VLSI Physical Design

Dr. Shubhajit Roy Chowdhury,

Centre for VLSI and Embedded Systems Technology,
IIIT Hyderabad, India

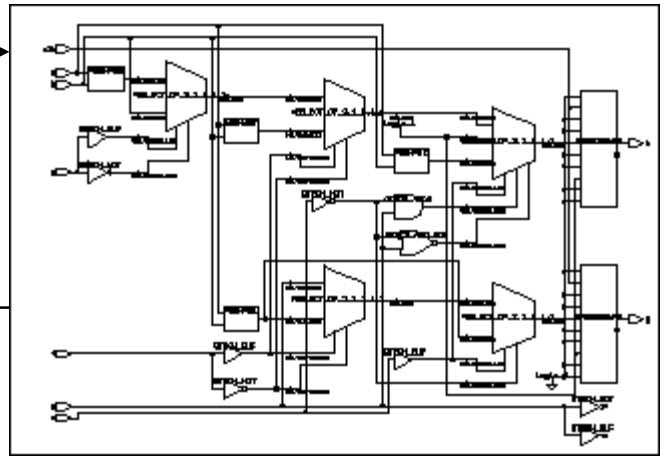
Email: src.vlsi@iiit.ac.in

Synthesis Flow

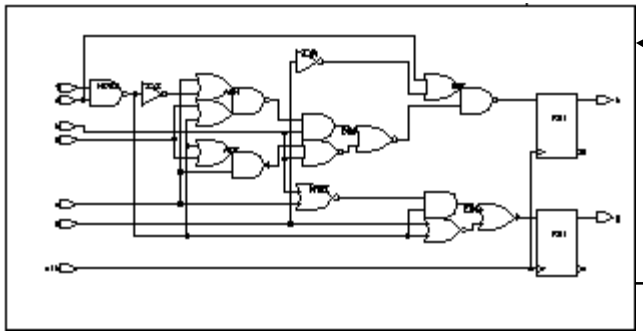
```
module example (clk, a, b, c, d, e, f, g, h);
input clk, a, b, c, d, e, f;
output g, h; reg g, h;

always @(posedge clk) begin
      g = a | b;
      if (d) begin
            if (e) h= a & ~b;
            else h= b;
            if (f) g= c; else h= a^ b;
            end else
                  if (c) h= 1; else h= a ^ b;
            end
endmodule
```

