Lecture 8: Floorplanning

ECE201A

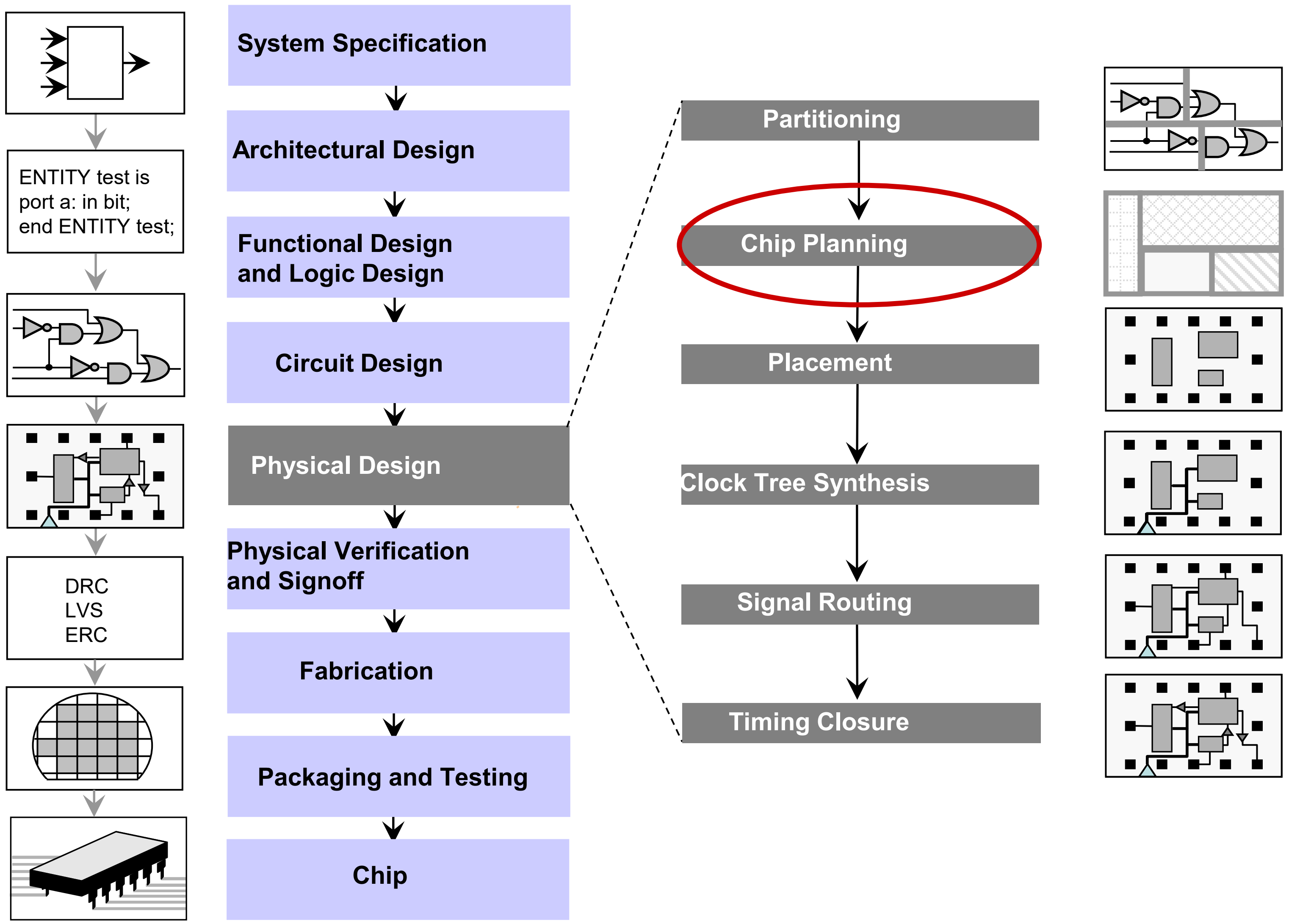
Some notes adopted from
Andrew B. Kahng
Lei He
Igor Markov
Mani Srivastava
Mohammad Tehranipoor

Puneet Gupta (puneet@ee.ucla.edu)

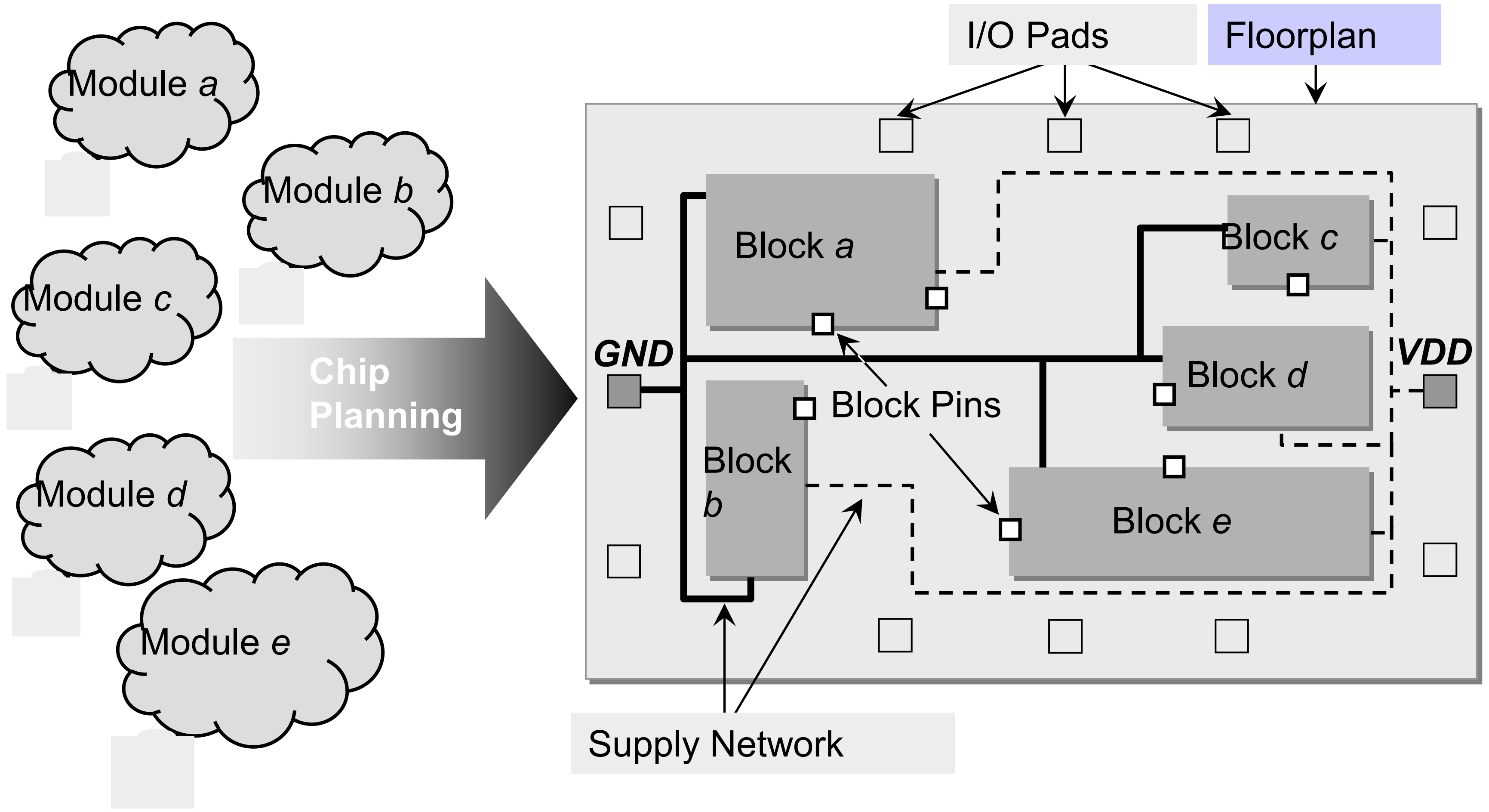
Logistics

- Some guidelines about grading in ECE201A
  - 80%+ : A
  - 70%-80% : A-
  - 65% - 70%: B+
  - 60% - 65%: B
  - 50%-60%: B-
  - You really shouldn't be getting below 50%!
- Remember Quiz 1 is next week on Gradescope in class.
- Please post your questions on Piazza regarding anything about lectures or labs or project
  - Everyone can benefit from responses and easier for us..

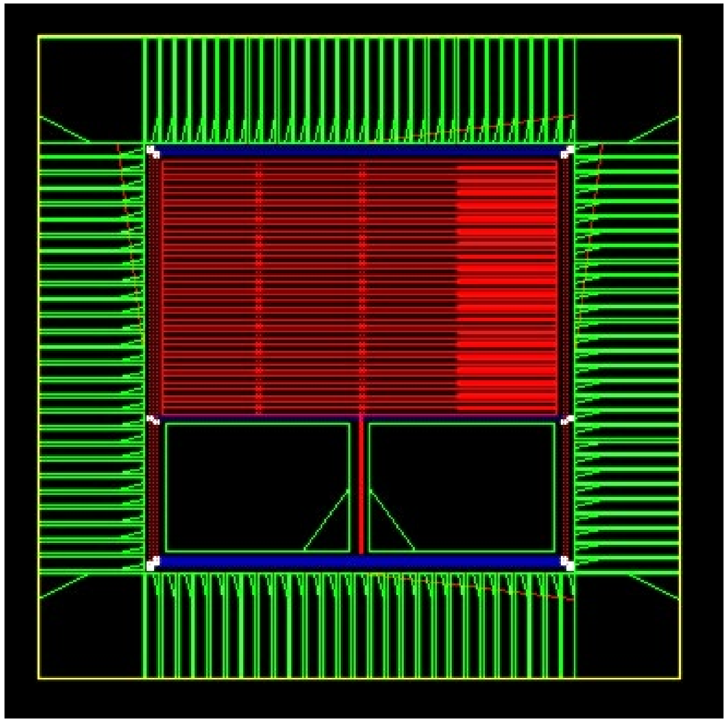
Introduction



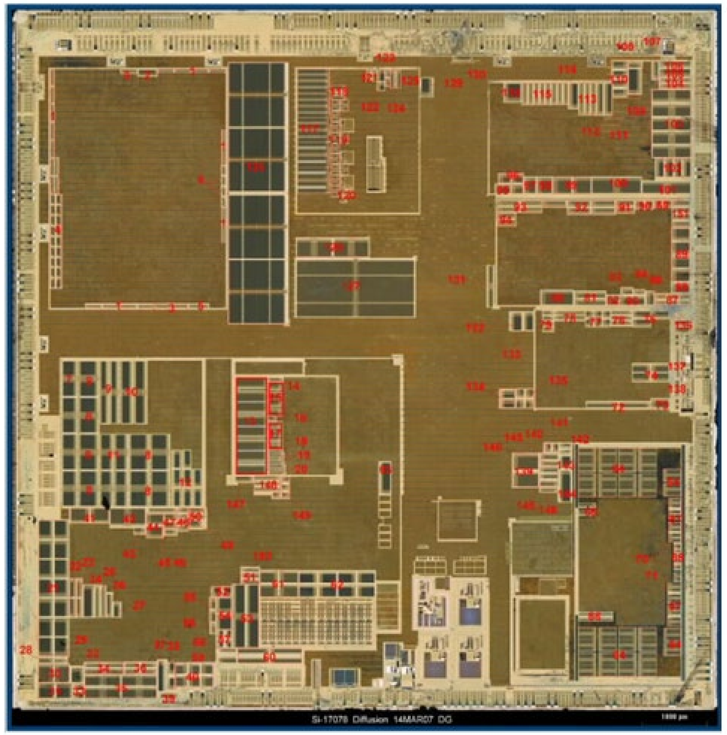
Introduction

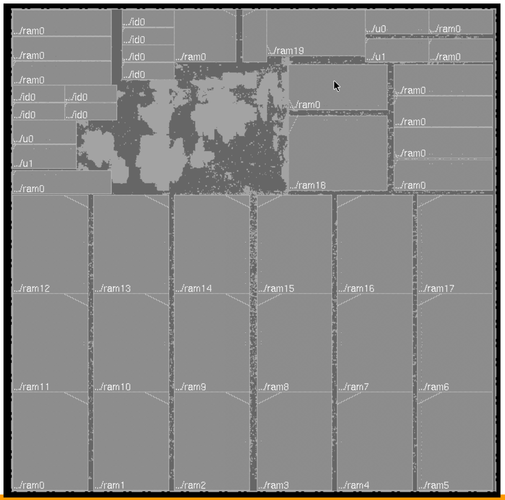


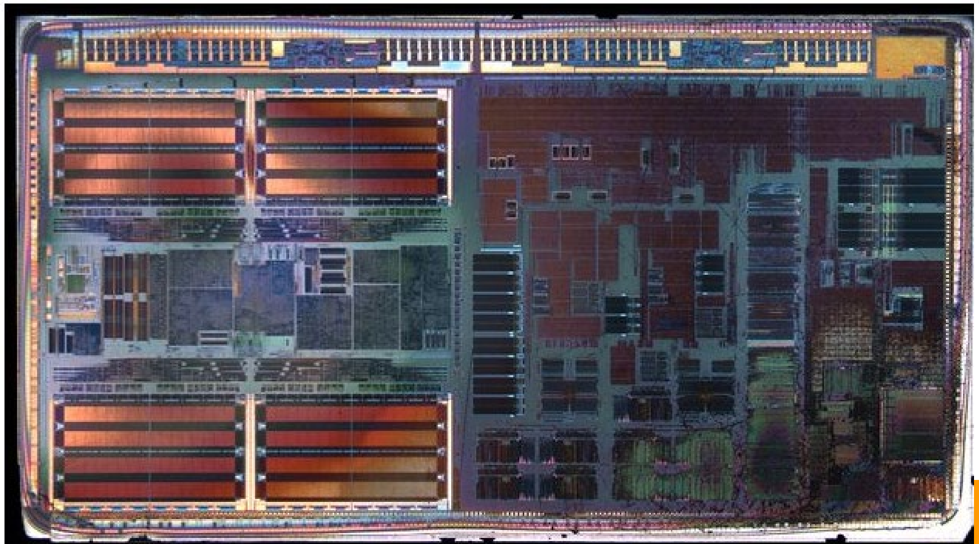
Floorplan Picture



Modern SoC: many memories, heavy power network



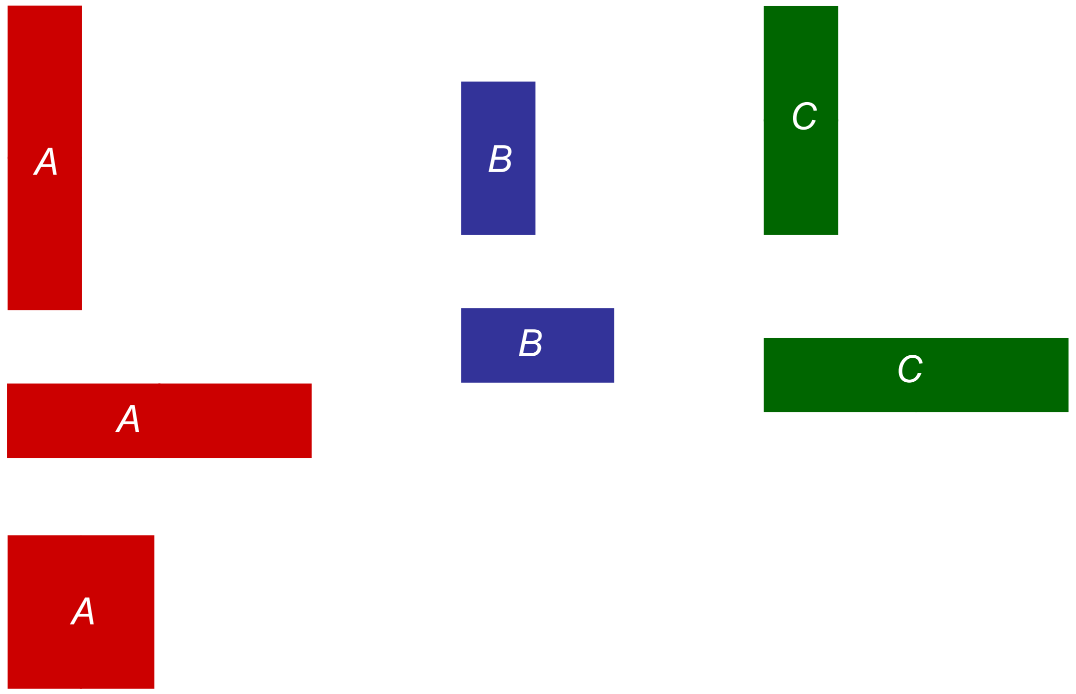




Example

Given: Three blocks with the following potential widths and heights
Block *A*: $w = 1, h = 4$ or $w = 4, h = 1$ or $w = 2, h = 2$
Block *B*: $w = 1, h = 2$ or $w = 2, h = 1$
Block *C*: $w = 1, h = 3$ or $w = 3, h = 1$

