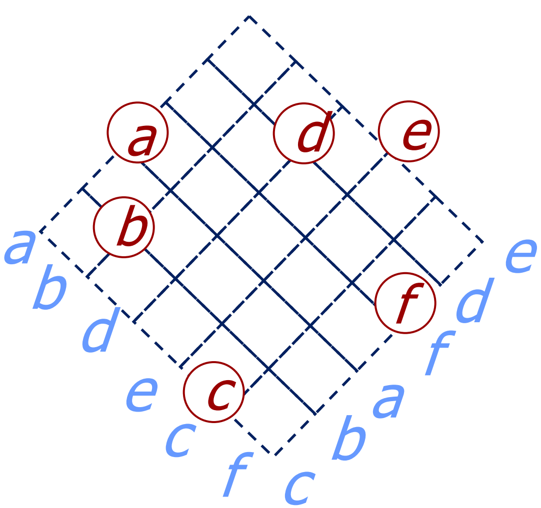


Loci of module: *b* — Positive loci Γ_+: *abdecf* — Negative loci Γ_-: *cbafde*

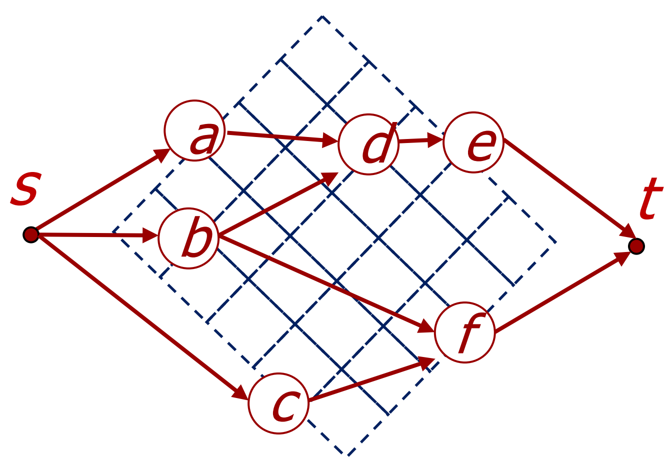
$(\Gamma_+, \Gamma_-) = (abdecf, cbafde)$

(Γ_+, Γ_-)-Packing

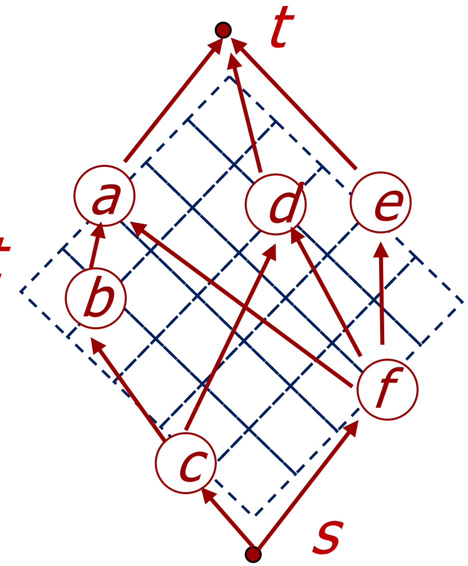
- For every SP (Γ_+, Γ_-), there is a (Γ_+, Γ_-) packing.
- **Horizontal constraint graph** $G_H(V, E)$ (similarly for **vertical constraint graph** $G_V(V, E)$):
  - V: source s, sink t, n vertices for modules.
  - E: (s, x) $((x, t))$ for the module x without module left (right) to it, and (x, y) iff 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: (*abdecf, cbafde*)



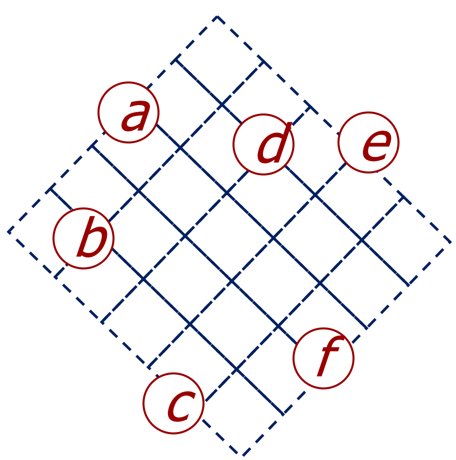
Horizontal constraint graph (*Transitive edges are not shown*)



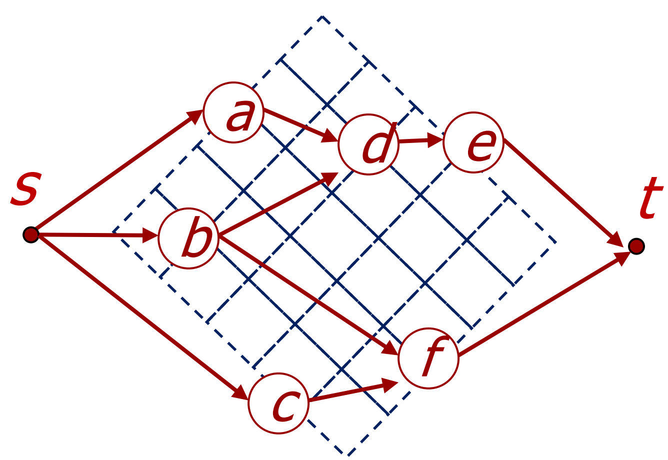
Vertical constraint graph (*Transitive edges are not shown*)

Cost Evaluation

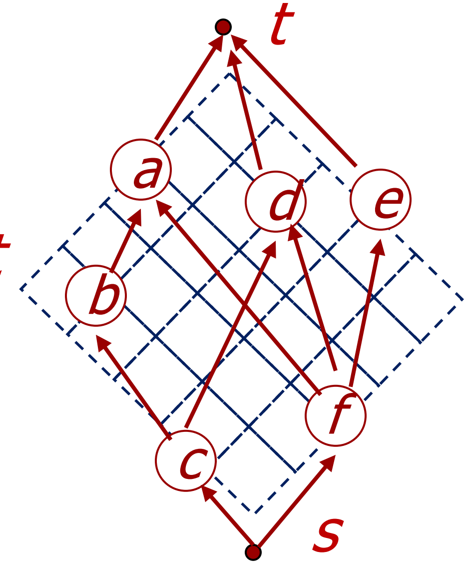
- **Optimal** (Γ_+, Γ_-)-Packing can be obtained in $O(n^2)$ time by applying a longest path algorithm on a vertex-weighted directed acyclic graph.
  - G_H and G_V are independent.
  - The X and Y coordinates of each module are the minimum values of the longest path length between s and the corresponding 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, DATE-2K, ASP-DAC-01)



Packing for sequence pair:
(*abdecf, cbfade*)



Horizontal constraint graph
(*Transitive edges are not shown*)



Vertical constraint graph
(*Transitive edges are not shown*)