# Partitioning and Floorplanning

Virendra Singh
Indian Institute of Science
Bangalore

E0-285: CAD of VLSI Systems

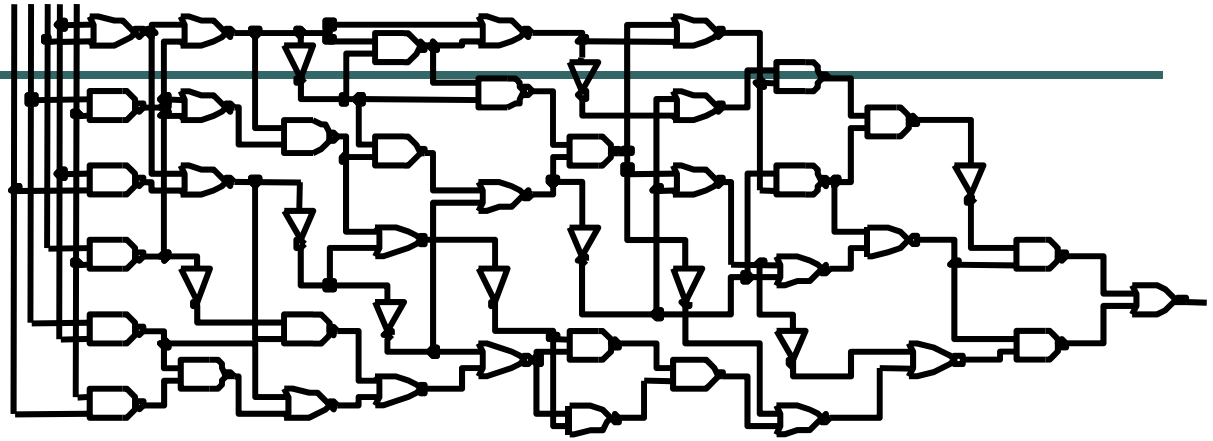
# Partitioning

- Decomposition of a complex system into smaller subsystems
  - Done hierarchically
  - Partitioning done until each subsystem has manageable size
  - Each subsystem can be designed independently
- Interconnections between partitions minimized
  - Less hassle interfacing the subsystems
  - Communication between subsystems usually costly

[©Sherwani]

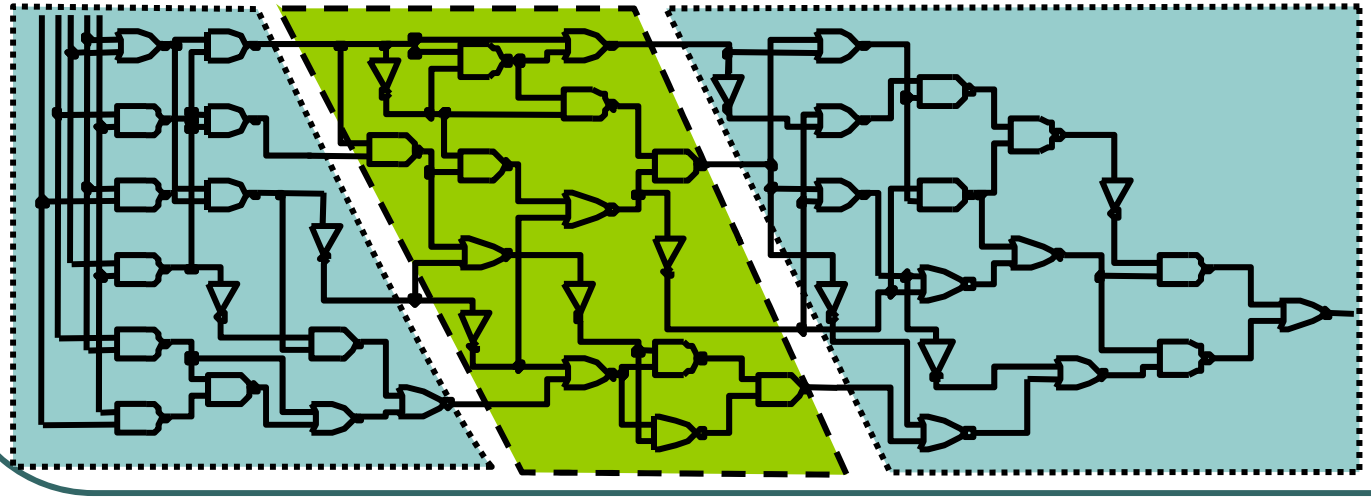
# Example: Partitioning of a Circuit

Input size: 48



Cut 1=4 Cut 2=4

Size 1=15 Size 2=16 Size 3=17



[©Sherwani]

# Partitioning: Formal Definition

- Input:
  - Graph or hypergraph
  - Usually with vertex weights
  - Usually weighted edges
- Constraints
  - Number of partitions (K-way partitioning)
  - Maximum capacity of each partition
    OR
    maximum allowable difference between partitions
- Objective
  - Assign nodes to partitions subject to constraints
    s.t. the cutsize is minimized
- Tractability
  - Is NP-complete ☹

# Kernighan-Lin (KL) Algorithm

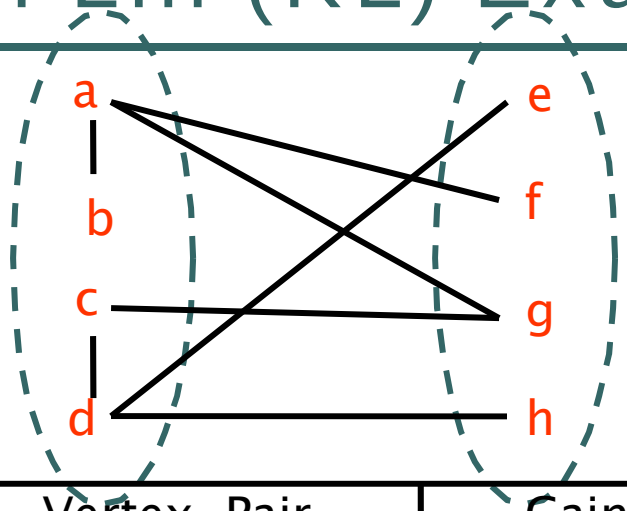
- On non-weighted graphs
- An iterative improvement technique
- A two-way (bisection) partitioning algorithm
- The partitions must be balanced (of equal size)
- Iterate as long as the cutsize improves:
  - Find a pair of vertices that result in the largest decrease in cutsize if exchanged
  - Exchange the two vertices (potential move)
  - “Lock” the vertices
  - If no improvement possible, and
    still some vertices unlocked, then
    exchange vertices that result in smallest increase in cutsize