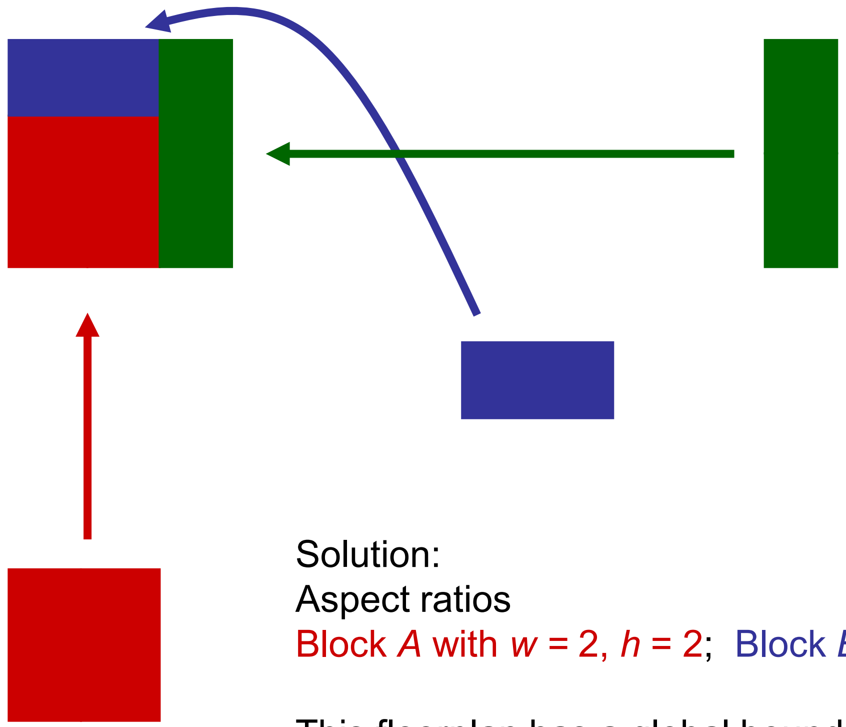
Task: Floorplan with minimum total area enclosed



Example

Given: Three blocks with the following potential widths and heights
Block *A*: *w* = 1, *h* = 4 or *w* = 4, *h* = 1 or *w* = 2, *h* = 2
Block *B*: *w* = 1, *h* = 2 or *w* = 2, *h* = 1
Block *C*: *w* = 1, *h* = 3 or *w* = 3, *h* = 1

Task: Floorplan with minimum total area enclosed

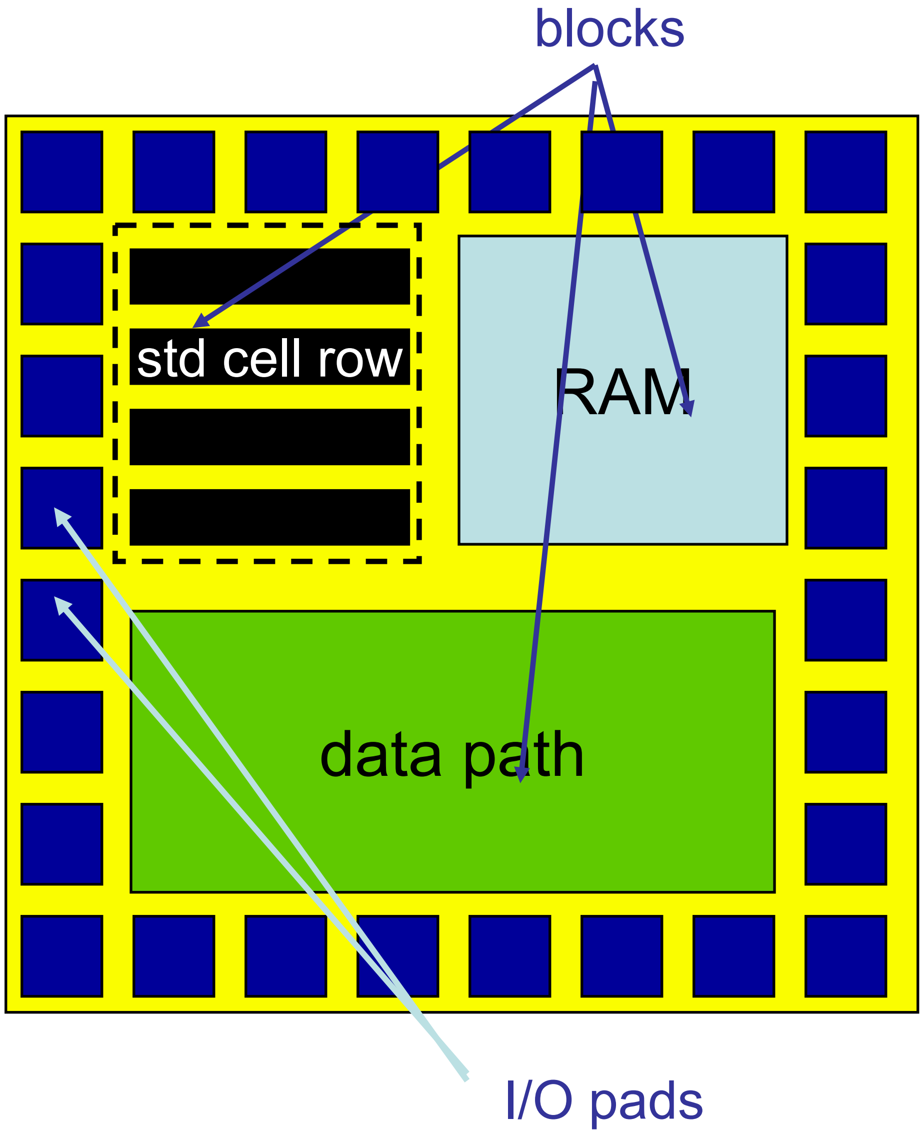


Solution:
Aspect ratios
Block *A* with *w* = 2, *h* = 2; Block *B* with *w* = 2, *h* = 1; Block *C* with *w* = 1, *h* = 3

This floorplan has a global bounding box with minimum possible area (9 square units).

Blocks

- **Blocks** are inside a pad frame
  - –**Hard** = defined outline = fixed (H,W)
  - –**Soft** = defined area, but (H,W) flexible
  - –**Semi-soft** = discrete set of (H,W) pairs
  - –Shapes: rectangular, L, T, rectilinear
  - –Pin locations defined
  - –Can rotate, mirror
- Routing inside, between blocks
  - –Sometimes, over blocks
- Floorplanning of different-sized blocks is harder than place and route of standard cells
  - –Block placement is done by hand
  - –Issues:
    - access to power supply (power-hungry blocks)
    - alignment of power grid to supply pins
    - soft blockage / "halo" to ensure routability
    - leave contiguous region for std-cell P&R
    - buffer sites for nets that want to get around a macro
    - data flow



Courtesy K. Yang, UCLA

Automated Floorplanning

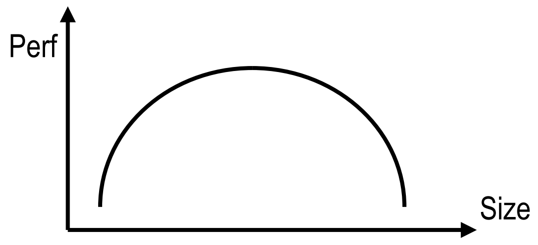
- Area and shape of the global bounding box
  - Minimizing the area involves finding (*x*,*y*) locations, as well as shapes, of the individual blocks.
- Total wirelength
  - Long connections between blocks may increase signal propagation delays in the design.
  - May make chip-level routing difficult
- Power integrity
  - Minimize IR drop → enough power to the blocks
- No automated floorplanning tool has ever made it to "prime time"
  - How should a floorplan be represented ?
    - Completeness: should be able to represent all possible floorplans
    - Efficiency: conversion between representation and actual realization
    - Redundancy: not good to evaluate two floorplans, only to find they are same
  - How do we search over the space of feasible floorplan representations?
    - Often, simulated annealing is used

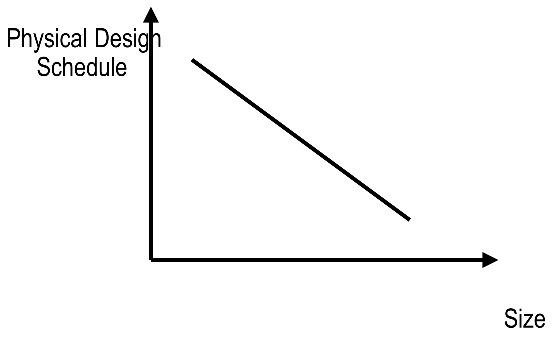
Power Grid Definition During Floorplanning

- Which layers are the primary mesh (= thick metal, e.g., M7, M8)
- What is the width and pitch of the power rails?
  - Depends on peak current draws (e.g., "1 um width per mA")
- How frequently to tap down from primary mesh to M1 rails
- What is the width and pitch of power rings?
- Choose power routing widths and pitch of via stacks carefully to avoid blocking extra routing tracks
  - Easy to make the design unroutable
  - If a track is blocked, then use the space…
- As soon as can get a quick placement, check IR drop before continuing
  - All modes including test mode

Size Estimation in Standard-Cell Blocks

- Why we care:
  - If area is too small: P&R will not finish or meet timing, will run too long
  - Schedule and size inversely related (but, size will win out for high-volume production – and everyone hopes that their chip will be high-volume…)
  - Performance and size have a complex relationship

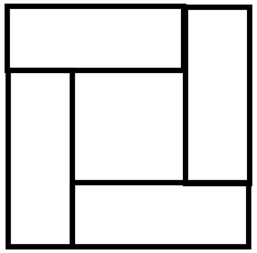




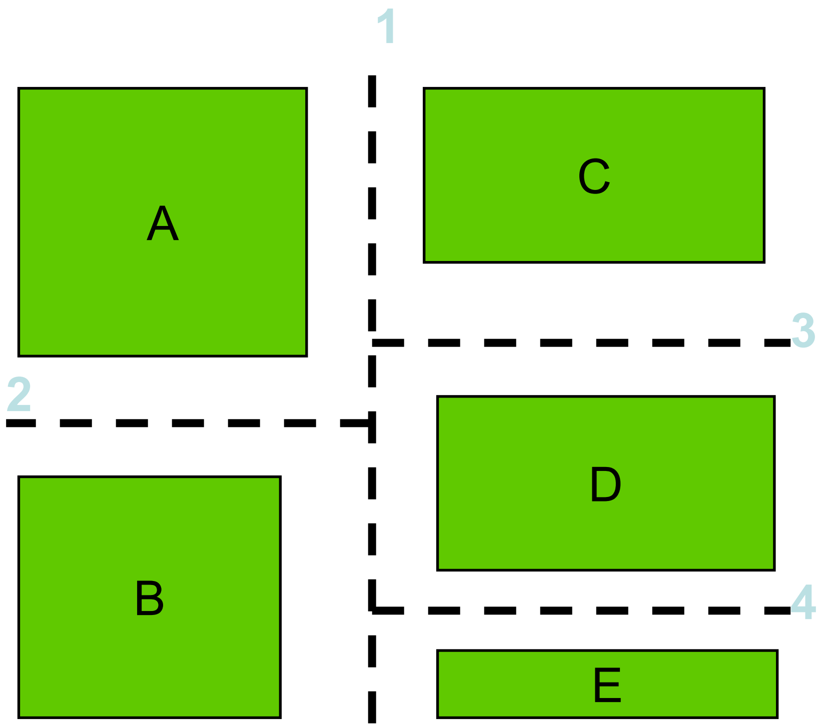
- Old rule of thumb (modulo corrections for power, clock, etc.):
    - 3LM: Cell utilization 65 percent
    - 4LM: Cell utilization 70 percent
    - 5LM: Cell utilization 75 percent
    - 6LM: Cell utilization 80 percent
- Metrics for standard-cell blocks
  - **Low interconnect density → Cell utilization** (std-cell area / std-cell row area)
  - **High interconnect density → Pin density** (causes routing hotspots);

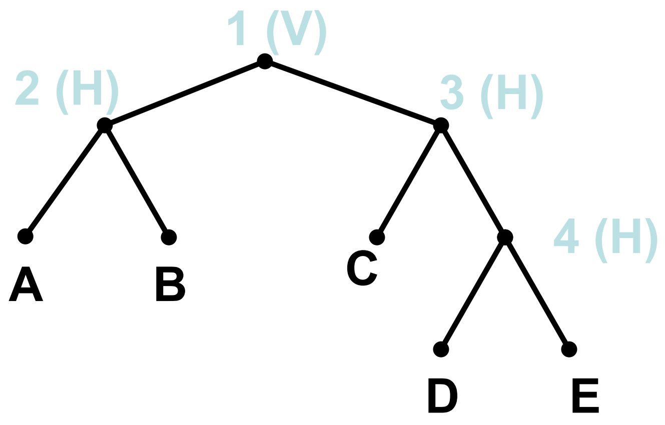
Slicing Floorplan Representation

- A *slicing* floorplan can be recursively cut in two without cutting any blocks
- A "wheel" is an example of a non-slicing floorplan



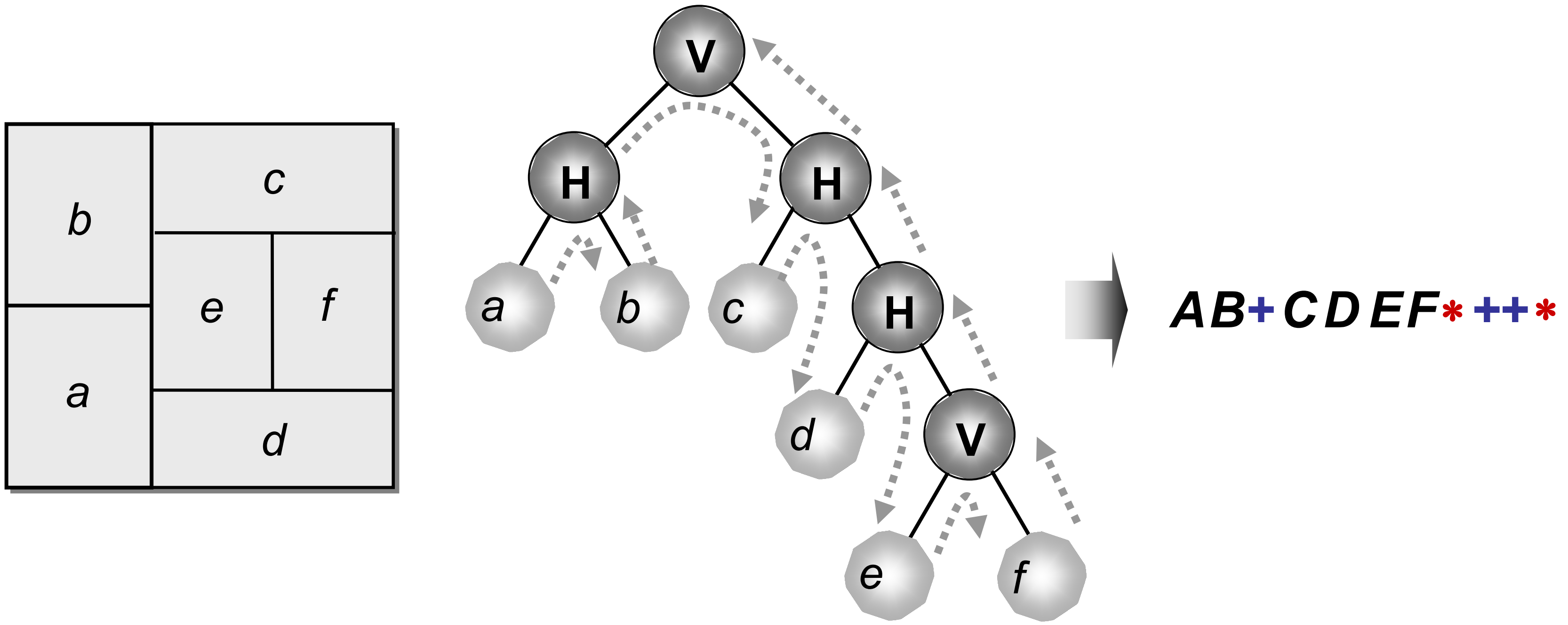
- A slicing floorplan can be represented as a binary tree, with internal nodes representing slices in the floorplan and leaves representing blocks.
  - Polish Expression (PE): Post-order (Left, right, root) listing of nodes in depth-first traversal of binary tree: ABHCDEHHV
  - For given slicing floorplan, PE not unique → some redundancy





Polish Expressions for Slicing Tree

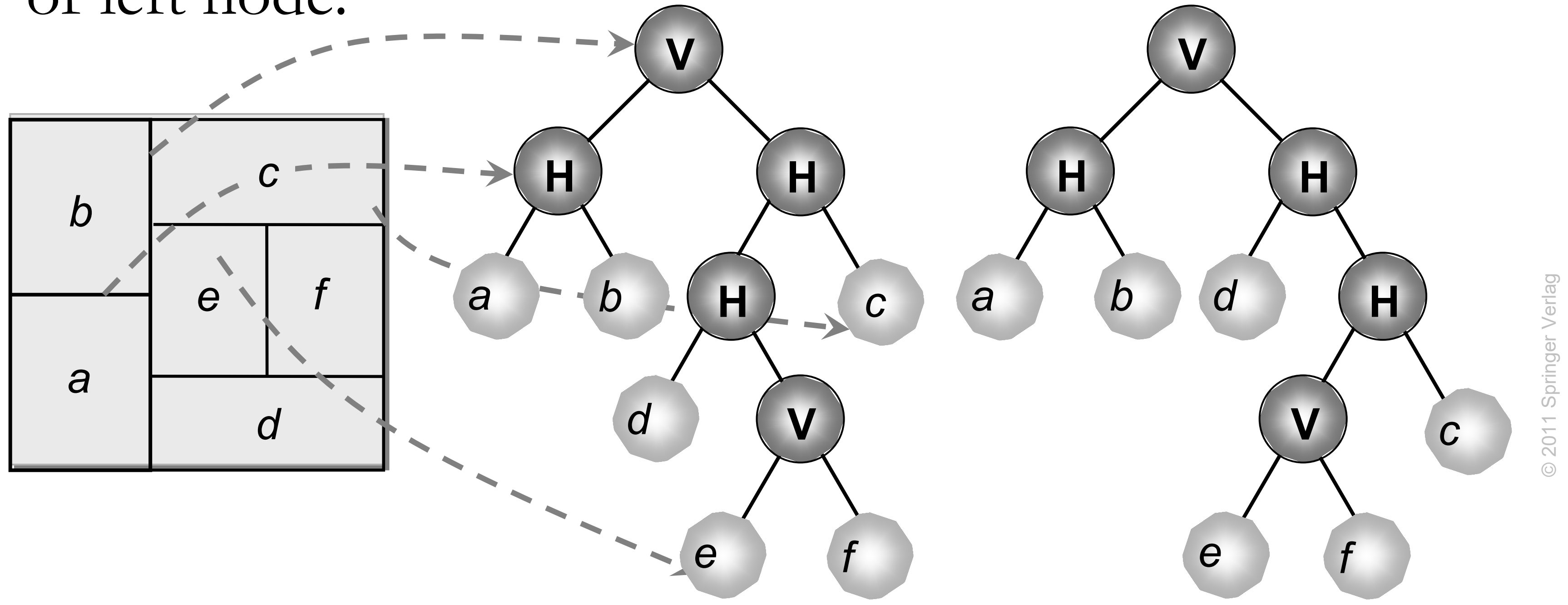
- Post-order representation of the graph