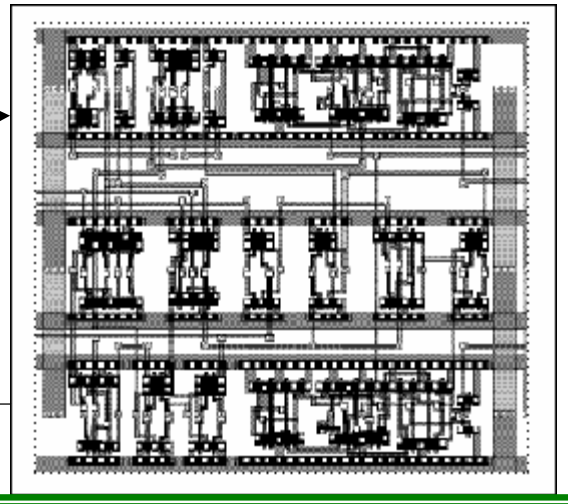
High-Level Synthesis



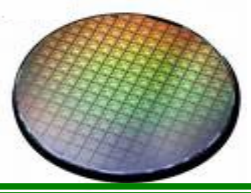
Logic Synthesis



Physical Design

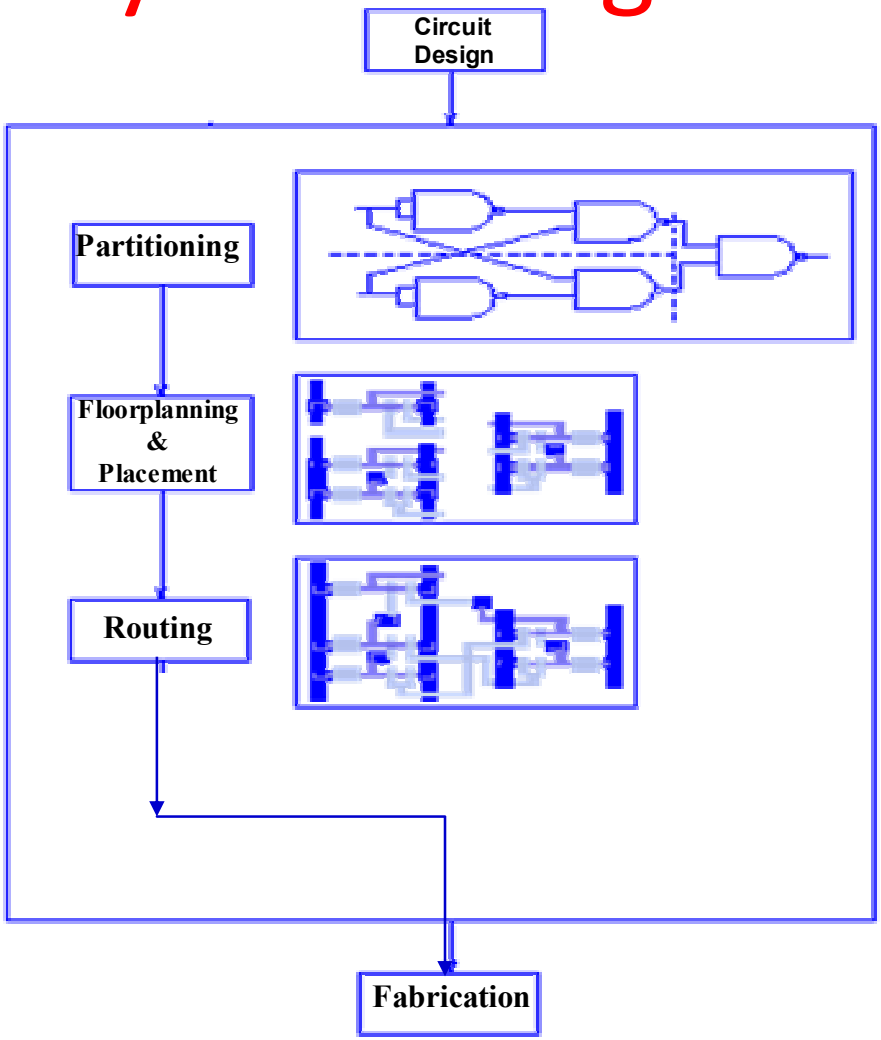


Fabrication and Packaging



Figures adopted with permission from Prof. Ciesielski, UMASS

The Physical Design Process



What is backend?

- Physical Design:

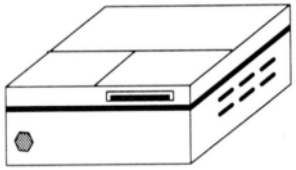
1. FloorPlanning : Architect's job

2. Placement : Builder's job

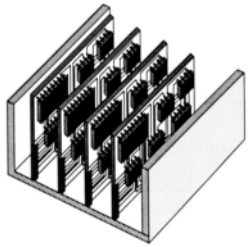
3. Routing : Electrician's job

So, what is Partitioning?