W. Kernighan and S. Lin, Bell System Technical Journal, 1970.

# Kernighan-Lin (KL) Algorithm

- Initialize
  - Bipartition G into $V_1$ and $V_2$, s.t., $|V_1| = |V_2| \pm 1$ *greedy*
  - n = |V|
- Repeat
  - for i=1 to n/2
    - Find a pair of unlocked vertices $v_{ai} \in V_1$ and $v_{bi} \in V_2$ whose exchange makes the largest decrease or smallest increase in cut-cost
    - Mark $v_{ai}$ and $v_{bi}$ as locked
    - Store the gain $g_i$.
  - Find k, s.t. $\sum_{i=1..k} g_i = Gain_k$ is maximized
  - If $Gain_k > 0$ then
    move $v_{a1},...,v_{ak}$ from $V_1$ to $V_2$ and
    $v_{b1},...,v_{bk}$ from $V_2$ to $V_1$.
- Until $Gain_k \leq 0$

# Kernighan-Lin (KL) Example

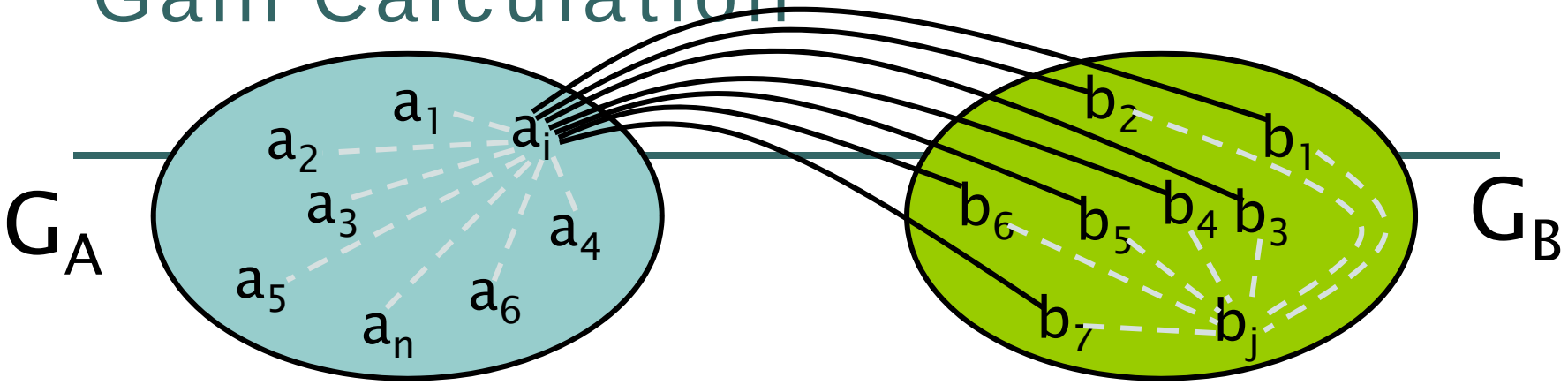



| Step No. | Vertex Pair | Gain | Cut-cost |
|---|---|---|---|
| 0 | -- | 0 | 5 |
| 1 | { d, g } | 3 | 2 |
| 2 | { c, f } | 1 | 1 |
| 3 | { b, h } | -2 | 3 |
| 4 | { a, e } | -2 | 5 |

# Kernighan-Lin (KL) : Analysis

- Time complexity?
  - Inner (for) loop
    - Iterates n/2 times
    - Iteration 1: (n/2) x (n/2)
    - Iteration i: $(n/2 – i + 1)^2$.
  - Passes? Usually independent of n
  - $O(n^3)$
- Drawbacks?
  - Local optimum
  - Balanced partitions only
  - No weight for the vertices
  - High time complexity
  - Only on edges, not hyper-edges

# Gain Calculation




$$I_{a_i} = \sum_{x \in A} C_{a_i x}, \qquad E_{a_i} = \sum_{y \in B} C_{a_i y}$$

Likewise,

$$D_{a_i} = E_{a_i} - I_{a_i}$$

$$D_{b_j} = E_{b_j} - I_{b_j} = \sum_{x \in A} C_{b_j x} - \sum_{y \in B} C_{b_j y}$$

# Gain Calculation (cont.)

- Lemma: Consider any $a_i \in A$, $b_j \in B$.
  If $a_i$, $b_j$ are interchanged, the gain is

$$g = D_{a_i} + D_{b_j} - 2C_{a_i b_j}$$

- Proof:

  Total cost before interchange (T) between A and B

$$T = E_{a_i} + E_{b_j} - C_{a_i b_j} + (\text{cost for all others})$$

  Total cost after interchange (T') between A and B

$$T = I_{a_i} + I_{b_j} + C_{a_i b_j} + (\text{cost for all others})$$

  Therefore

$$g = T - T' = \underbrace{E_{a_i} - I_{a_i}}_{D_{a_i}} + \underbrace{E_{b_j} - I_{b_j}}_{D_{b_j}} - 2C_{a_i b_j}$$

[©Kang]

# Gain Calculation (cont.)

- Lemma:
  - Let $D_x$', $D_y$' be the new D values for elements of A - $\{a_i\}$ and B - $\{b_j\}$. Then after interchanging $a_i$ & $b_j$,