- Bottom up: **V** → * and **H** → +
- Length 2*n*-1 (*n* = Number of leaves of the slicing tree)

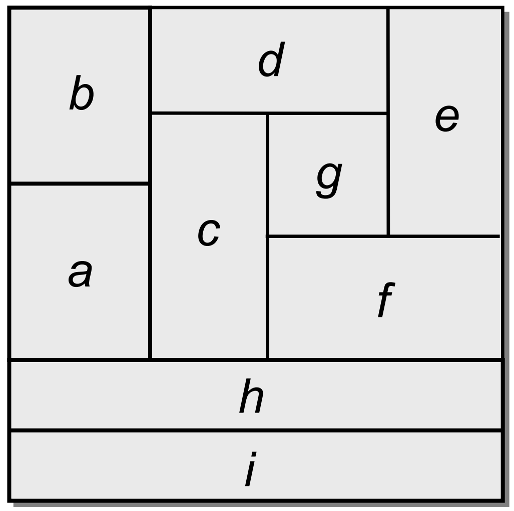
Slicing Tree

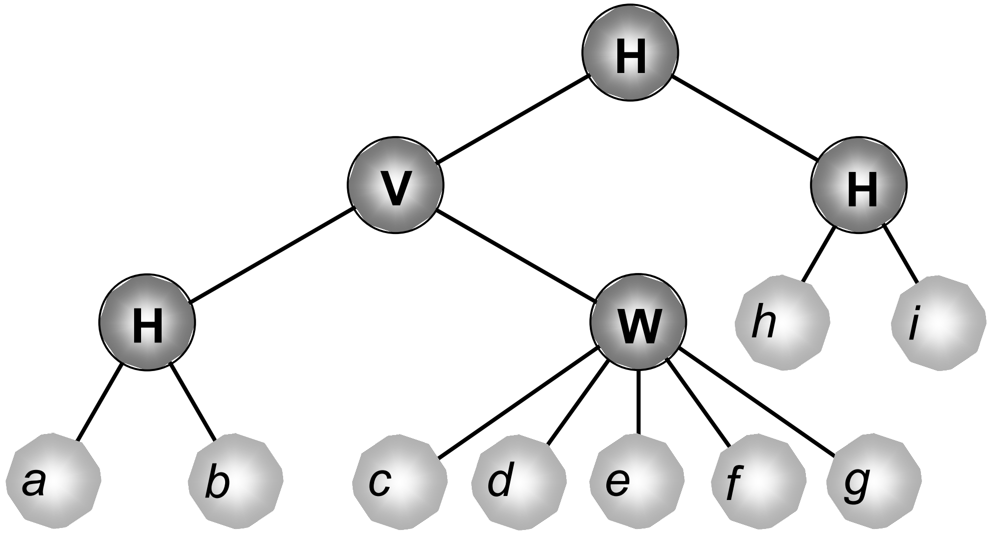
Slicing floorplan and two possible corresponding slicing trees. Remember convention is that left child is bottom or left node.

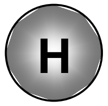


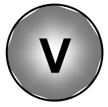
Floorplan Tree with Wheels

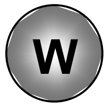
Floorplan tree: Tree that represents a hierarchical floorplan





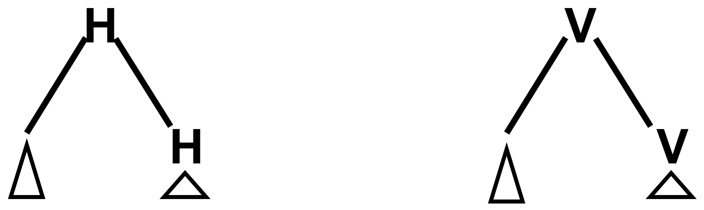

Horizontal division
(objects to the top and bottom)


Vertical division
(objects to the left and right)


Wheel (4 objects cycled around a center object)

Normalized Polish Expressions

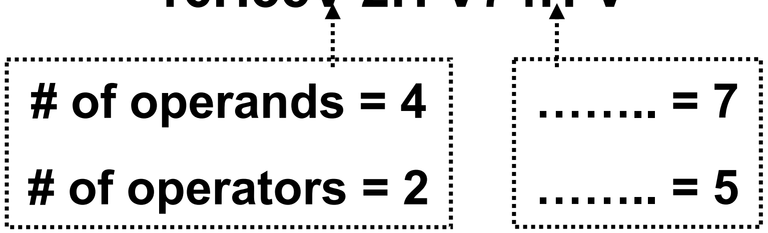
Normalization: do not allow following slicing trees



••• HH •••

••• VV•••

16H35V 2H V74H V



# of operands = 4

# of operators = 2

…….. = 7

…….. = 5

- #operands > #operators for any subexpression
- Total length =2n-1
- Permutation of { 1, 2, …, n} and # of operators =n-1
- No consecutive operators of the same type (due to normalization)

Simulated Annealing

- Generate an initial solution and evaluate its cost
- Generate a new solution by performing a random move
- Solution is accepted or rejected based on a temperature parameter T
- Higher T indicates higher probability to accept a solution with higher cost
- T slowly decreases to form the finalized solution.

- Boltzmann acceptance criterion:

$$e^{-(cost(next_{sol})-cost(curr_{sol}))/T}$$

$curr_{sol}$: current solution

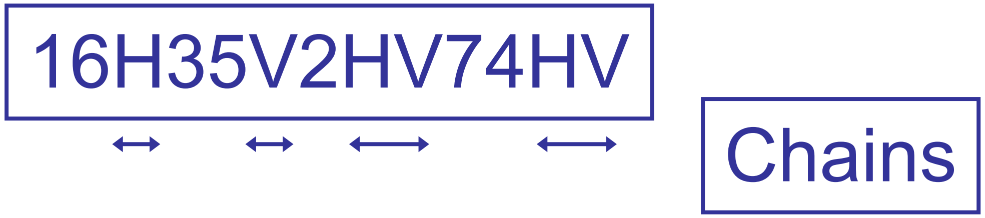
$next_{sol}$: new solution after perturbation

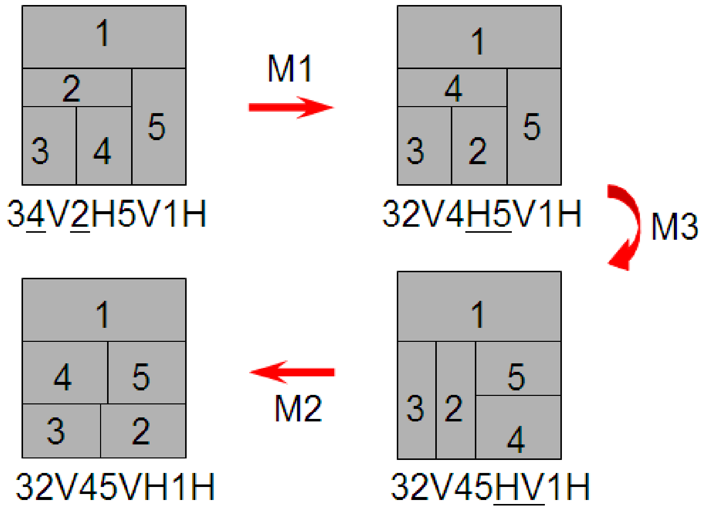
T: current temperature

r: random number between[0,1) from normal distr.

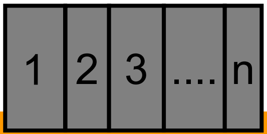
Annealing of *Slicing* Floorplans

- Chain: HVHVH.... or VHVHV....





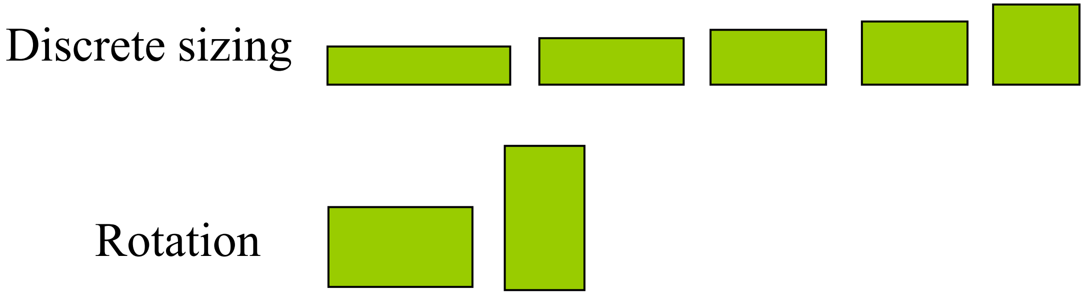
- Neighborhood operators ("moves")
  M1: Swap adjacent operands (ignoring chains)
  M2: Complement (H→V; V→H) some chain
  M3: Swap an adjacent operand and operator
  (can give an invalid NPE, so must check validity of this move)
- Fact: every pair of valid NPE's is connected by some move sequence → "reachability" within neighborhood structure
  – Initial SA solution: 12V3V…nV

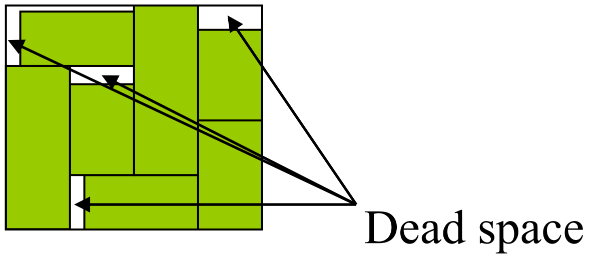


Adapted from D. Pan, EE382V Fall 2008, UT Austin

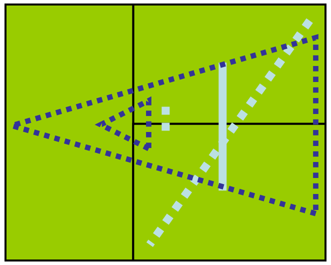
Estimating Cost

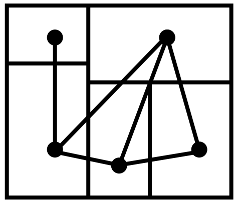
- Floorplanning is difficult for at least two reasons
  - Blocks have bounded or discrete aspect ratio (AR) = max (H/W, W/H)
  - Non-overlapping constraint: minimum area = minimum "dead space"





- Classical objective function: C = α·Area + β·Wirelength
  - **Issue:** How to estimate WL when pin locations are not known, blockages not comprehended, etc.
    - 2-pin net
    - 3-pin net

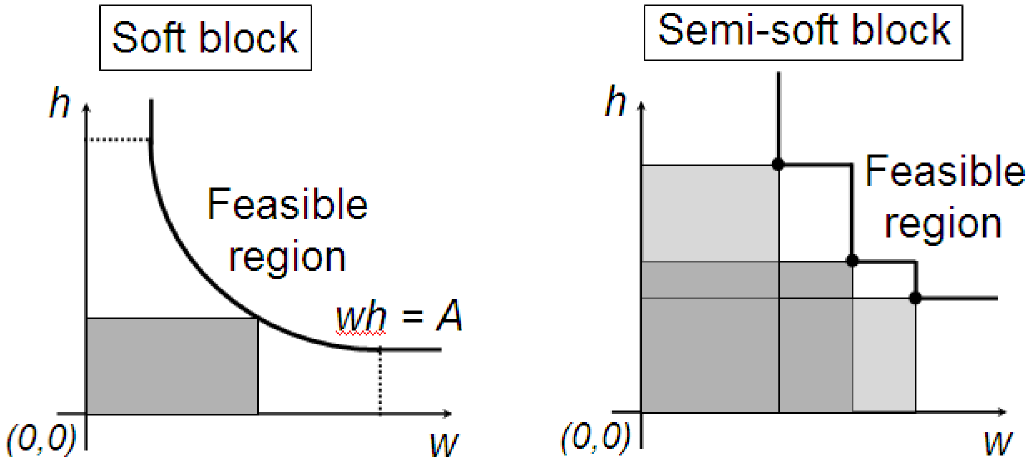


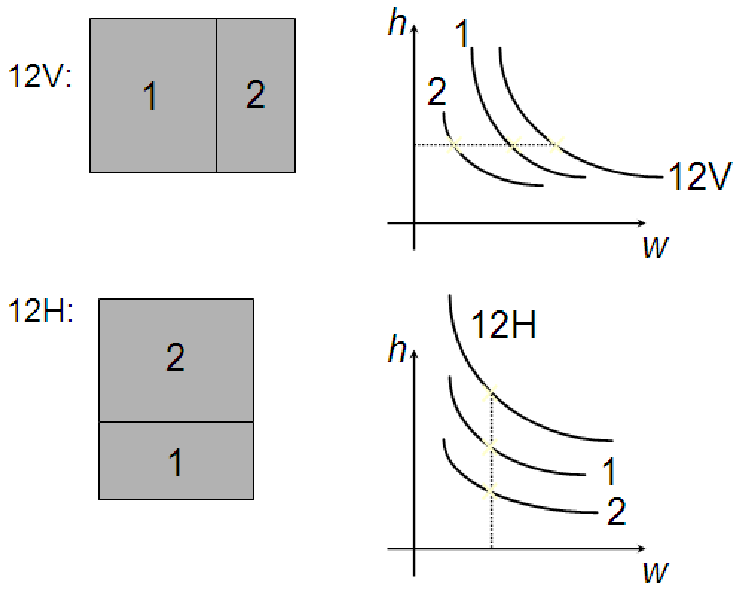


Adapted from D. Pan, EE382V Fall 2008, UT Austin

Realization of *Slicing* Floorplans

- **What is the implied area of a slicing tree?**
- *Shape function* captures set of feasible (W, H) pairs for each node in slicing tree
- Shape functions can be combined recursively (bottom-up) in slicing tree
  - Pick best-area implementation of root node
  - Maintain k points on each shape curve → O(kn) time to compute shape function of slicing floorplan
  - Can be updated incrementally as well

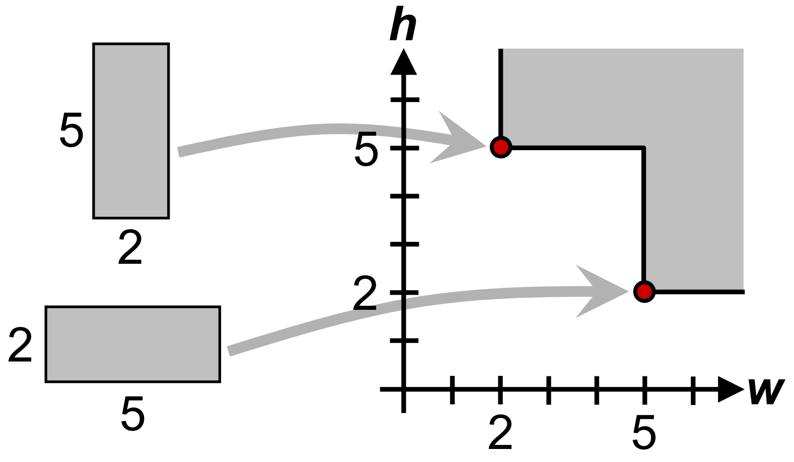




Adapted from D. Pan, EE382V Fall 2008, UT Austin

Shape Function of Hard IP

Corner points



Floorplan Sizing

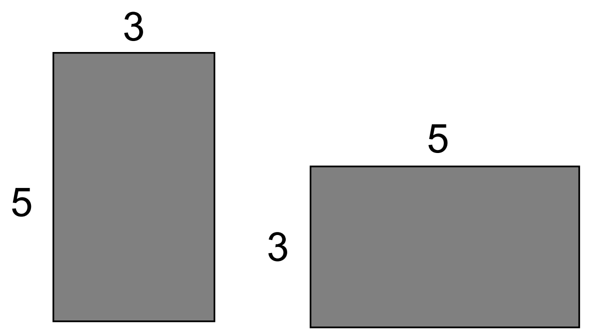
This algorithm finds the **minimum floorplan area** for a given slicing floorplan in polynomial time. For non-slicing floorplans, the problem is NP-hard.

- Construct the shape functions of all individual blocks
- Bottom up: Determine the shape function of the top-level floorplan from the shape functions of the individual blocks
- Top down: From the corner point that corresponds to the minimum top-level floorplan area, trace back to each block's shape function to find that block's dimensions and location.