System Level Partitioning

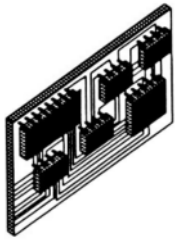


System



PCBs

Board Level Partitioning

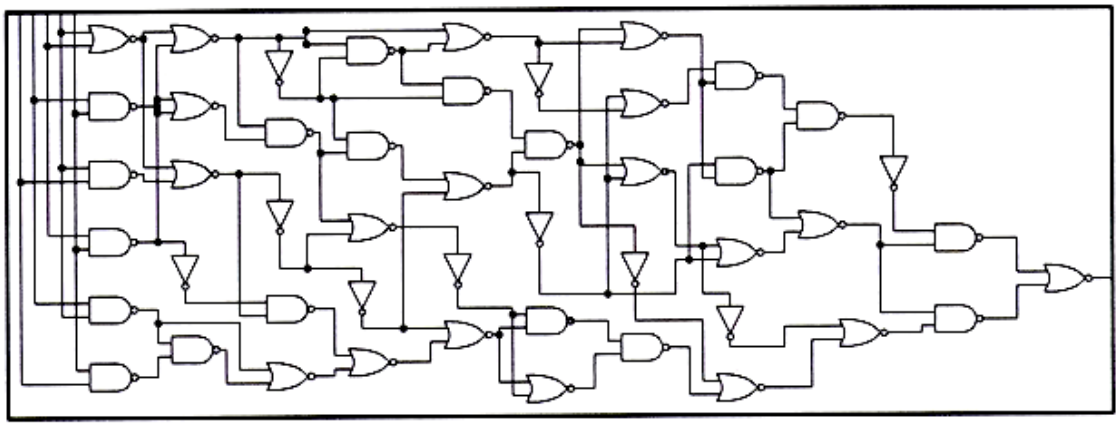


Chips

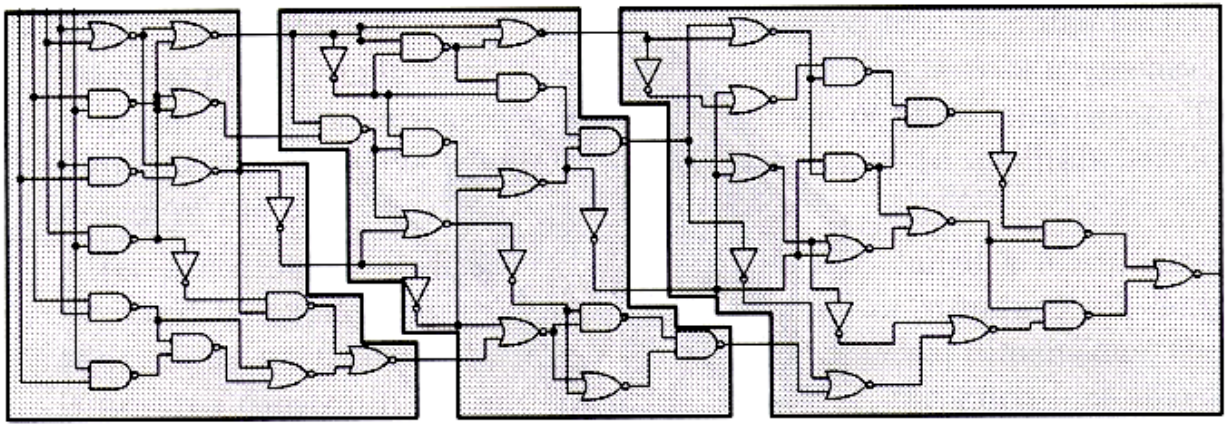
Chip Level Partitioning

Subcircuits / Blocks

Partitioning



(a)



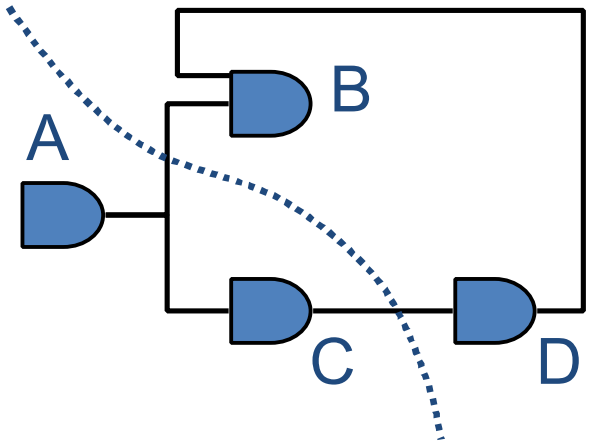
(b)

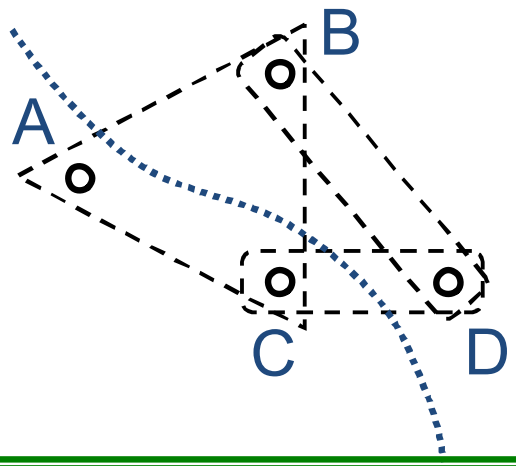
Objectives of Partitioning

- Since each partition can correspond to a chip, interesting objectives are:
  - Minimum number of partitions
    - Subject to maximum size (area) of each partition
  - Minimum number of interconnections between partitions
    - Since they correspond to off-chip wiring with more delay and less reliability
    - Less pin count on ICs (larger IO pins, much higher packaging cost)
  - Balanced partitioning given bound for area of each partition