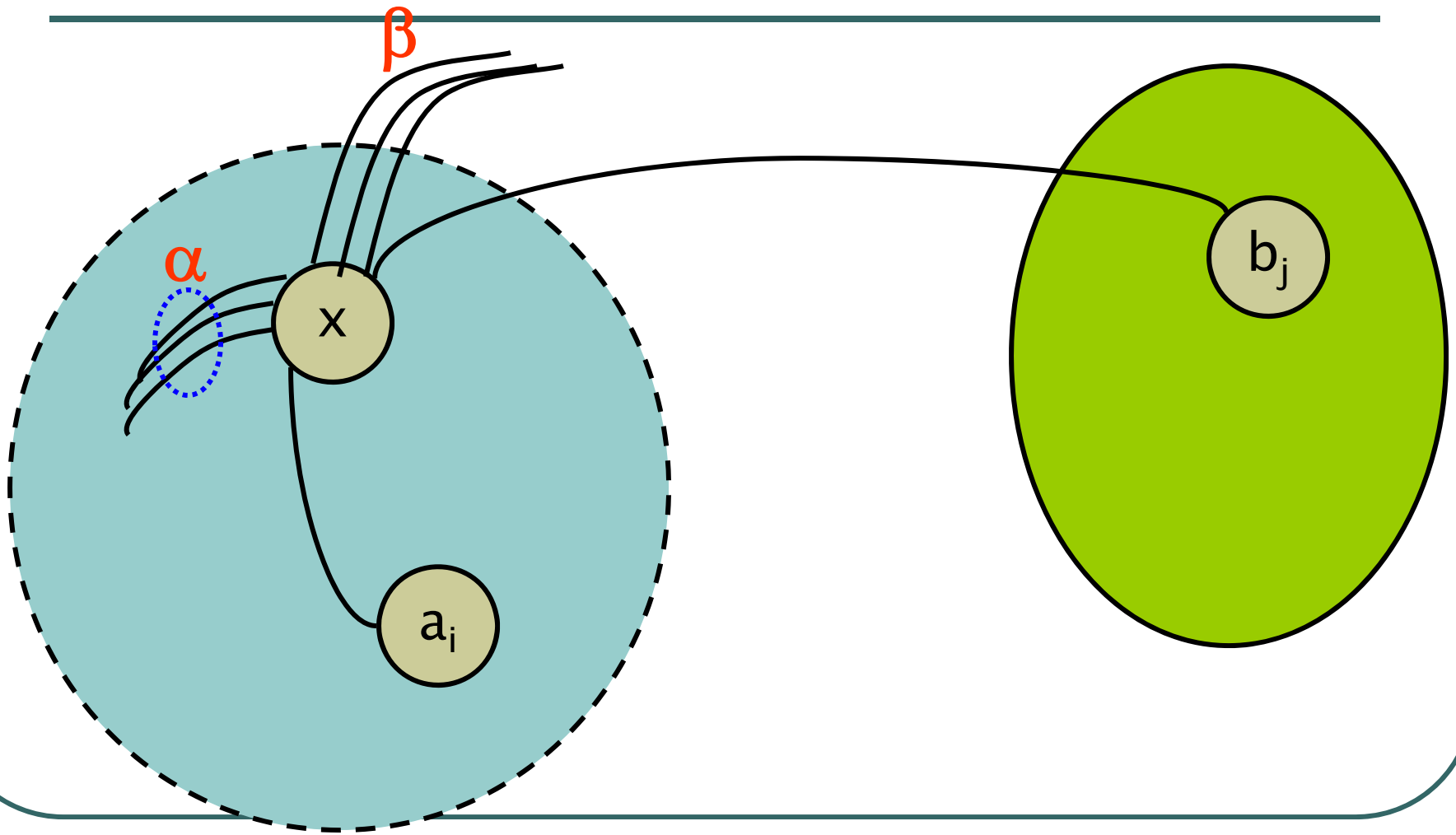
$$D_x' = D_x + 2C_{xa_i} - 2C_{xb_j} \quad , \quad x \in A - \{a_i\}$$

$$D_y' = D_y + 2C_{yb_j} - 2C_{ya_i} \quad , \quad y \in B - \{b_j\}$$

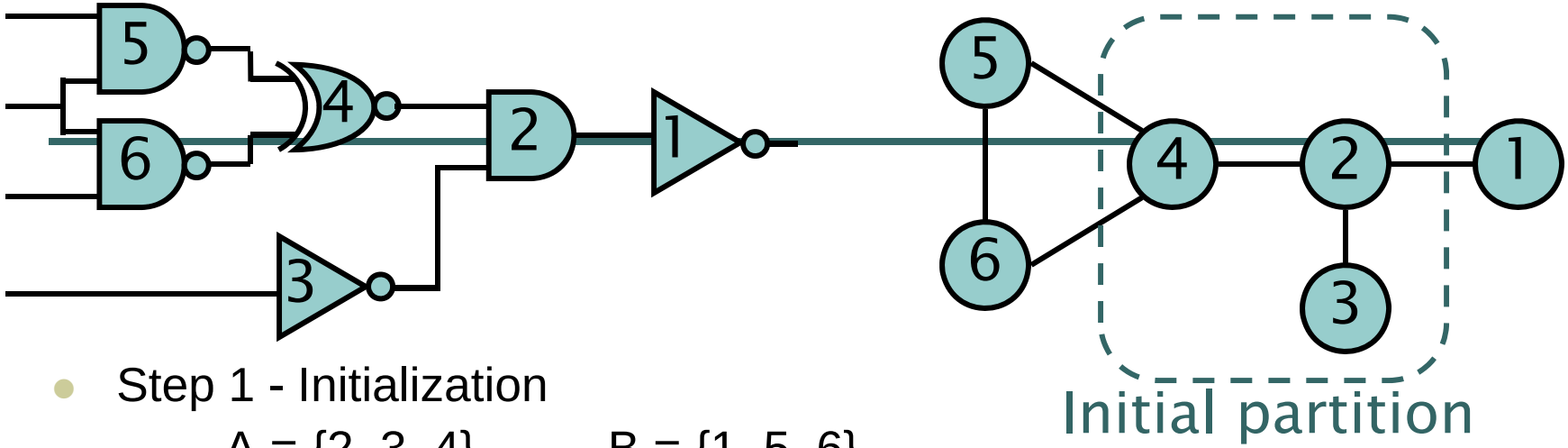
- Proof:
  - The edge x-$a_i$ changed from internal in $D_x$ to external in $D_x$'
  - The edge y-bj changed from internal in $D_x$ to external in $D_x$'
  - The x-$b_j$ edge changed from external to internal
  - The y-$a_i$ edge changed from external to interna