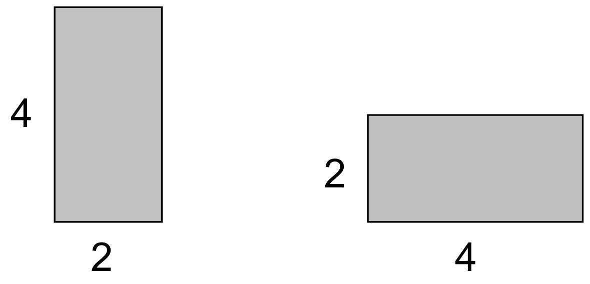
Floorplan Sizing – Example

Step 1: Construct the shape functions of the blocks

Block *A*:



Block *B*:

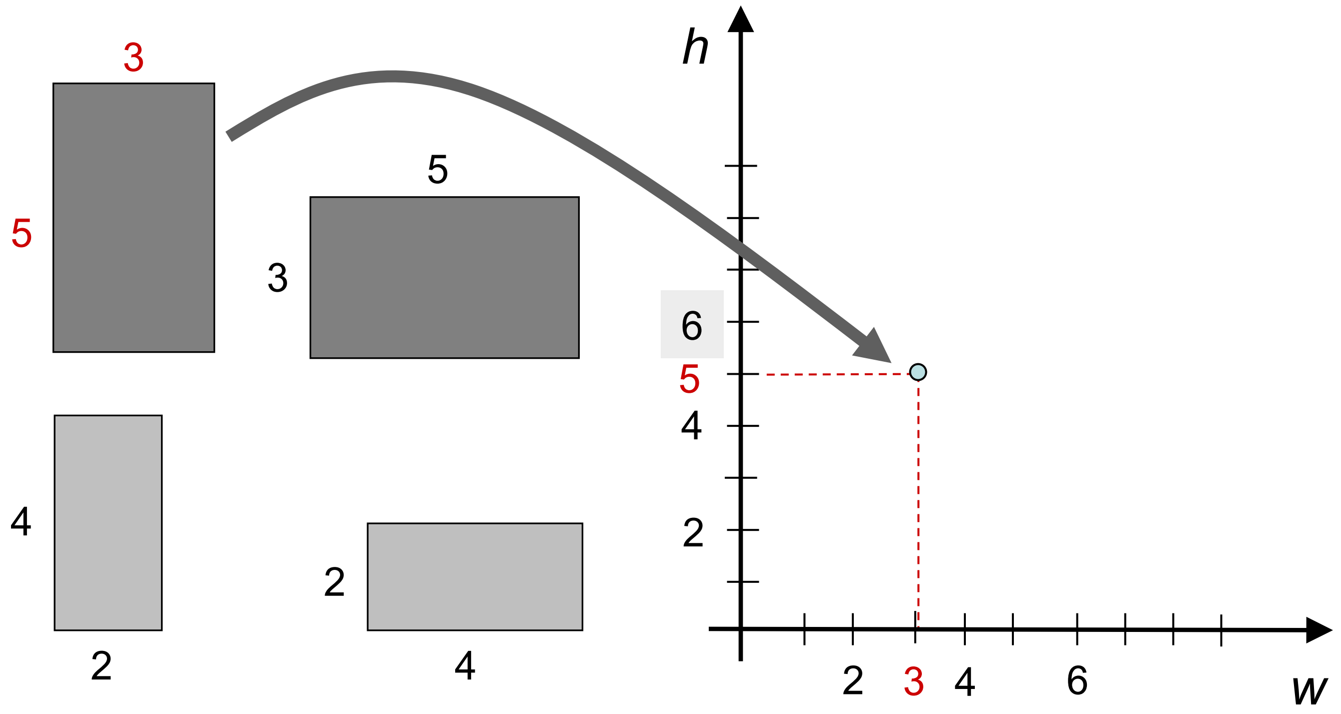


Floorplan Sizing – Example

Step 1: Construct the shape functions of the blocks

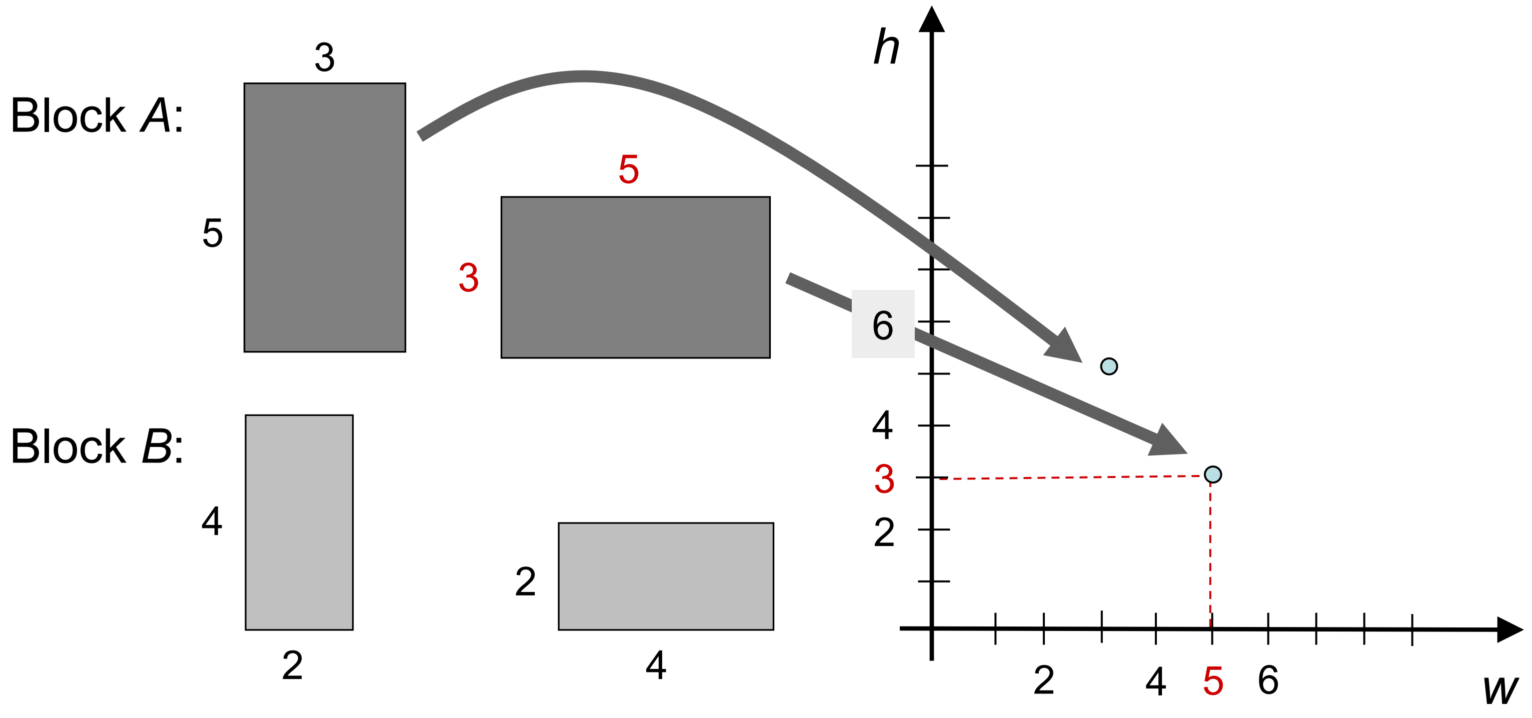
Block *A*:

Block *B*:



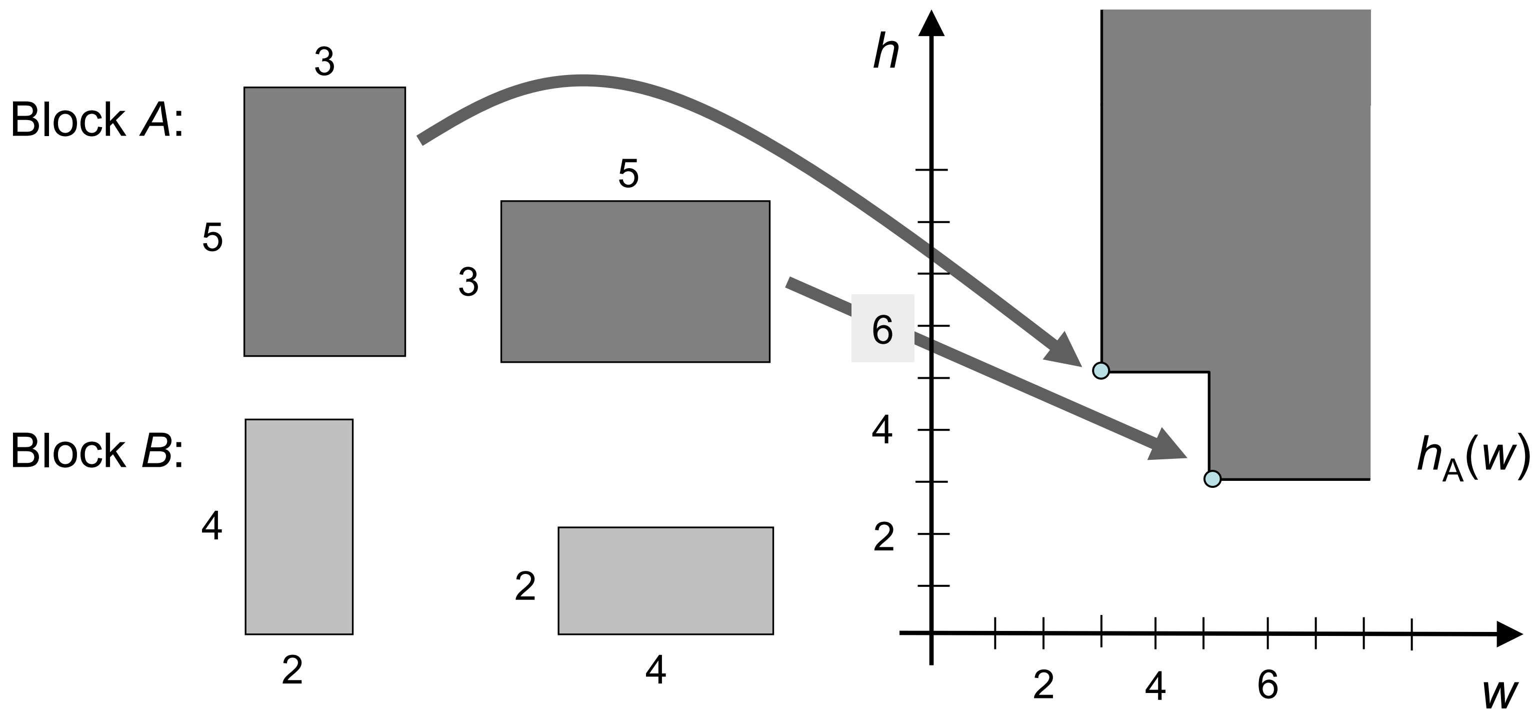
Floorplan Sizing – Example

Step 1: Construct the shape functions of the blocks



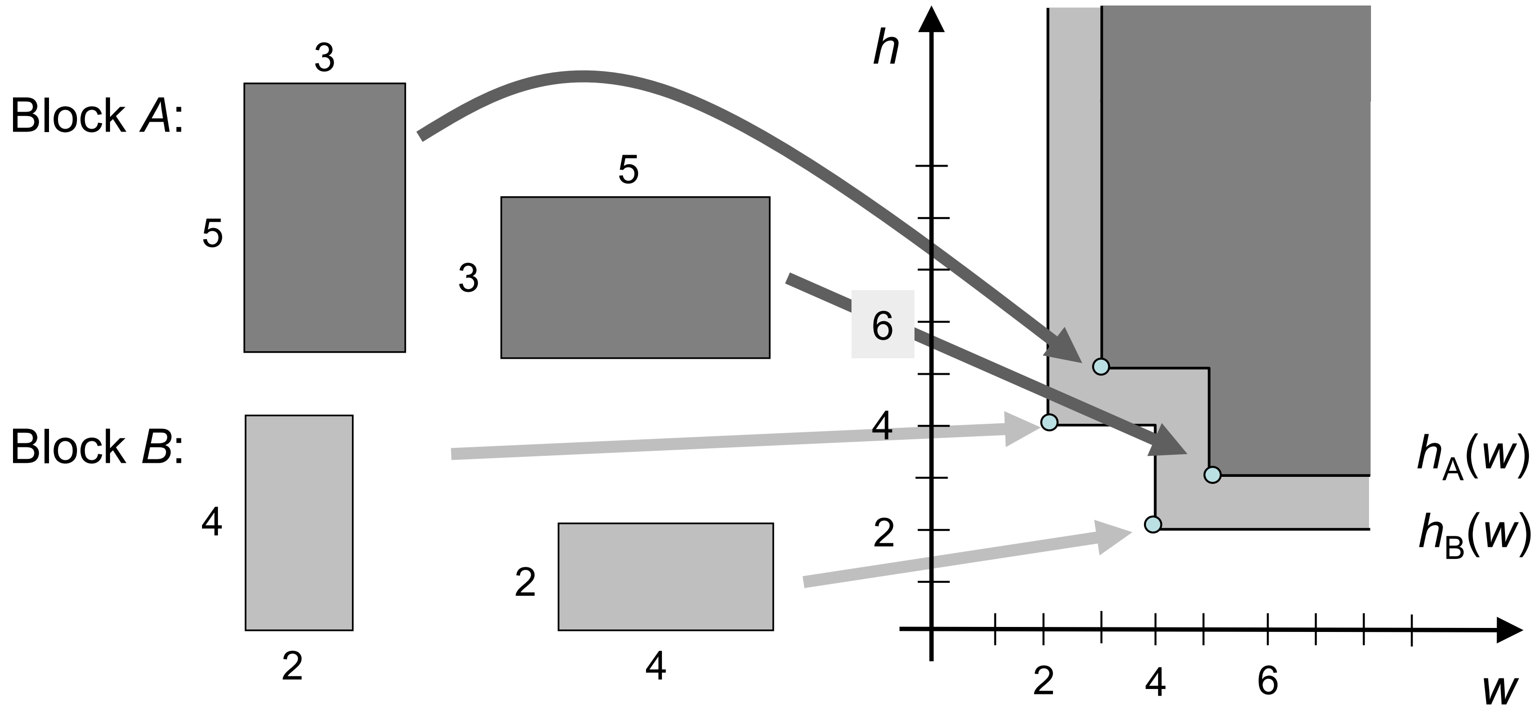
Floorplan Sizing – Example

Step 1: Construct the shape functions of the blocks



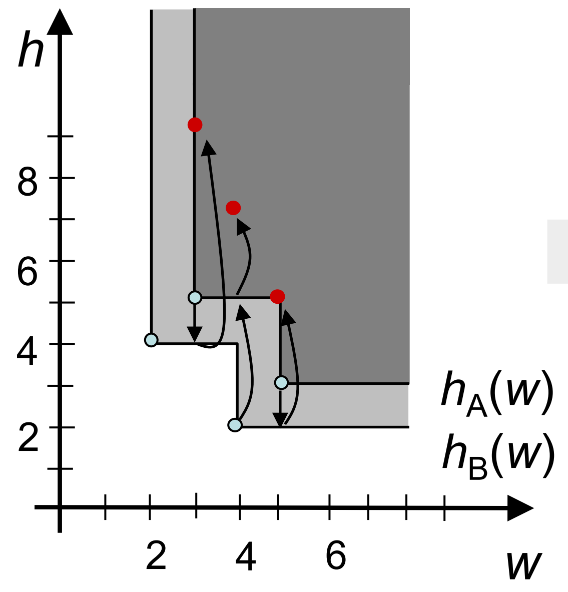
Floorplan Sizing – Example

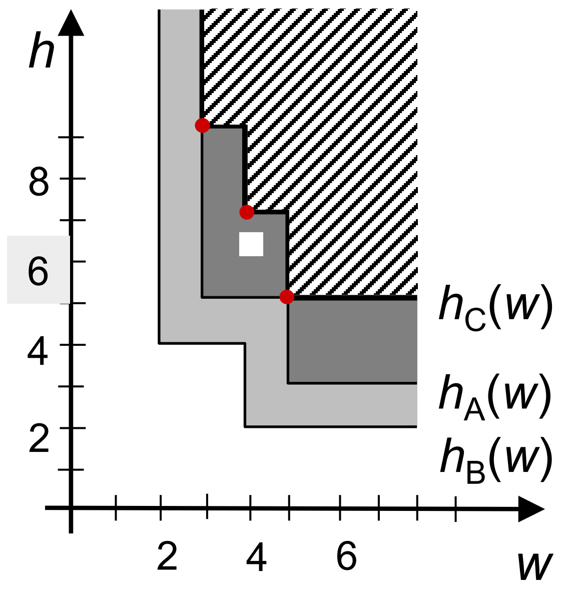
Step 1: Construct the shape functions of the blocks