Circuit Representation by Hypergraph

- Netlist:
  - Gates: A, B, C, D
  - Nets: {A,B,C}, {B,D}, {C,D}
- Hypergraph:
  - Vertices: A, B, C, D
  - Hyperedges: {A,B,C}, {B,D}, {C,D}
  - Vertex label: Gate size/area
  - Hyperedge label: Importance of net (weight)

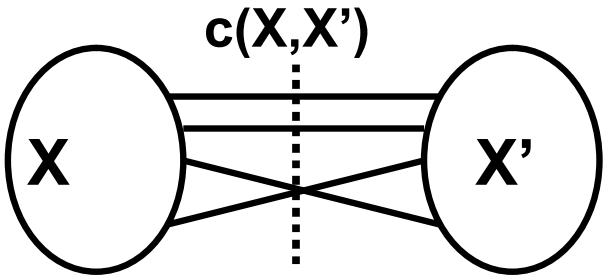




Circuit Partitioning: Formulation

Bi-partitioning formulation:

Minimize interconnections between partitions



- Minimum cut: min c(x, x')
- minimum bisection: min c(x, x') with |x|= |x'|
- minimum ratio-cut: min c(x, x') / |x||x'|

Problem Formulation

- Input: A graph with
  - Set vertices V (|V| = 2n)
  - Set of edges E (|E| = m)
  - Cost c_{AB} for each edge {A, B} in E
- Output: 2 partitions X & Y such that
  - Total cost of edge cuts is minimized
  - Each partition has n vertices
- This problem is NP-Complete!!!!!

A Trivial Approach

- Try all possible bisections and find the best one
- If there are 2n vertices,

  No. of possibilities = $(2n)! / n!^2 = n^{O(n)}$

- For 4 vertices (a,b,c,d), 3 possibilities
  1. X={a,b} & Y={c,d}
  2. X={a,c} & Y={b,d}
  3. X={a,d} & Y={b,c}
- For 100 vertices, 5×10^{28} possibilities
- Need 1.59×10^{13} years if one can try 100M possbilities per second

Definitions

- **Definition 1**: *Consider any node* a *in block* X. *The contribution of node* a *to the cutset is called the external cost of* a *and is denoted as* Ea, *where*

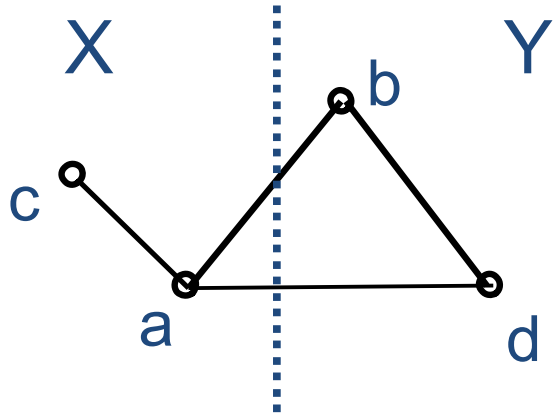
  $E_a = \sum c_{av}$ (for all v in Y)

- **Definition 2:** *The internal cost* Ia *of node* a in X *is defined as follows:*

  $I_a = \sum c_{av}$ (for all v in X)