[©Kang]

# Clarification of the Lemma



# Example: KL




- Step 1 - Initialization

  A = {2, 3, 4}, B = {1, 5, 6}

  A' = A = {2, 3, 4}, B' = B = {1, 5, 6}

- Step 2 - Compute D values

  $D_1 = E_1 - I_1 = 1\text{-}0 = +1$

  $D_2 = E_2 - I_2 = 1\text{-}2 = -1$

  $D_3 = E_3 - I_3 = 0\text{-}1 = -1$

  $D_4 = E_4 - I_4 = 2\text{-}1 = +1$

  $D_5 = E_5 - I_5 = 1\text{-}1 = +0$

  $D_6 = E_6 - I_6 = 1\text{-}1 = +0$

[©Kang]

# Example: KL (cont.)

- Step 3 - compute gains

  $g_{21} = D_2 + D_1 - 2C_{21} = (-1) + (+1) - 2(1) = -2$

  $g_{25} = D_2 + D_5 - 2C_{25} = (-1) + (+0) - 2(0) = -1$

  $g_{26} = D_2 + D_6 - 2C_{26} = (-1) + (+0) - 2(0) = -1$

  $g_{31} = D_3 + D_1 - 2C_{31} = (-1) + (+1) - 2(0) = 0$

  $g_{35} = D_3 + D_5 - 2C_{35} = (-1) + (0) - 2(0) = -1$

  $g_{36} = D_3 + D_6 - 2C_{36} = (-1) + (0) - 2(0) = -1$

  $g_{41} = D_4 + D_1 - 2C_{41} = (+1) + (+1) - 2(0) = +2$

  $g_{45} = D_4 + D_5 - 2C_{45} = (+1) + (+0) - 2(+1) = -1$

  $g_{46} = D_4 + D_6 - 2C_{46} = (+1) + (+0) - 2(+1) = -1$

- The largest g value is $g_{41} = +2$

  $\Longrightarrow$interchange 4 and 1 $(a_1, b_1) = (4, 1)$

  A' = A' - {4} = {2, 3}

  B' = B' - {1} = {5, 6}  both not empty

# Example: KL (cont.)

*internal from 4 to 2 become external nodes*

- Step 4 - update D values of node connected to vertices (4, 1)

  $D_2' = D_2 + 2C_{24} - 2C_{21} = (-1) + 2(+1) - 2(+1) = -1$

  $D_5' = D_5 + 2C_{51} - 2C_{54} = +0 + 2(0) - 2(+1) = -2$

  $D_6' = D_6 + 2C_{61} - 2C_{64} = +0 + 2(0) - 2(+1) = -2$

- Assign $D_i = D_i'$, repeat step 3 :

  $g25 = D_2 + D_5 - 2C_{25} = -1 - 2 - 2(0) = -3$

  $g26 = D_2 + D_6 - 2C_{26} = -1 - 2 - 2(0) = -3$

  $g35 = D_3 + D_5 - 2C_{35} = -1 - 2 - 2(0) = -3$

  $g36 = D_3 + D_6 - 2C_{36} = -1 - 2 - 2(0) = -3$

- All values are equal;
  arbitrarily choose $g_{36} = -3 \Rightarrow$ (a2, b2) = (3, 6)

  $A' = A' - \{3\} = \{2\}, \quad B' = B' - \{6\} = \{5\}$

  New D values are:

  $D_2' = D_2 + 2C_{23} - 2C_{26} = -1 + 2(1) - 2(0) = +1$

  $D_5' = D_5 + 2C_{56} - 2C_{53} = -2 + 2(1) - 2(0) = +0$

- New gain with $D_2 \leftarrow D_2'$, $D_5 \leftarrow D_5'$

  $g_{25} = D_2 + D_5 - 2C_{52} = +1 + 0 - 2(0) = +1 \Rightarrow$ (a3, b3) = (2, 5)