Floorplan Sizing – Example

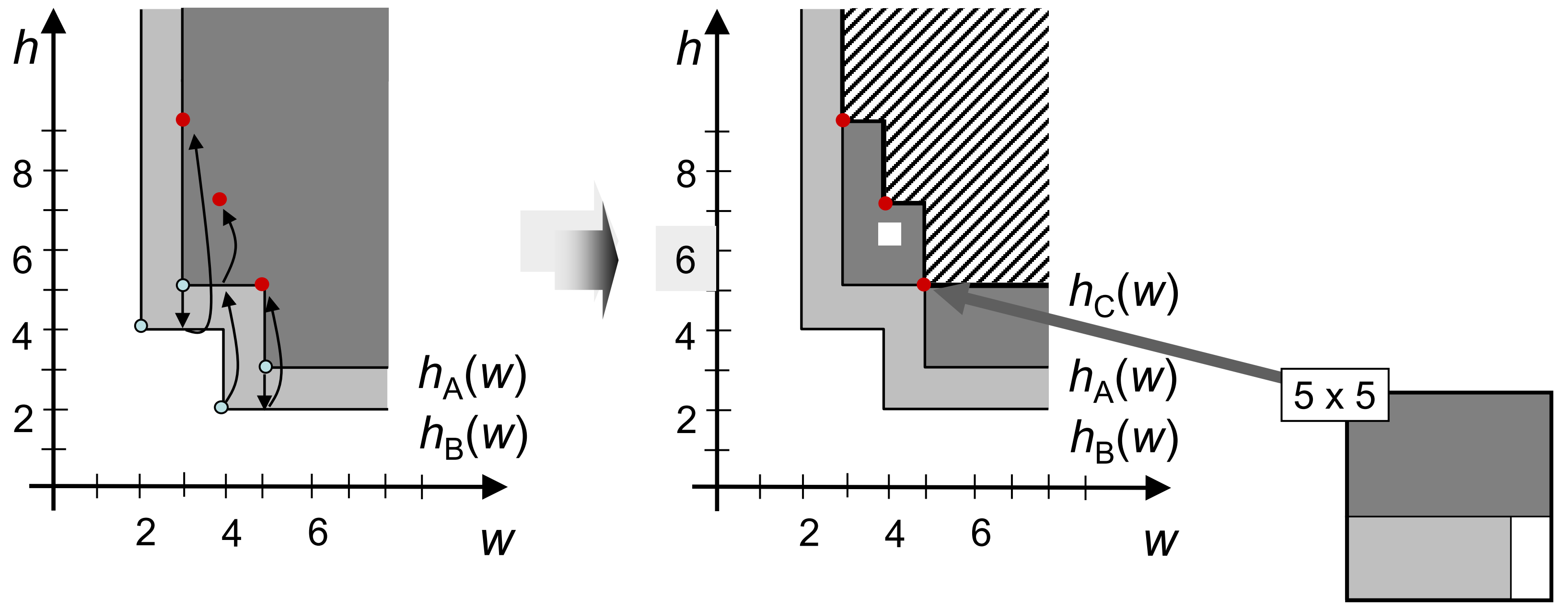
Step 2: Determine the shape function of the top-level floorplan (vertical)





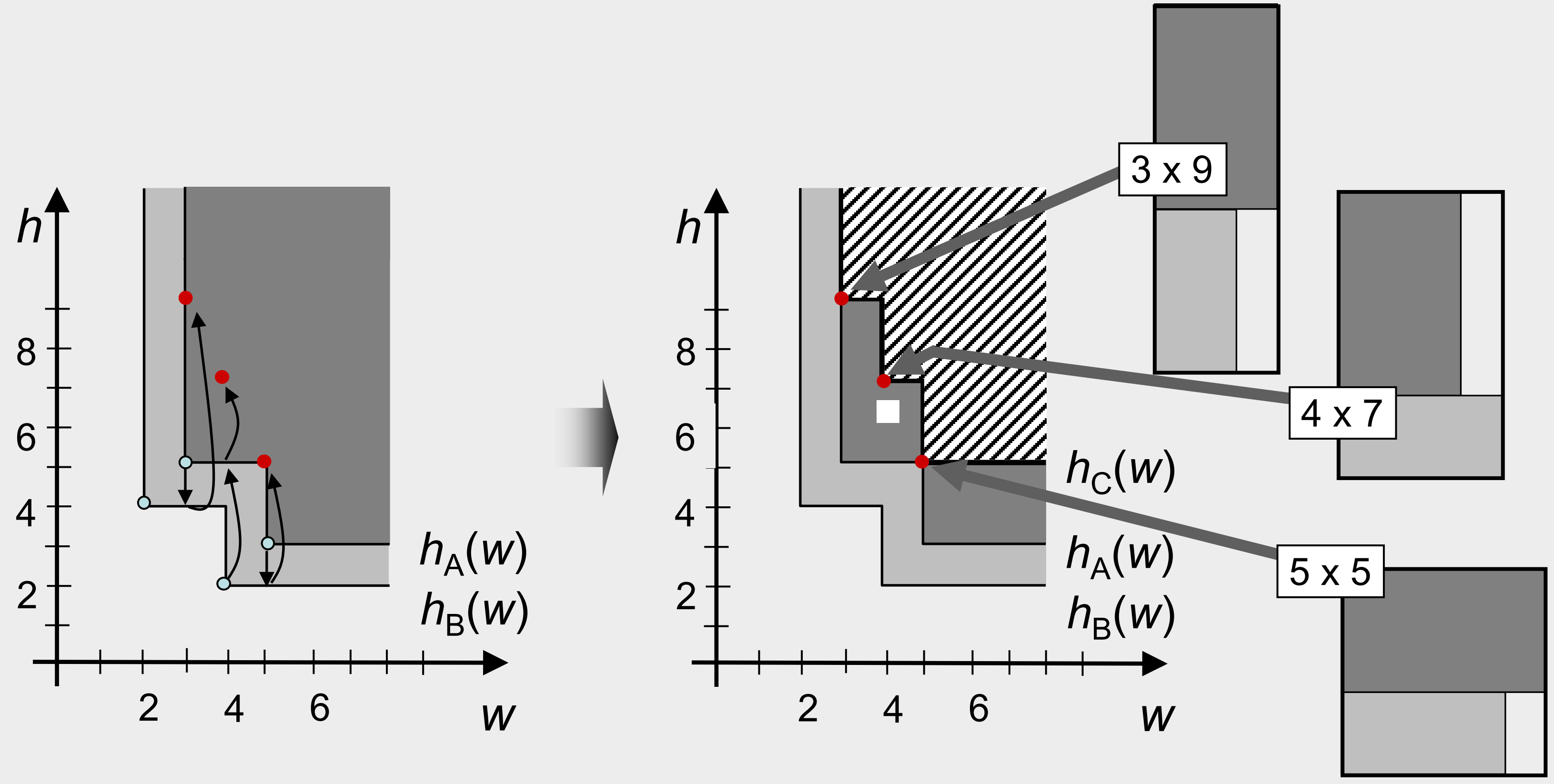
Floorplan Sizing – Example

Step 2: Determine the shape function of the top-level floorplan (vertical)



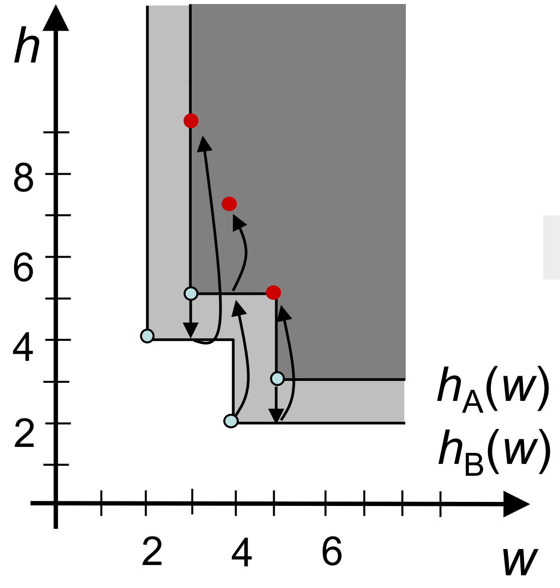
Floorplan Sizing – Example

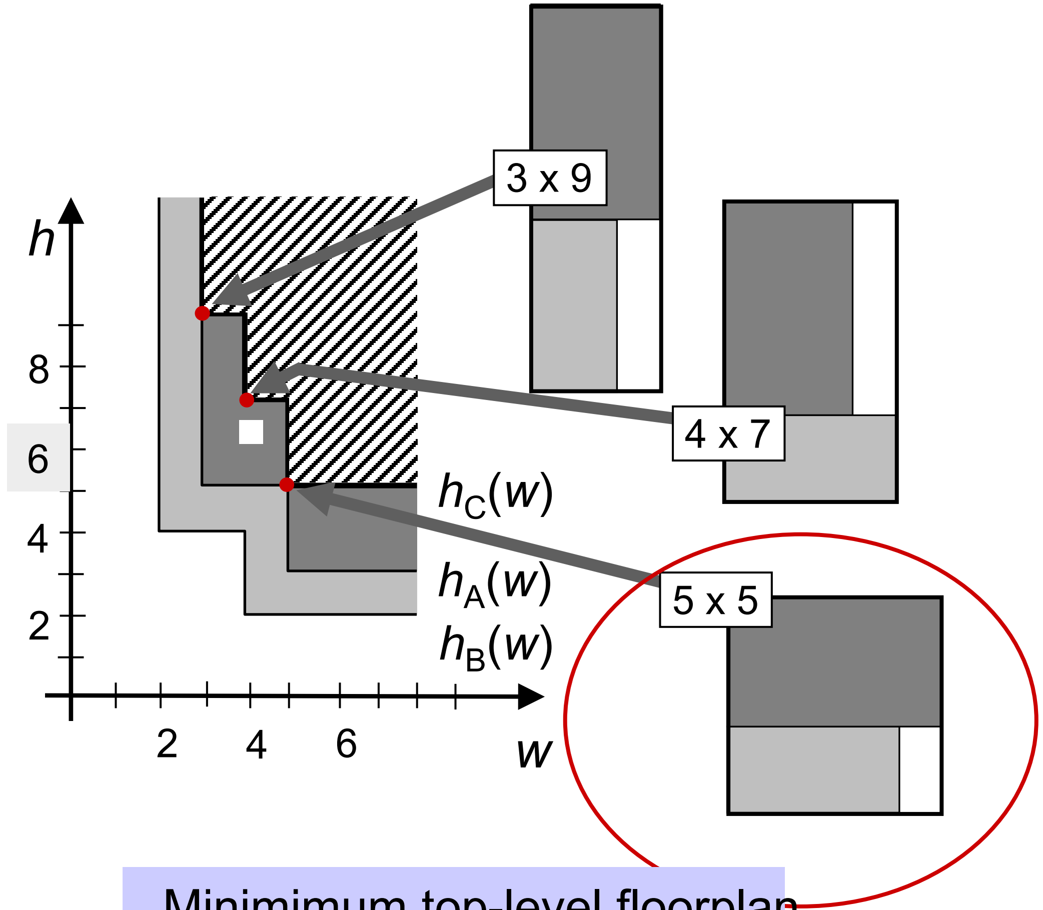
Step 2: Determine the shape function of the top-level floorplan (vertical)



Floorplan Sizing – Example

Step 2: Determine the shape function of the top-level floorplan (vertical)

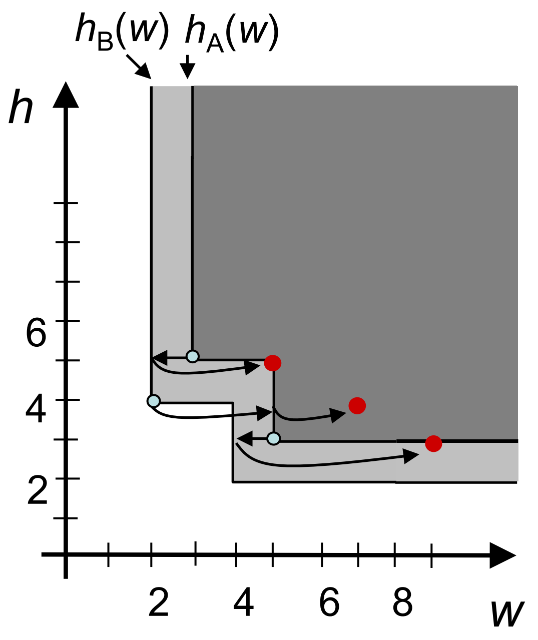


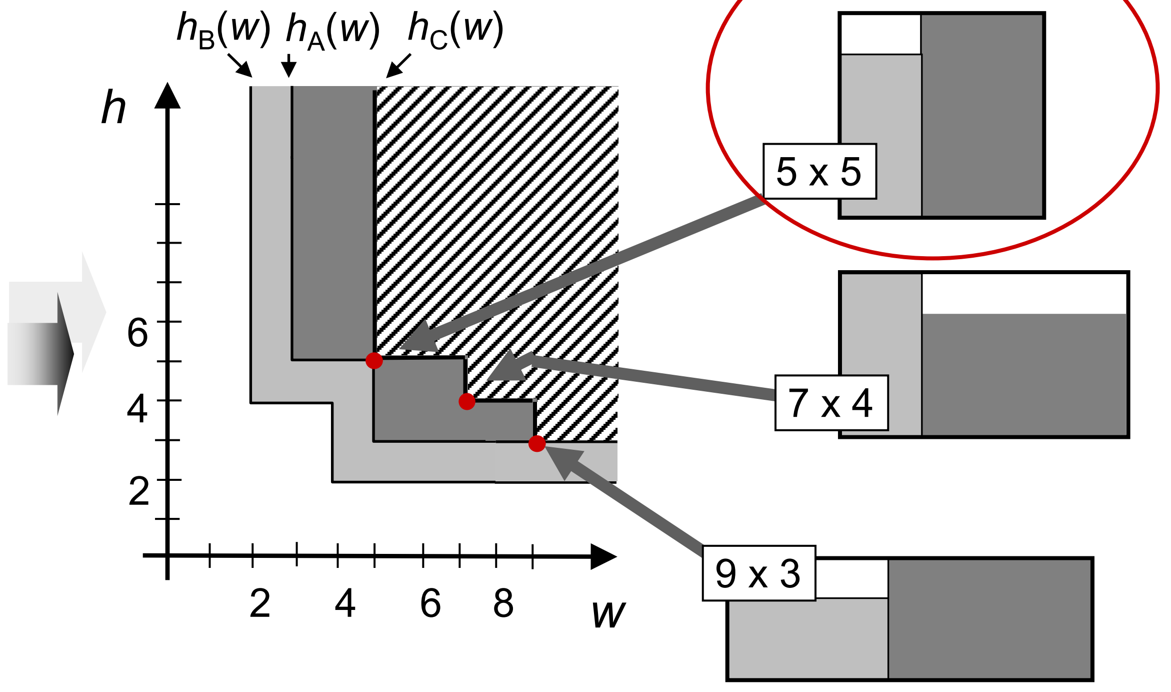


Minimimum top-level floorplan with vertical composition

Floorplan Sizing – Example

Step 2: Determine the shape function of the top-level floorplan (horizontal)