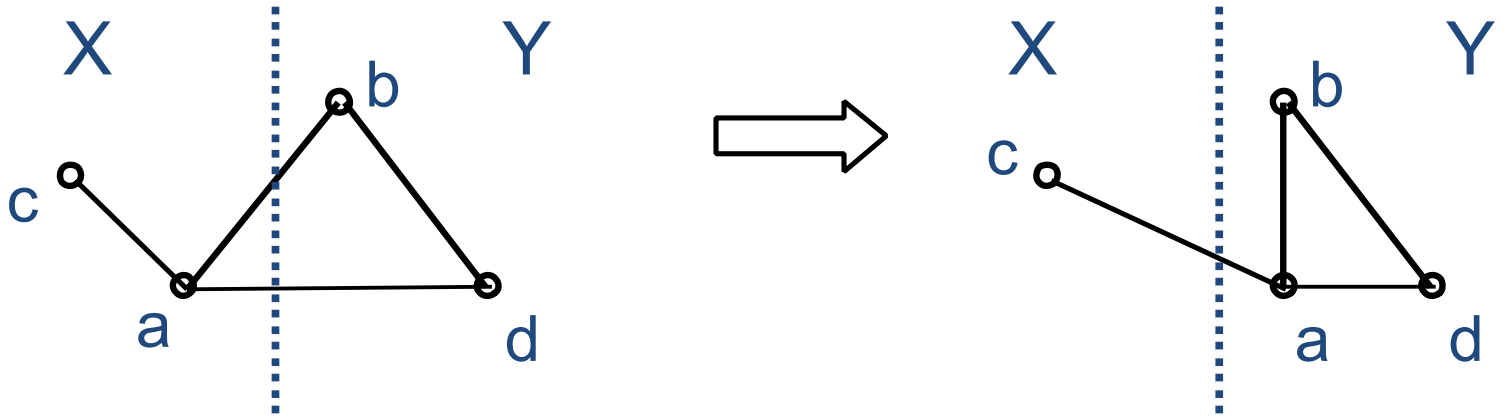
Example

- External cost (connection) $E_a = 2$
- Internal cost $I_a = 1$



Idea of Kernighan Lin (KL) Algorithm

- D_a = Decrease in cut value if moving a = E_a-I_a
  - Moving node a from block X to block Y would decrease the value of the cutset by E_a and increase it by I_a



$D_a = 2-1 = 1$
$D_b = 1-1 = 0$

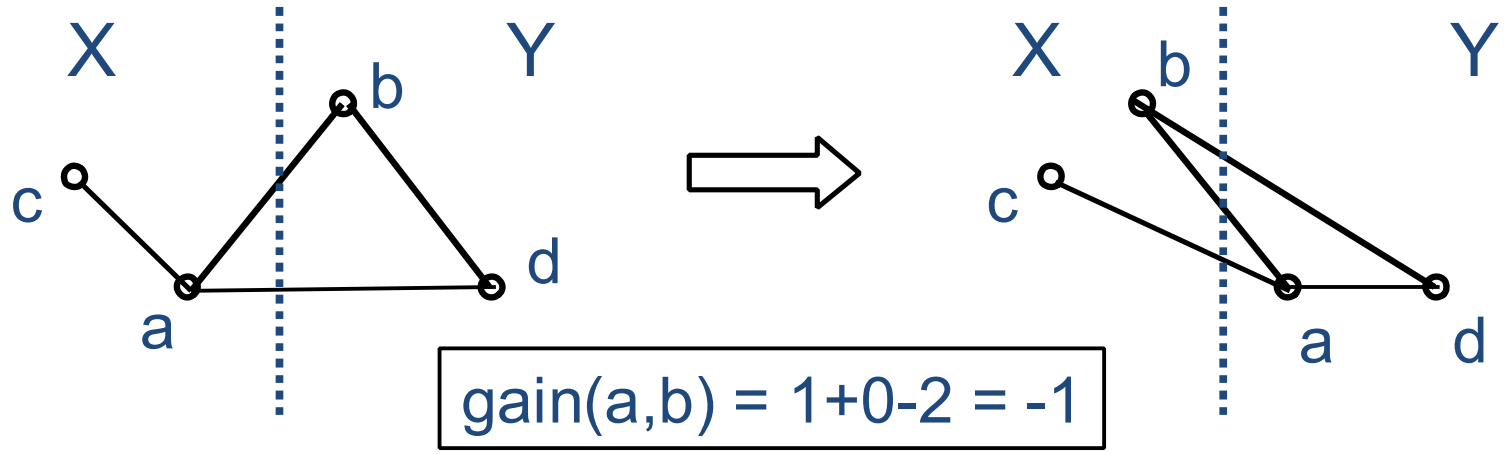
Useful Lemmas

- To maintain balanced partition, we must move a node from Y to X each time we move a node from X to Y
- The effect of swapping two modules a in X with b in Y is characterized by the following lemma:
- **Lemma 1:** *If two elements* a in X *and* b in Y *are interchanged, the reduction in the cost is given by:*

$$\text{gain}(a,b) = g_{ab} = D_a + D_b - 2c_{ab}$$

Example

- If switch a & b, gain(a,b) = $D_a+D_b-2c_{ab}$
  - c_{ab}: edge cost for ab



Useful Lemmas

- The following lemma tells us how to update the D-values after a swap.
- **Lemma 2:** *If two elements* a in X *and* b in Y *are interchanged, then the new* D-*values are given by*