# Example: KL (cont.)

- Step 5 - Determine the # of moves to take

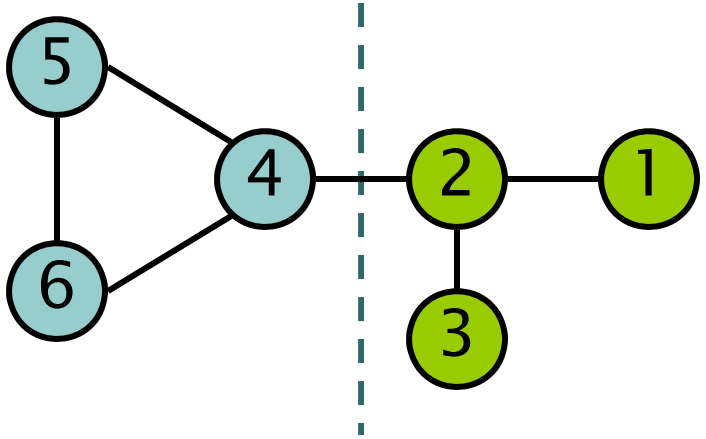
  $g_1 = +2$

  $g_1 + g_2 = +2 - 3 = -1$

  $g_1 + g_2 + g_3 = +2 - 3 + 1 = 0$
- The value of k for max G is 1

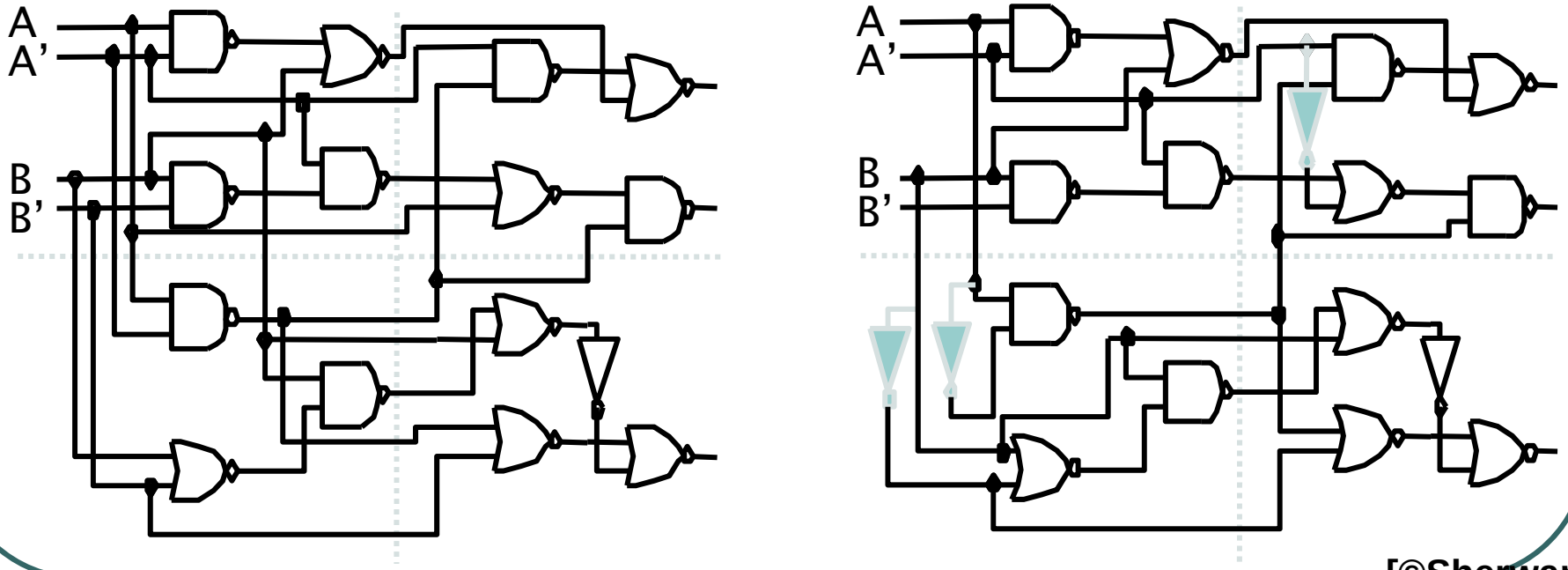
  $X = \{a_1\} = \{4\}$, $Y = \{b_1\} = \{1\}$
- Move X to B, Y to A $\Rightarrow$ A = {1, 2, 3}, B = {4, 5, 6}
- Repeat the whole process:

  • • • • •
- The final solution is A = {1, 2, 3}, B = {4, 5, 6}



# Subgraph Replication to Reduce Cutsize

- Vertices are replicated to improve cutsize
- Good results if limited number of components replicated

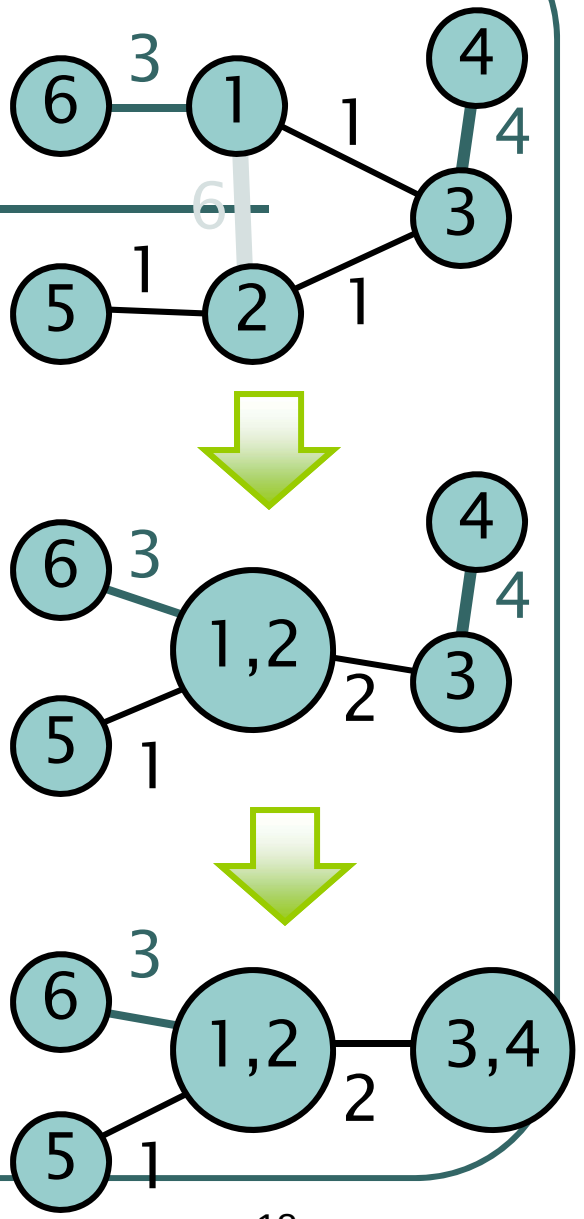



C. Kring and A. R. Newta, ICCAD, 1991.

[©Sherwani]