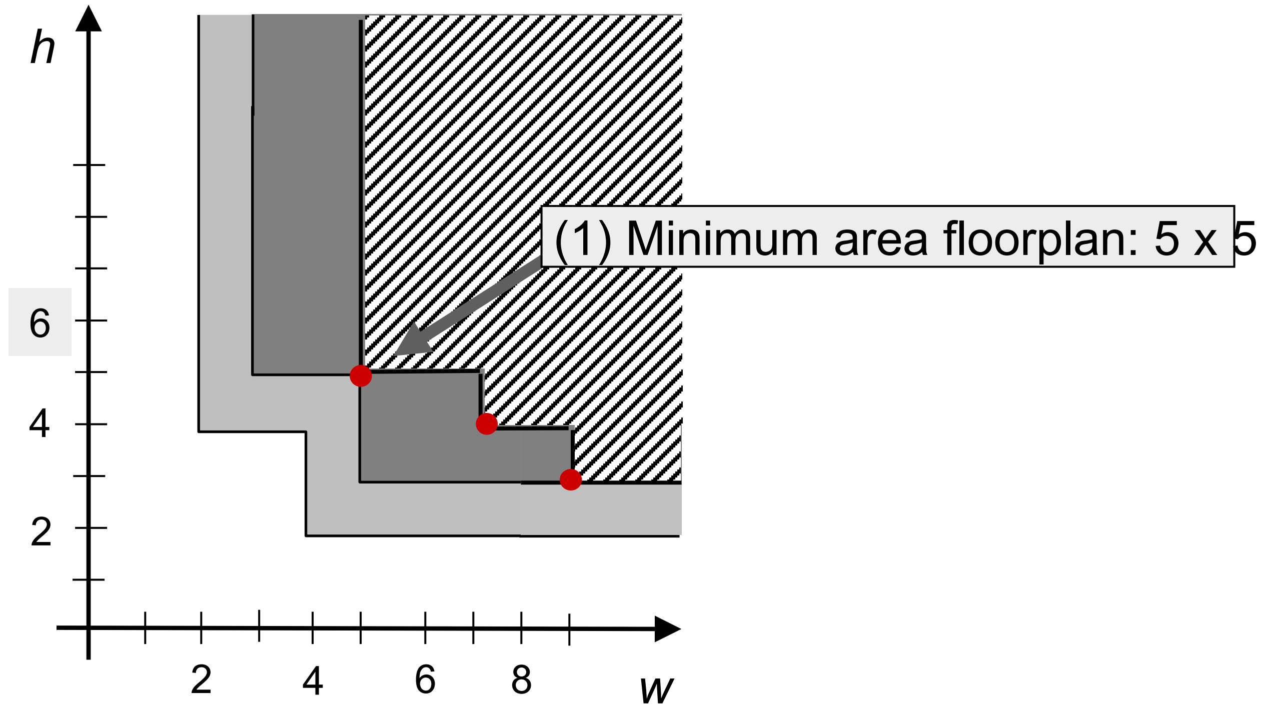




Minimimum top-level floorplan with horizontal composition

Floorplan Sizing – Example

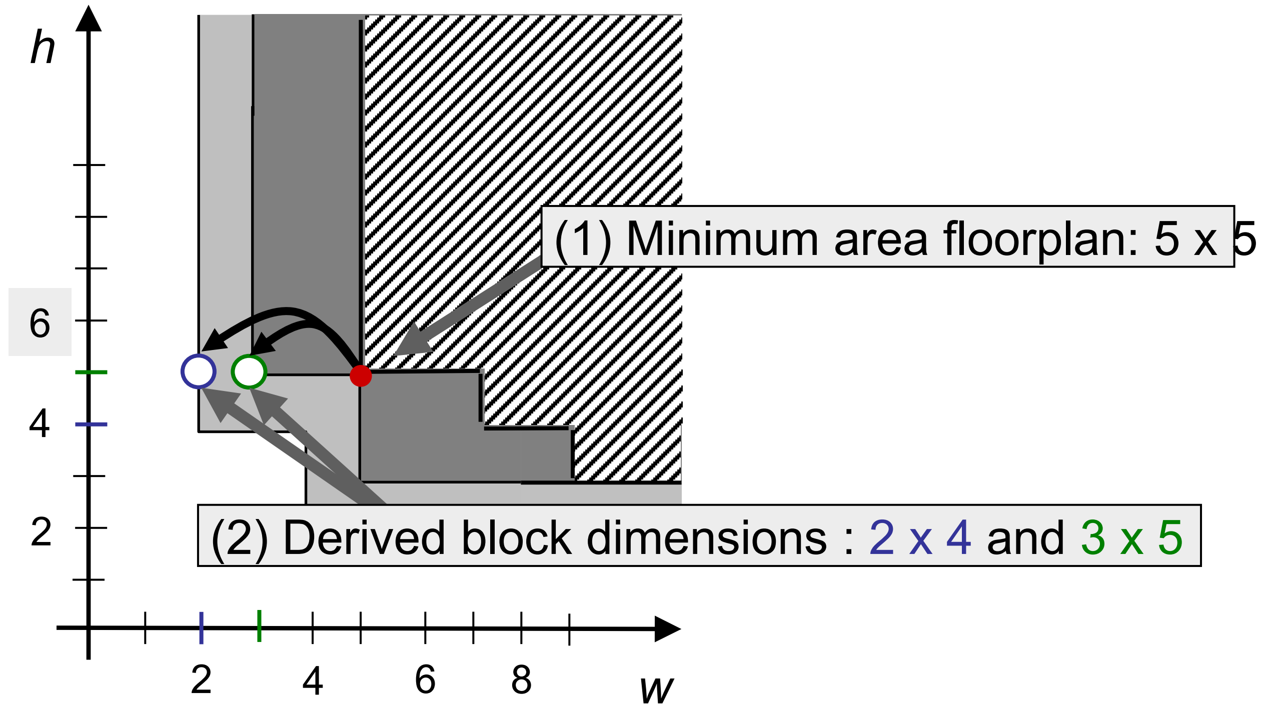
Step 3: Find the individual blocks' dimensions and locations



Horizontal composition

Floorplan Sizing – Example

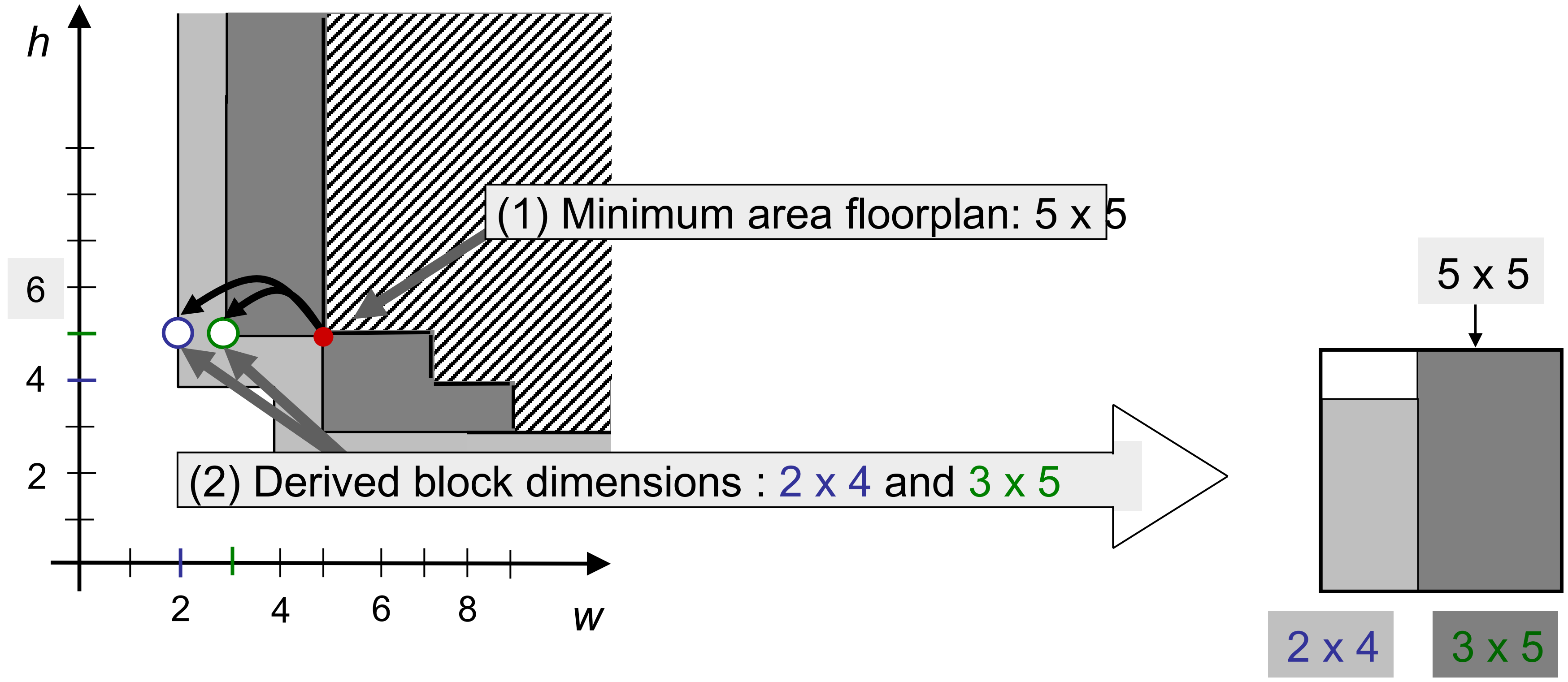
Step 3: Find the individual blocks' dimensions and locations


h
6
4
2
2
4
6
8
w
(1) Minimum area floorplan: 5 x 5
(2) Derived block dimensions : 2 x 4 and 3 x 5

Horizontal composition

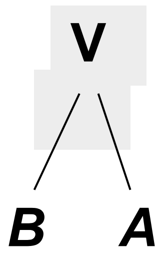
Floorplan Sizing – Example

Step 3: Find the individual blocks' dimensions and locations

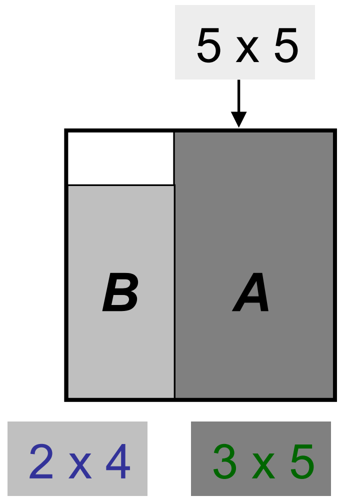

h
6
4
2
2
4
6
8
w
(1) Minimum area floorplan: 5 x 5
(2) Derived block dimensions : 2 x 4 and 3 x 5
5 x 5
2 x 4
3 x 5

Horizontal composition

Floorplan Sizing – Example



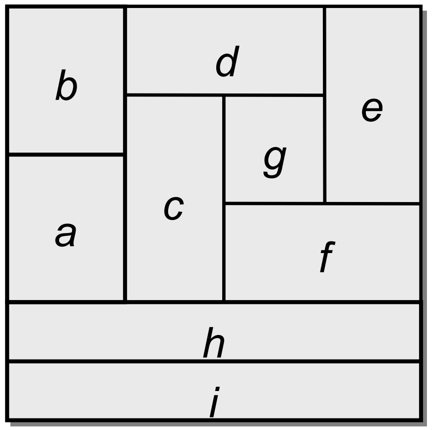
Resulting slicing tree

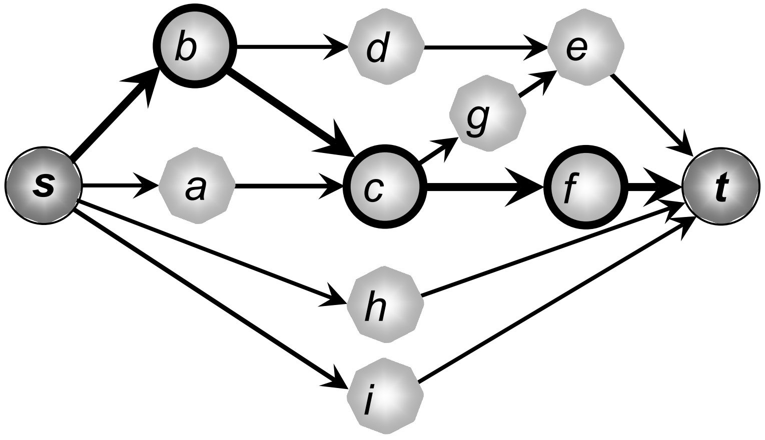


5 min break

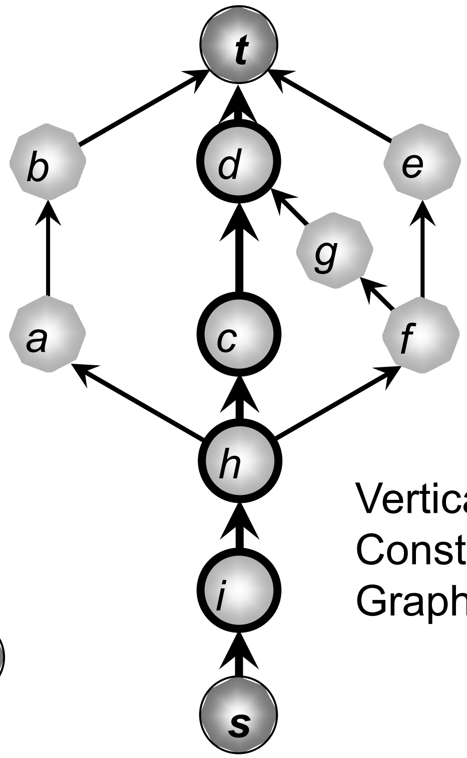
Other Floorplan representations: Constraint Graph Pair Representation of Floorplan

Constraint graphs





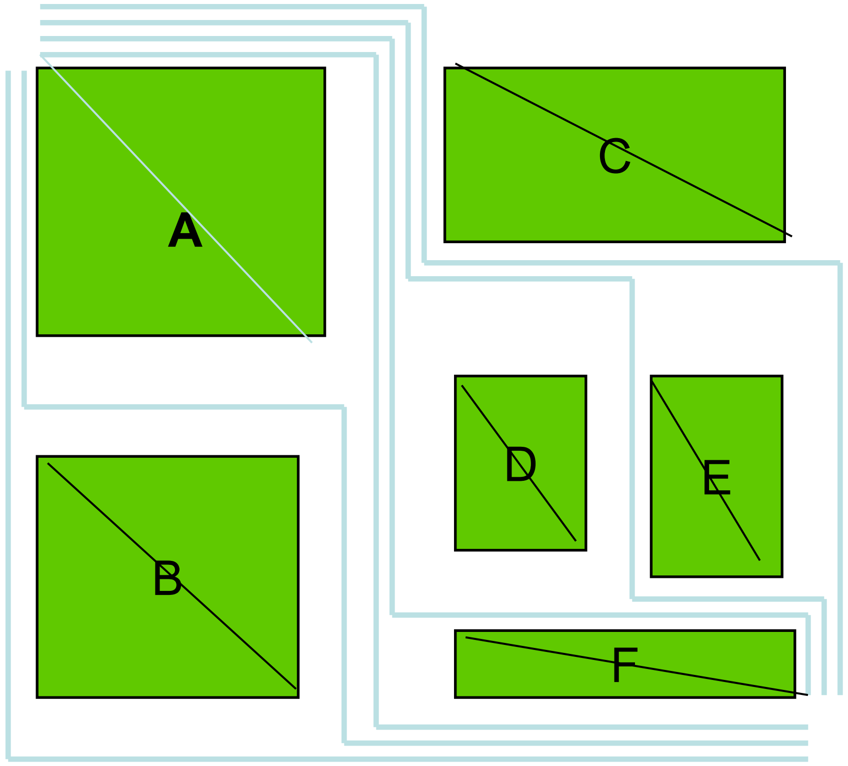
Horizontal Constraint Graph



Vertical Constraint Graph

Sequence Pair Floorplan Representation

- Based on layout partitions by non-overlapping ascending/descending staircases
- Coded in two node sequences
  - E.g., CEDFAB for descending staircases or negative loci and
    - Start with bottom left corner of each module and move up-left and down-right
    - Linear ordering of loci is a sequence
  - ABCDEF for ascending staircases or positive loci
  - Sequence pair is (S+, S-) = (ABCDEF , CEDFAB)
  - (weighted) Longest common subsequence (S+, S-) gives height of floorplan
    - Lcs(ABCDEF , CEDFAB) = AB or CEF or CDF
  - Lcs($S+^R$, S-) gives width of floorplan
    - Lcs(FEDCBA, CEDFAB) = FB or EDB or EDA
- Can represent non-slicing floorplans
- Optimize floorplan by searching over these representations

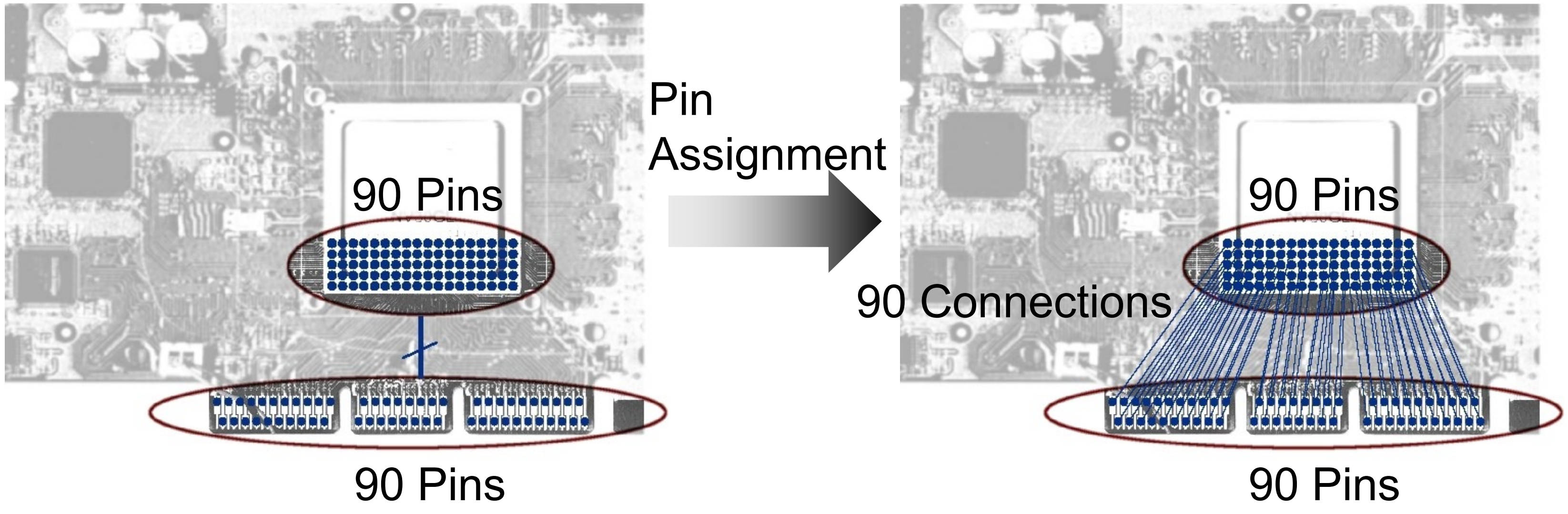


Courtesy K. Yang, UCLA

Pin Assignment

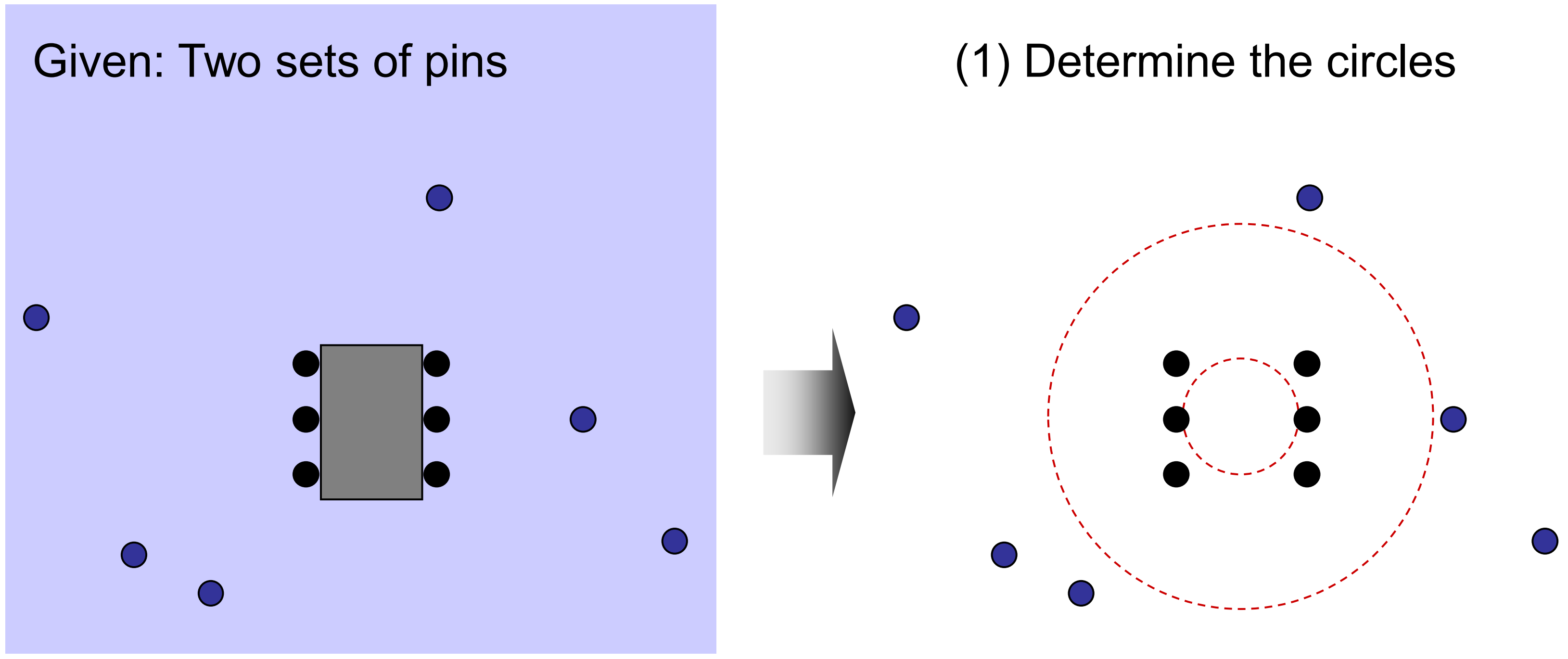
During pin assignment, all nets (signals) are assigned to unique pin locations such that the overall design performance is optimized.