# Clustering

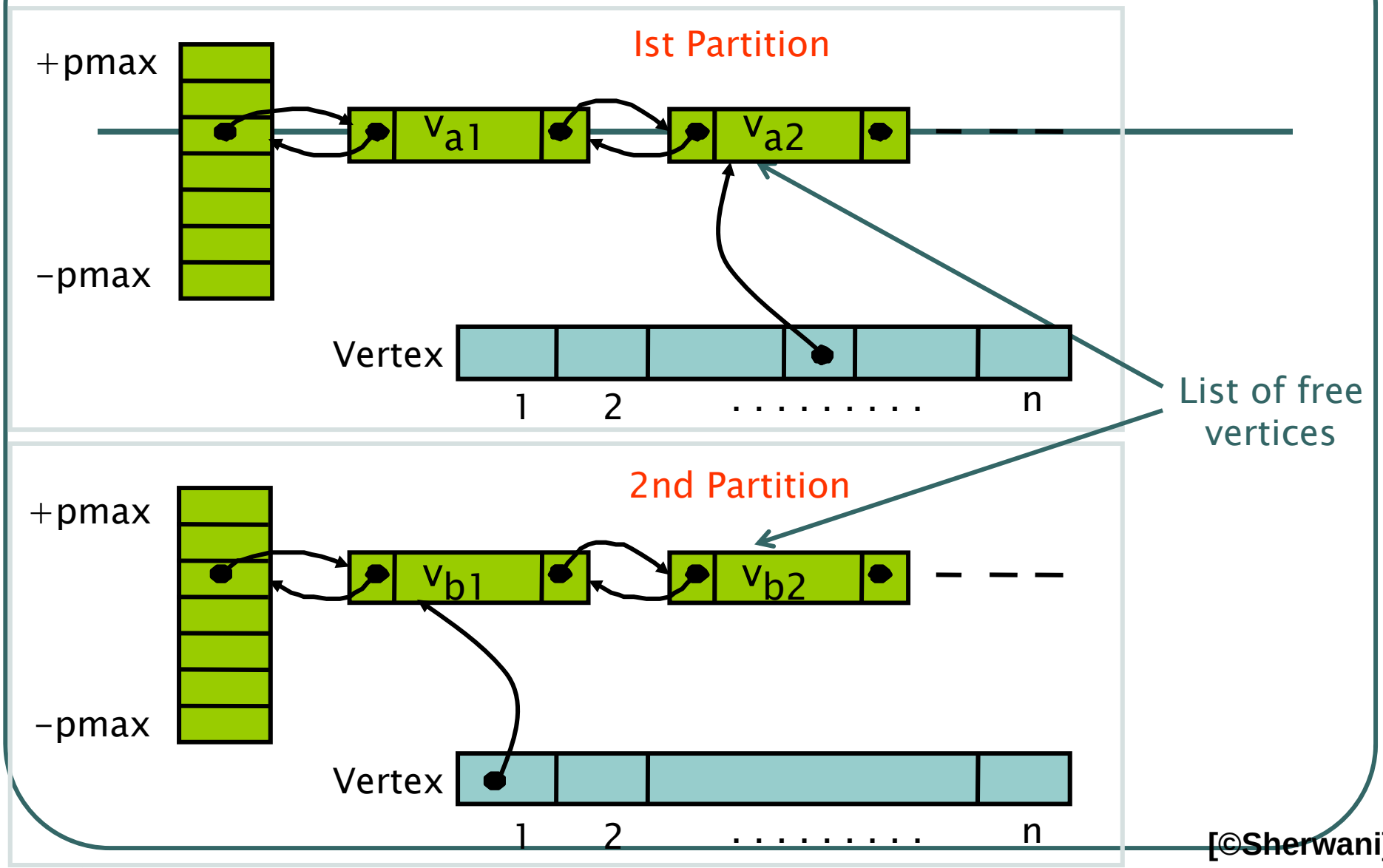
- Clustering
  - Bottom-up process
  - Merge heavily connected components into clusters
  - Each cluster will be a new “node”
  - “Hide” internal connections (i.e., connecting nodes within a cluster)
  - “Merge” two edges incident to an external vertex, connecting it to two nodes in a cluster
- Can be a preprocessing step before partitioning
  - Each cluster treated as a single node



# Fiduccia-Mattheyses (FM) Algorithm

- Modified version of KL
- A single vertex is moved across the cut in a single move
  - ➔ Unbalanced partitions
- Vertices are weighted
- Concept of cutsize extended to hypergraphs
- Special data structure to improve time complexity to $O(n^2)$
  - (Main feature)
- Can be extended to multi-way partitioning

# The FM Algorithm: Data Structure




[©Sherwani]

# The FM Algorithm: Data Structure

- Pmax
  - Maximum gain
  - $p_{max} = d_{max} . w_{max}$, where
    $d_{max}$ = max degree of a vertex (# edges incident to it)
    $w_{max}$ is the maximum edge weight
  - What does it mean intuitively?
- -Pmax .. Pmax array
  - Index $i$ is a pointer to the list of unlocked vertices with gain $i$.
- Limit on size of partition
  - A maximum defined for the sum of vertex weights in a partition (alternatively, the maximum ratio of partition sizes might be defined)

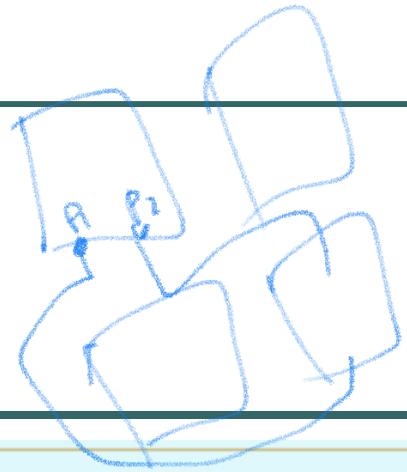
# The FM Algorithm

- Initialize
  - Start with a balance partition A, B of G
    (can be done by sorting vertex weights in decreasing order, placing them in A and B alternatively)
- Iterations
  - Similar to KL
  - A vertex cannot move if violates the balance condition
  - Choosing the node to move:
    pick the max gain in the partitions
  - Moves are tentative (similar to KL)
  - When no moves possible or no more unlocked vertices available, the pass ends
  - When no move can be made in a pass, the algorithm terminates

# Other Partitioning Methods

- KL and FM have each held up very well
- Min-cut / max-flow algorithms
  - Ford-Fulkerson – for unconstrained partitions
- Ratio cut
- Genetic algorithm
- Simulated annealing

# Partitioning



- **Input:**
  - A set of blocks, both fixed and flexible.
    - Area of the block $A_i = w_i \times h_i$
    - Constraint on the shape of the block (rigid/flexible)
  - Pin locations of fixed blocks.
  - A netlist.
- **Requirements:**
  - Find locations for each block so that no two blocks overlap.
  - Determine shapes of flexible blocks.
- **Objectives:**
  - Minimize area.
  - Reduce wire-length for critical nets.

# Difference between FP and Placement

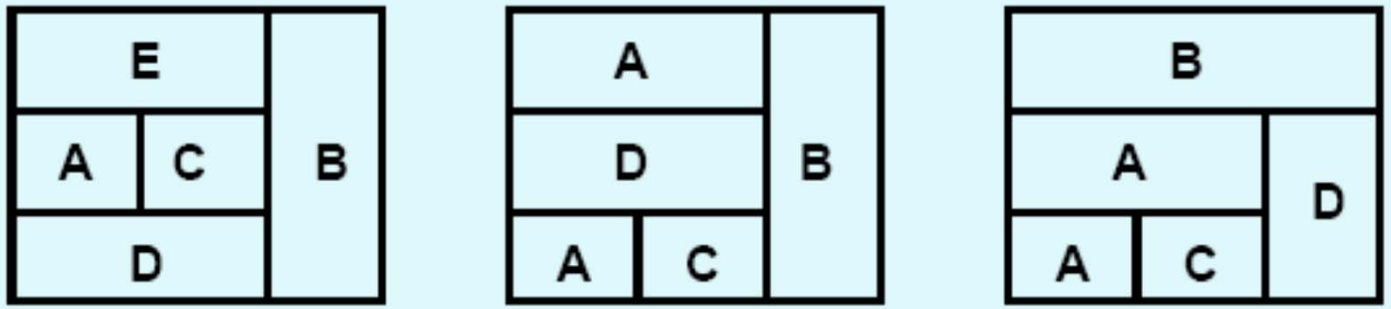
- The problems are similar in nature.
- Main differences:
  - In floorplanning, some of the blocks may be flexible, and the exact locations of the pins not yet fixed.
  - In placement, all blocks are assumed to be of well-defined geometrical shapes, with defined pin locations.
- Points to note:
  - Floorplanning problem is more difficult as compared to placement.
    - Multiple choice for the shape of a block.
  - In some of the VLSI design styles, the two problems are identical.

Bin Packing problem
I know L S W Objective is to minimize Combined Area

# Examples of Rigid Block

| Module | Width | Height |
|---|---|---|
| A | 1 | 1 |
| B | 1 | 3 |
| C | 1 | 1 |
| D | 1 | 2 |
| E | 2 | 1 |

## Some of the Feasible Floorplans



# Design Style Specific Issues

- **Full Custom**
  - All the steps required for general cells.
- **Standard Cell**
  - Dimensions of all cells are fixed.
  - Floorplanning problem is simply the placement problem.
  - For large netlists, two steps:
    - First do global partitioning.
    - Placement for individual regions next.
- **Gate Array**
  - Floorplanning problem same as placement problem.

# Cost of Floorplanning

- The number of feasible solutions of a floorplanning problem is very large.
  - Finding the best solution is NP-hard.
- Several criteria used to measure the quality of floorplans:
  a) Minimize area
  b) Minimize total length of wire
  c) Maximize routability
  d) Minimize delays
  e) Any combination of above

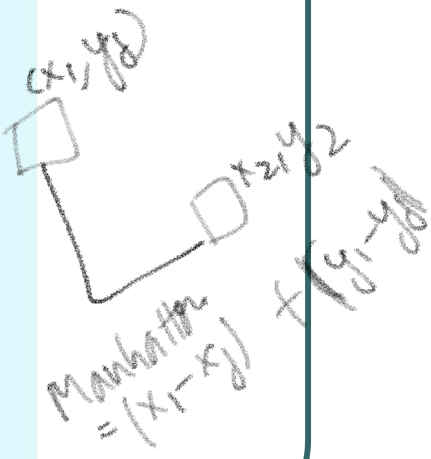
Minimise the wire length to min the delay (gate length are already determined)

# Cost of FP

- **How to determine area?**
  - Not difficult.
  - Can be easily estimated because the dimensions of eac block is known.
  - Area ***A*** computed for each candidate floorplan.
- **How to determine wire length?**
  - A coarse measure is used.
  - Based on a model where all I/O pins of the blocks are merged and assumed to reside at its center.
  - Overall wiring length $L = \Sigma_{i,j} (c_{ij} * d_{ij})$

    where $c_{ij}$ : connectivity between blocks i and j

    $d_{ij}$ : Manhattan distances between the centres of rectangles of blocks i and j