Example: chip i/o to board i/o:



Pin Assignment: Concentric Circle Method

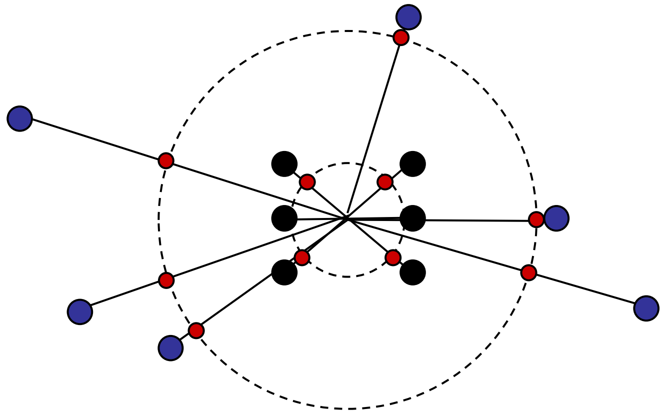
- Goal: Assign legal pin locations so that there is no net overlap and minimum wirelength



Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

Pin Assignment – Example

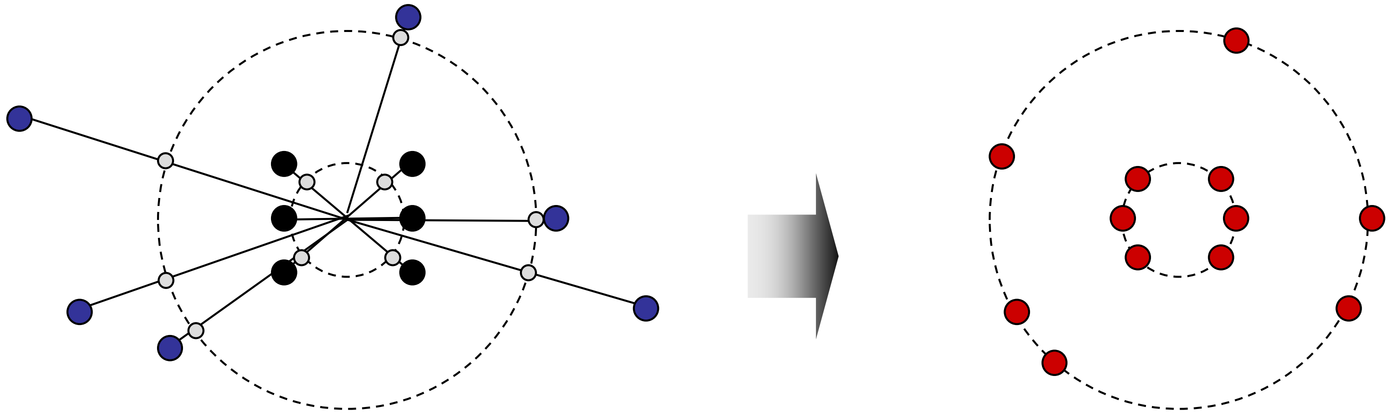
(2) Determine the points



Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

Pin Assignment – Example

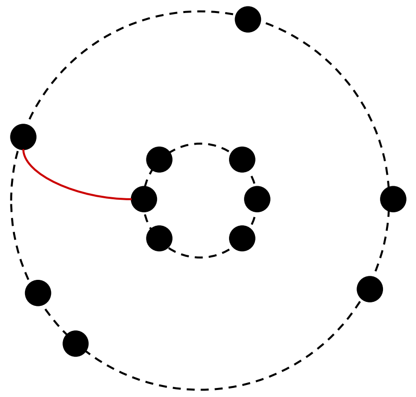
(2) Determine the points



Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

Pin Assignment – Example

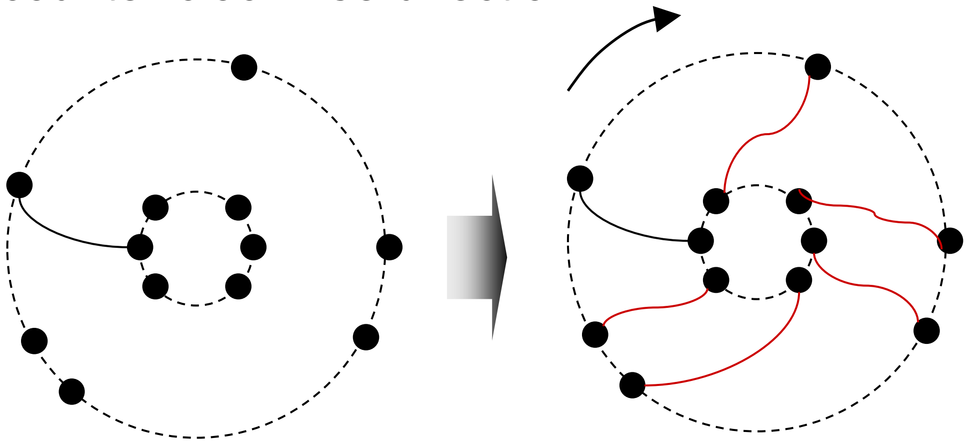
(3) Determine initial mapping: choose an initial point to point mapping arbitrarily



Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

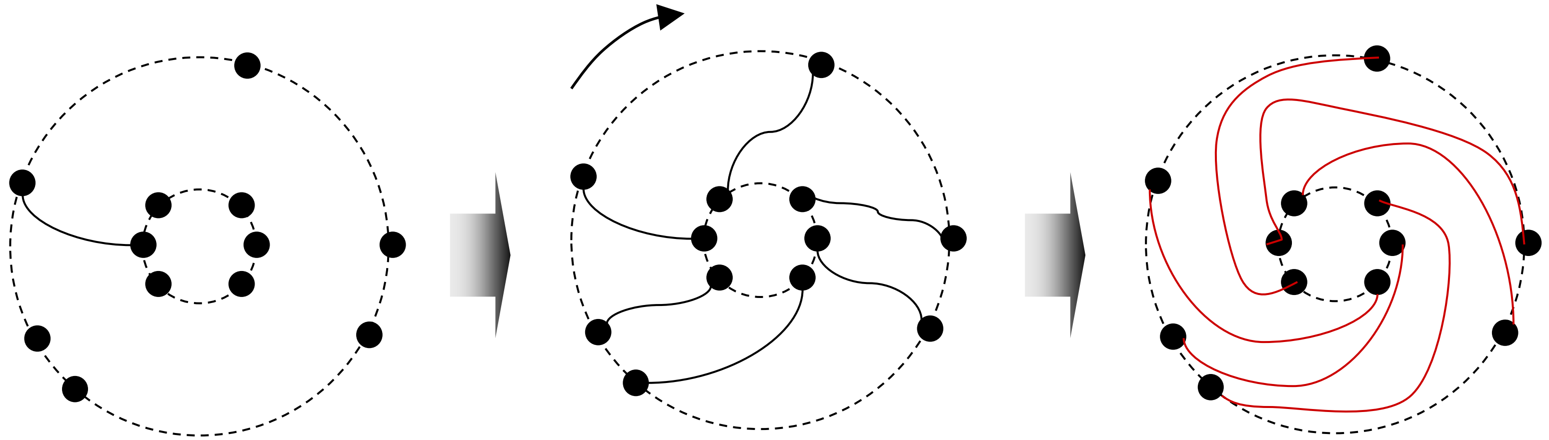
Pin Assignment – Example

(3) Determine initial mapping: assign remaining points clockwise or counter-clockwise direction



Pin Assignment – Example

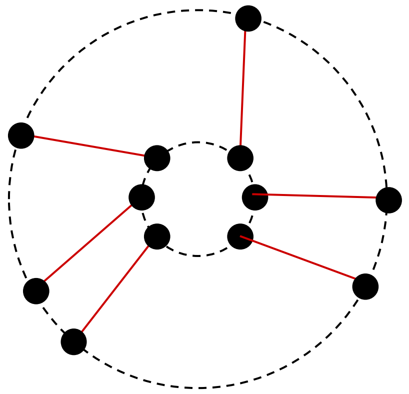
(4) optimize the mapping (complete rotation): Repeat initial mapping with a different start point mapping. Do this till all point mappings have been tried



Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

Pin Assignment – Example

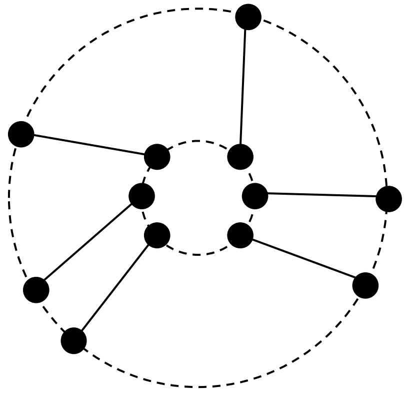
(4) Best mapping (shortest Euclidean distance)

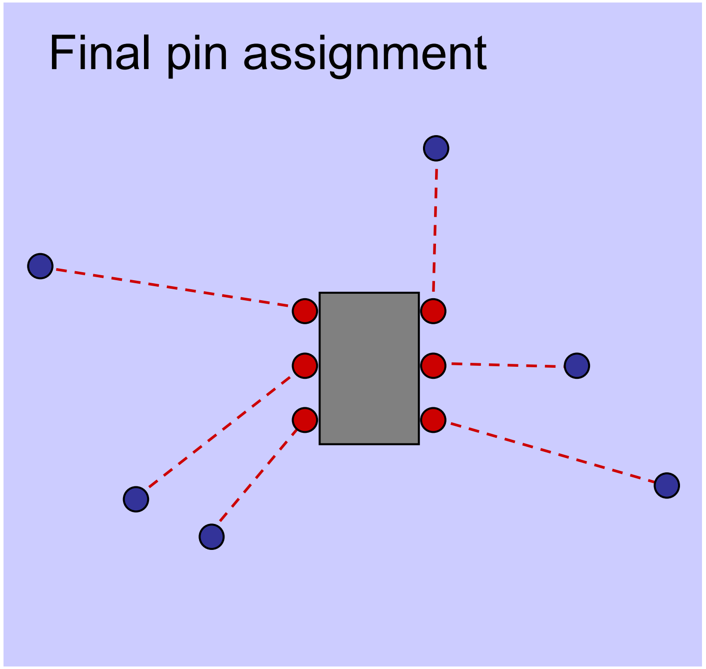


Koren, N. L.: Pin Assignment in Automated Printed Circuit Boards

Pin Assignment – Example

(4) Best mapping

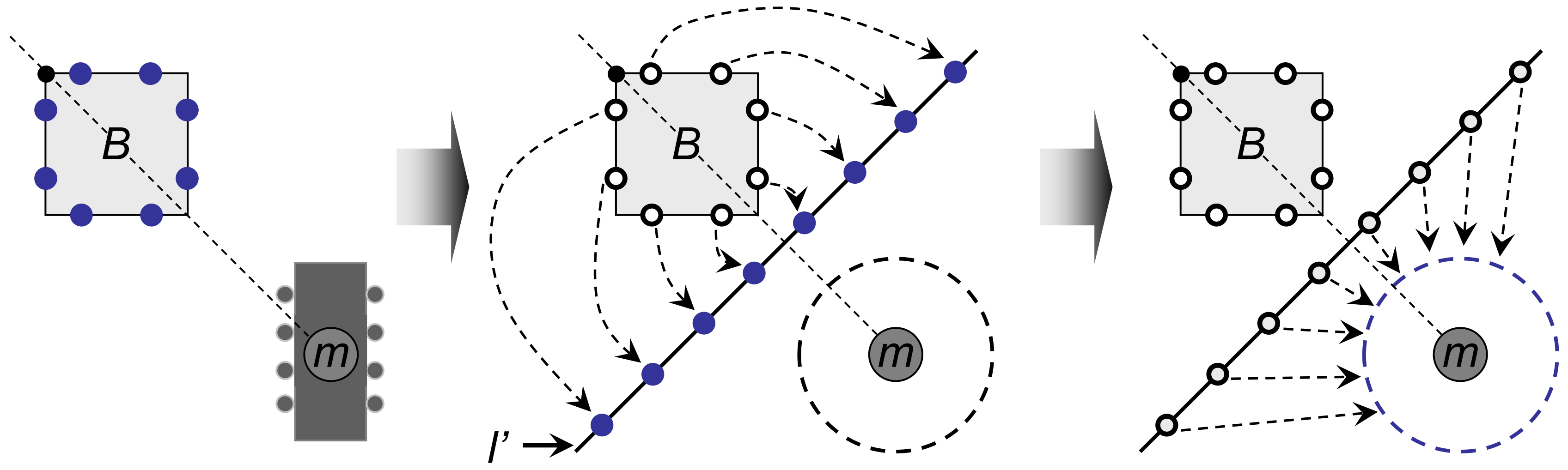




Topological Pin Assignment

Pin assignment to an external block *B*

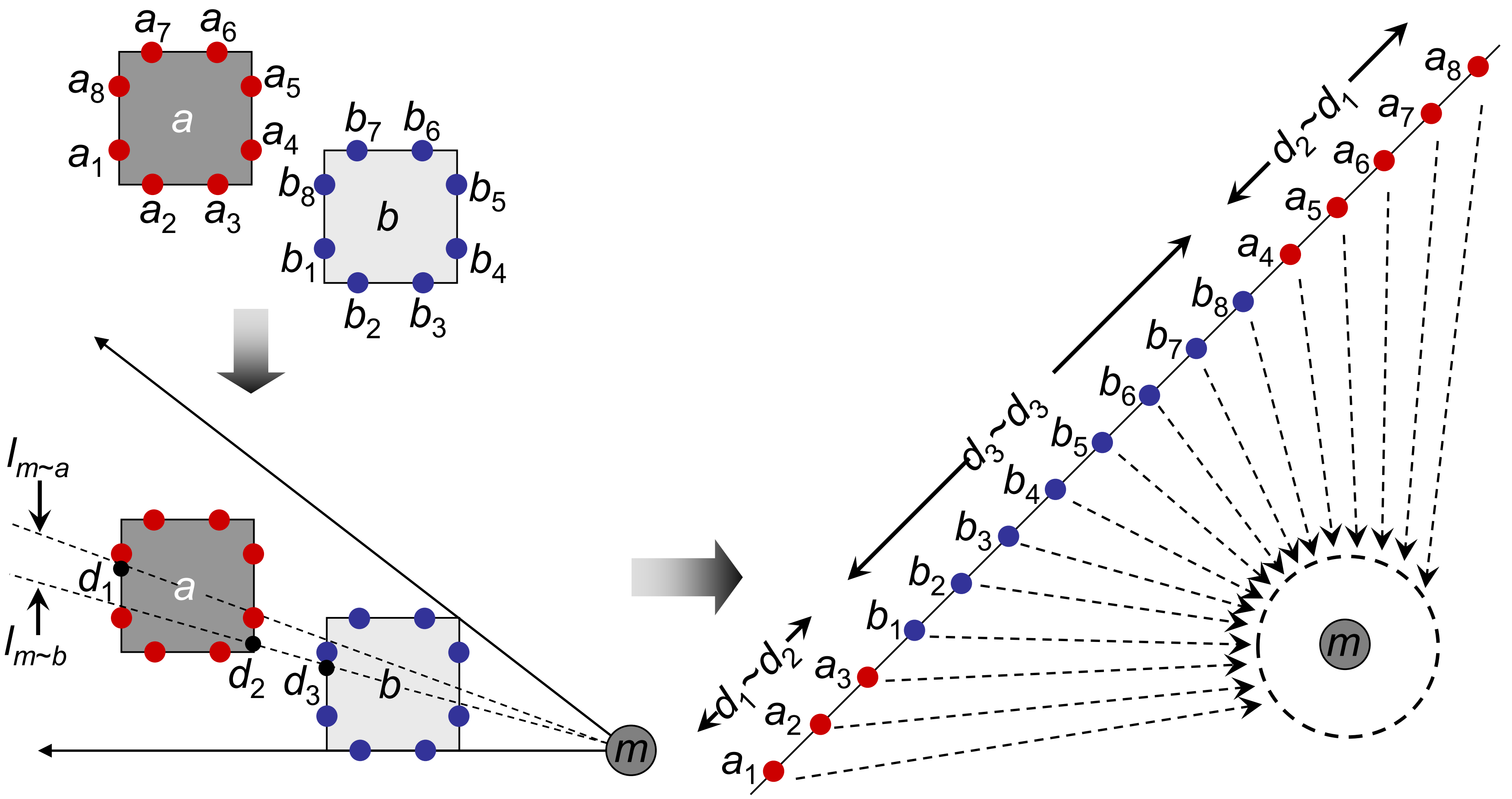
- Draw a midpoint line from center of m through B; draw a dividing line l'; "unwrap" the pins of B onto line from the dividing point (point where midpoint line intersects B farthest from m) ; project pins from line to outer circle