# Cost of FP

- **Typical cost function used:**

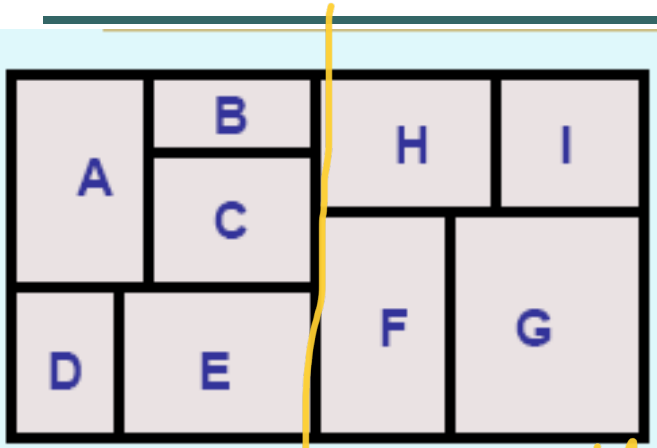
  **Cost = w1 * A + w2 * L**

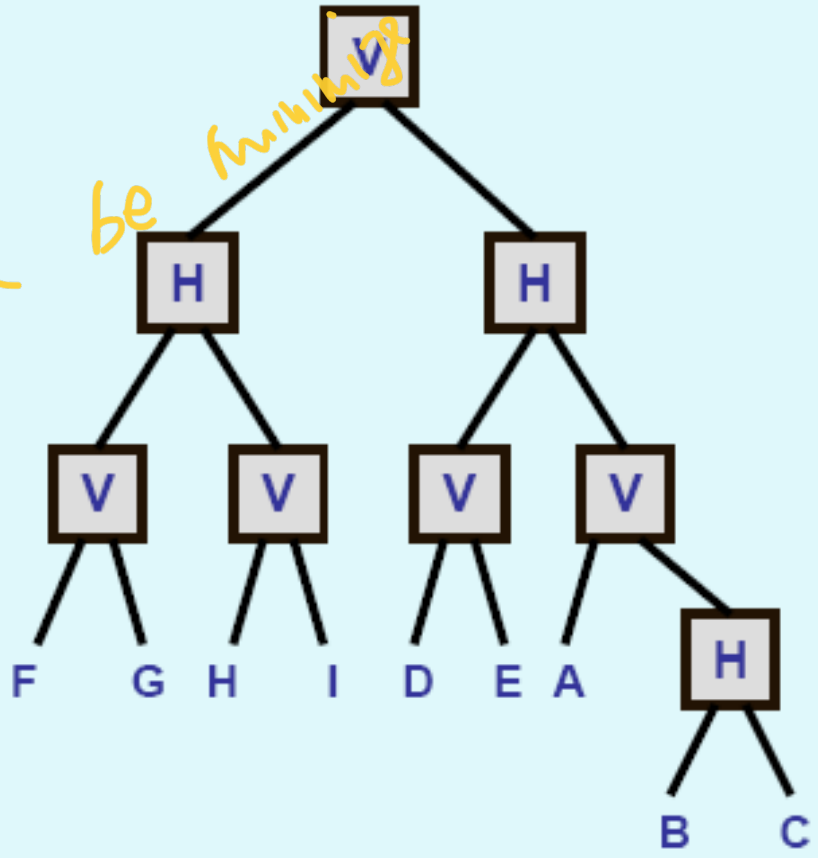
  where w1 and w2 are user-specified parameters.

# Slicing Structure

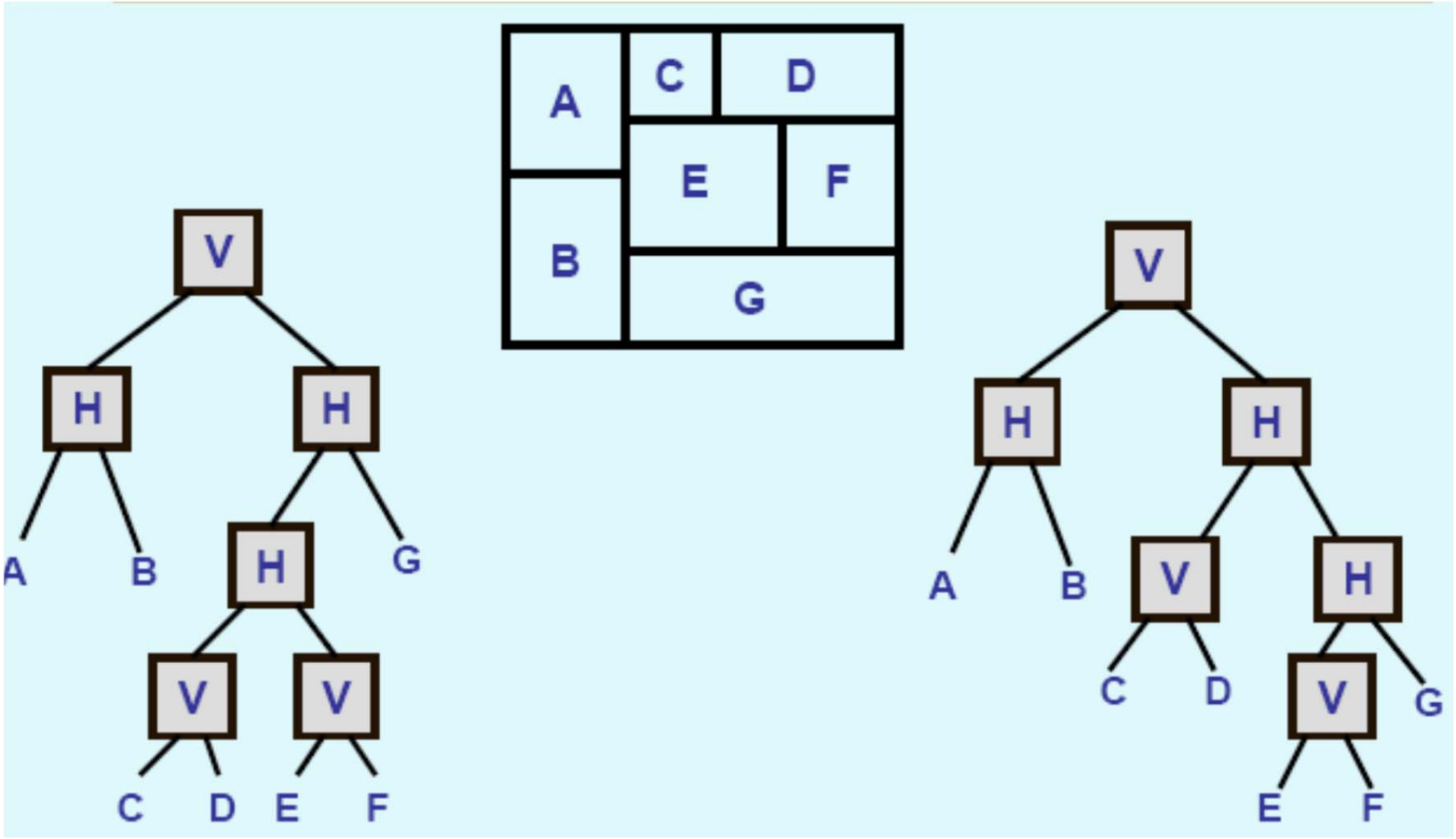
- **Definition**
  - A rectangular dissection that can be obtained by repeatedly splitting rectangles by horizontal and vertical lines into smaller rectangles.
- **Slicing Tree**
  - A binary tree that models a slicing structure.
  - Each node represents a *vertical cut line* (V), or a *horizontal cut line* (H).
    - A third kind of node called *Wheel (W)* appears for non-sliceable floorplans (discussed later).
  - Each leaf is a basic block (rectangle).

# Slicing Structure



# Slicing Tree is not Unique



# FP Algorithms

- **Several broad classes of algorithms:**
  - **Integer programming based**
  - **Rectangular dual graph based**
  - **Hierarchical tree based**
  - **Simulated annealing based**
  - **Other variations**

# ILP Formulation

- The problem is modeled as a set of linear equations using 0/1 integer variables.
- Given:
  - Set of *n* blocks $S = \{B_1, B_2, \ldots, B_n\}$ which are rigid and have fixed orientation.
  - 4-tuple associated with each block

    $(x_i, y_i, w_i, h_i)$