H. N. Brady, "An Approach to Topological Pin Assignment", IEEE Trans. on CAD 3(3) (1984), pp. 250-255

Topological Pin Assignment

Pin assignment to two external blocks A and *B*



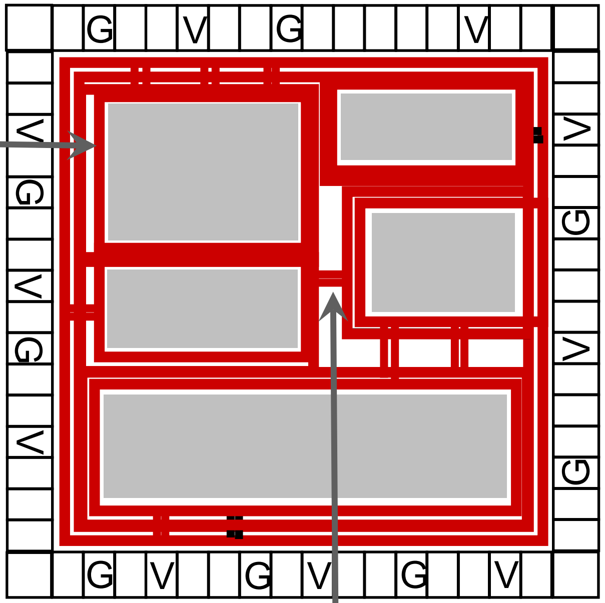
a_7 a_6 a_8 a_5 a a_4 a_1 a_2 a_3
b_7 b_6 b_8 b_5 b b_1 b_4 b_2 b_3
$l_{m\sim a}$
$l_{m\sim b}$
d_1 d_2 d_3
m
$d_2\sim d_1$
$d_3\sim d_3$
$d_1\sim d_2$

H. N. Brady, "An Approach to Topological Pin Assignment", IEEE Trans. on CAD 3(3) (1984), pp. 250-255

Power and Ground Routing

Power-ground distribution for a chip floorplan

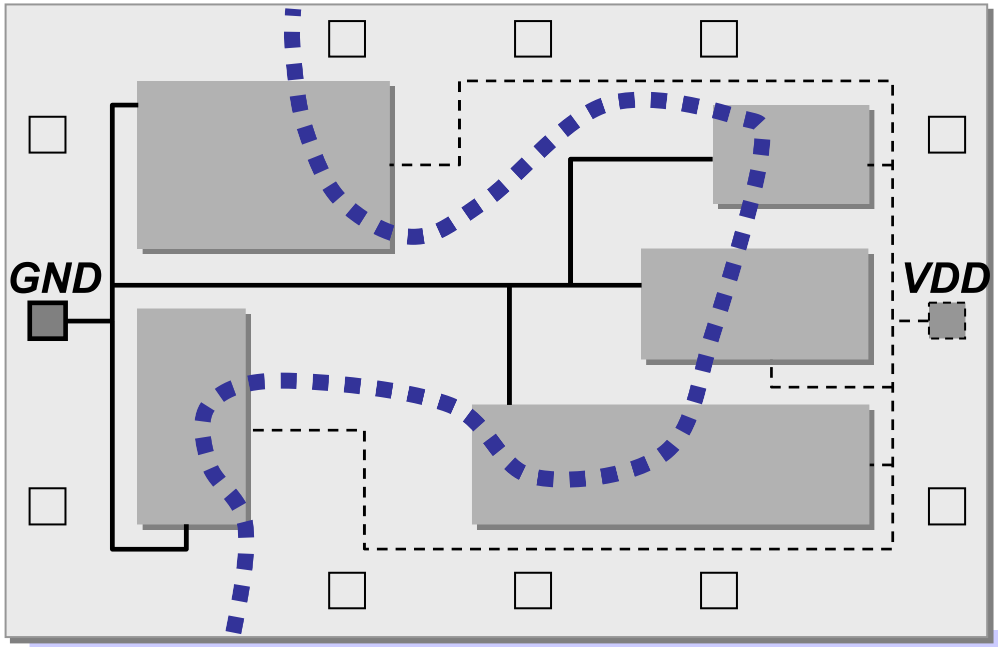
Power and ground rings per block or abutted blocks



Trunks connect rings to each other or to top-level power ring

Power and Ground Routing

Planar routing: More common in analog/custom



Hamiltonian path: shortest path which touches all nodes: split the design → left of path used for ground routing tree; right for Vdd routing tree

Power and Ground Routing

Planar routing

Step 1: Planarize the topology of the nets

- As both power and ground nets must be routed on one layer, the design should be split using the Hamiltonian path

Step 2: Layer assignment

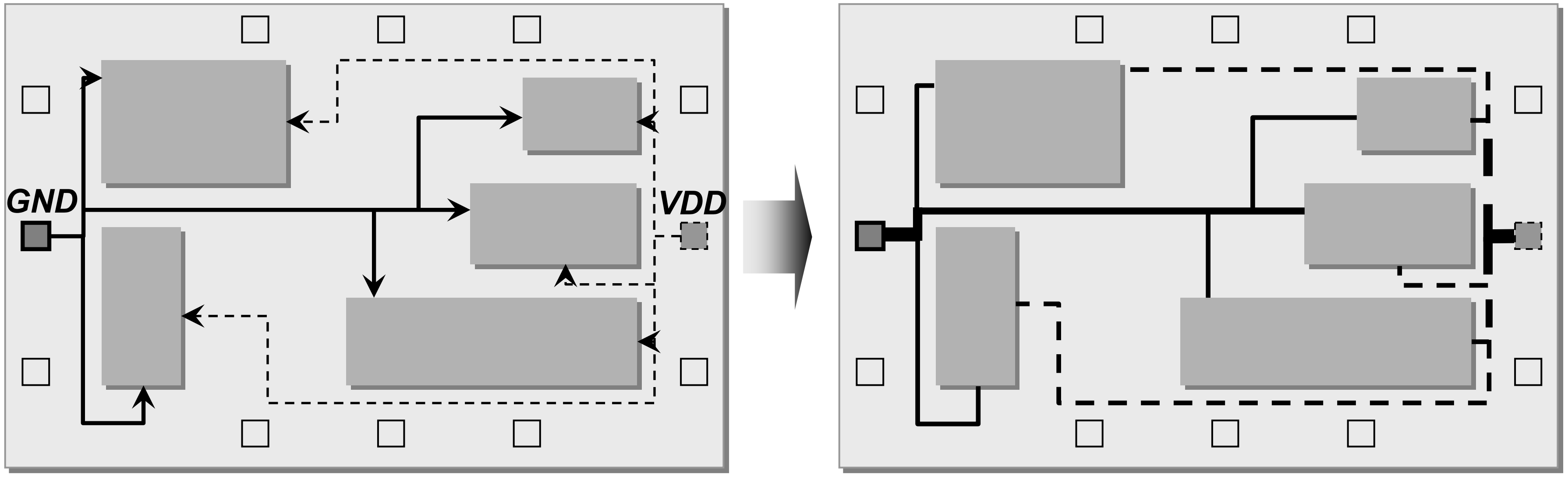
- Net segments are assigned to appropriate routing layers

Step 3: Determining the widths of the net segments

- A segment's width is determined from the sum of the currents from all the cells to which it connects

Power and Ground Routing

Planar routing



Generating topology of the two supply nets

Adjusting widths of the segments with regard to their current loads

Power and Ground Routing

Mesh routing: more common for large digital ICs

Step 1: Creating a ring

- A ring is constructed to surround the entire core area of the chip, and possibly individual blocks.

Step 2: Connecting I/O pads to the ring

Step 3: Creating a mesh

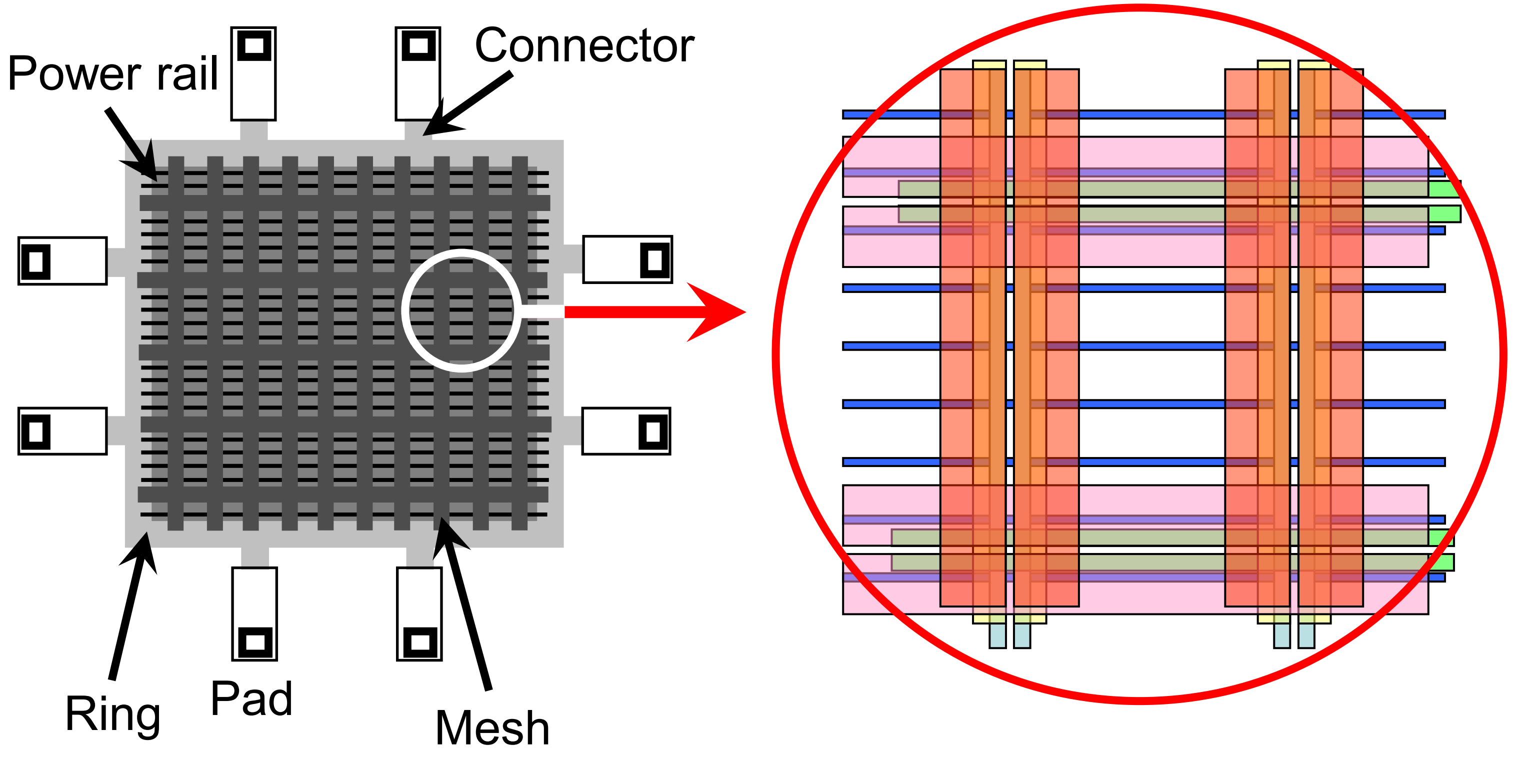
- A power mesh consists of a set of stripes at defined pitches on two or more layers

Step 4: Creating Metal1 rails

Step 5: Connecting the Metal1 rails to the mesh

Power and Ground Routing

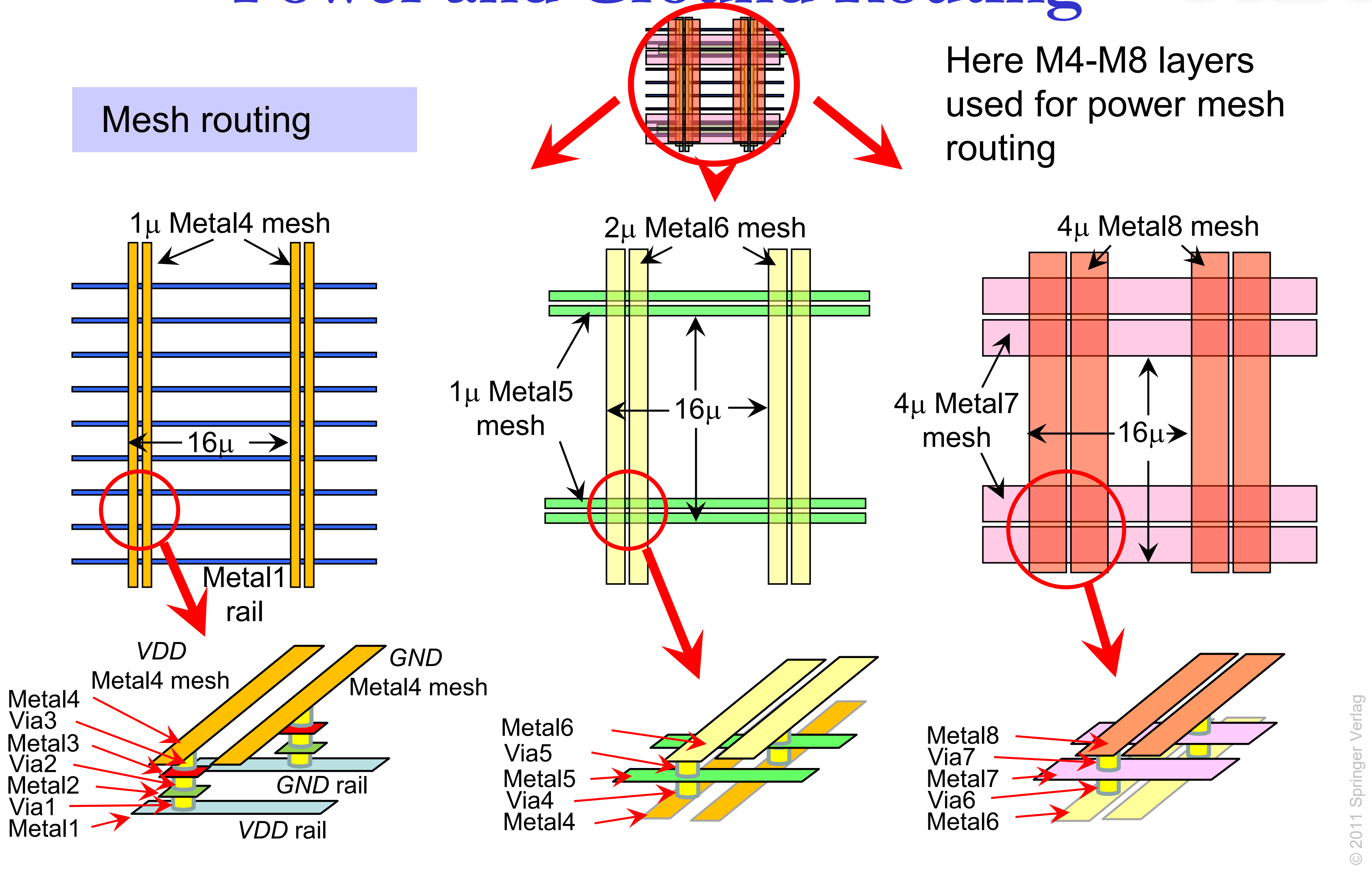
Mesh routing



Power and Ground Routing

Mesh routing

Here M4-M8 layers used for power mesh routing



M1-to-M4 connection **M4-to-M6 connection** **M6-to-M8 connection**

Floorplanning Summary

- Traditional floorplanning
  - Assumes area estimates for top-level circuit modules
  - Determines shapes and locations of circuit modules
  - Minimizes chip area and length of global interconnect
  - Slicing versus non-slicing
    - Representation is key to efficiency and optimality
  - Fixed-outline floorplanning
    - Chip size is fixed, focus on interconnect optimization
    - Can be applied to individual chip partitions (hierarchically)
- Additional aspects
  - Pin assignment
    - Peripheral I/Os versus area-array I/Os
  - Defining channels between blocks for routing and buffering
  - Power and ground routing
    - Planar routing in channels between blocks
    - Can form rings around blocks to increase current supplied and to improve reliability
    - Mesh routing