  $D'_k = D_k + 2c_{ka} - 2c_{kb}$; for all k in X – {a}

  $D'_m = D_m + 2c_{mb} - 2c_{ma}$; for all m in Y – {b}

- Notice that if a module j is neither connected to a nor to b then $c_{ja} = c_{jb} = 0$, and, $D_j = D'_j$

Overview of KL Algorithm

- Start from an initial partition {X,Y} of n elements each
- Use lemmas 1 and 2 together with a greedy procedure to identify two subsets A in X, and B in Y, of equal cardinality, such that when interchanged, the partition cost is improved
- A and B may be empty, indicating in that case that the current partition can no longer be improved

Idea of KL Algorithm

- Start with any initial legal partitions X and Y
- A pass (exchanging each vertex exactly once) is described below:
  1. For i := 1 to n do
     From the unlocked (unexchanged) vertices,
     choose a pair (A,B) s.t. gain(A,B) is largest
     Exchange A and B. Lock A and B.
     Let g_i = gain(A,B)
  2. Find the k s.t. $G=g_1+...+g_k$ is maximized
  3. Switch the first k pairs
- Repeat the pass until there is no improvement (G=0)

State Space Search Problem

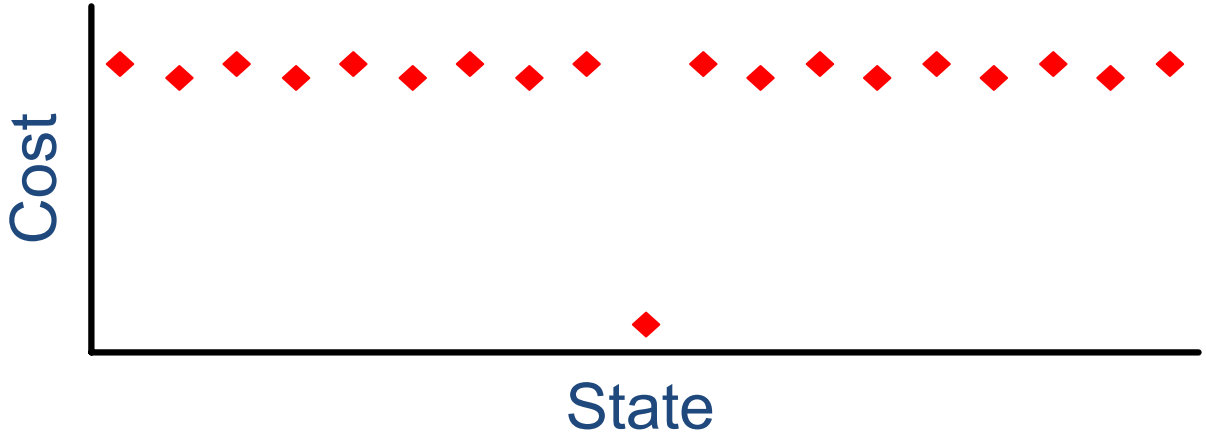
- Combinatorial optimization problems (like partitioning) can be thought as a State Space Search Problem.
- A State is just a configuration of the combinatorial objects involved.
- The State Space is the set of all possible states (configurations).
- A Neighbourhood Structure is also defined (which states can one go in one step).
- There is a cost corresponding to each state.
- Search for the min (or max) cost state.

Greedy Algorithm

- A very simple technique for State Space Search Problem.
- Start from any state.
- Always move to a neighbor with the min cost (assume minimization problem).
- Stop when all neighbors have a higher cost than the current state.

Problem with Greedy Algorithms

- Easily get stuck at local minimum.
- Will obtain non-optimal solutions.



- Optimal only for convex (or concave for maximization) funtions.

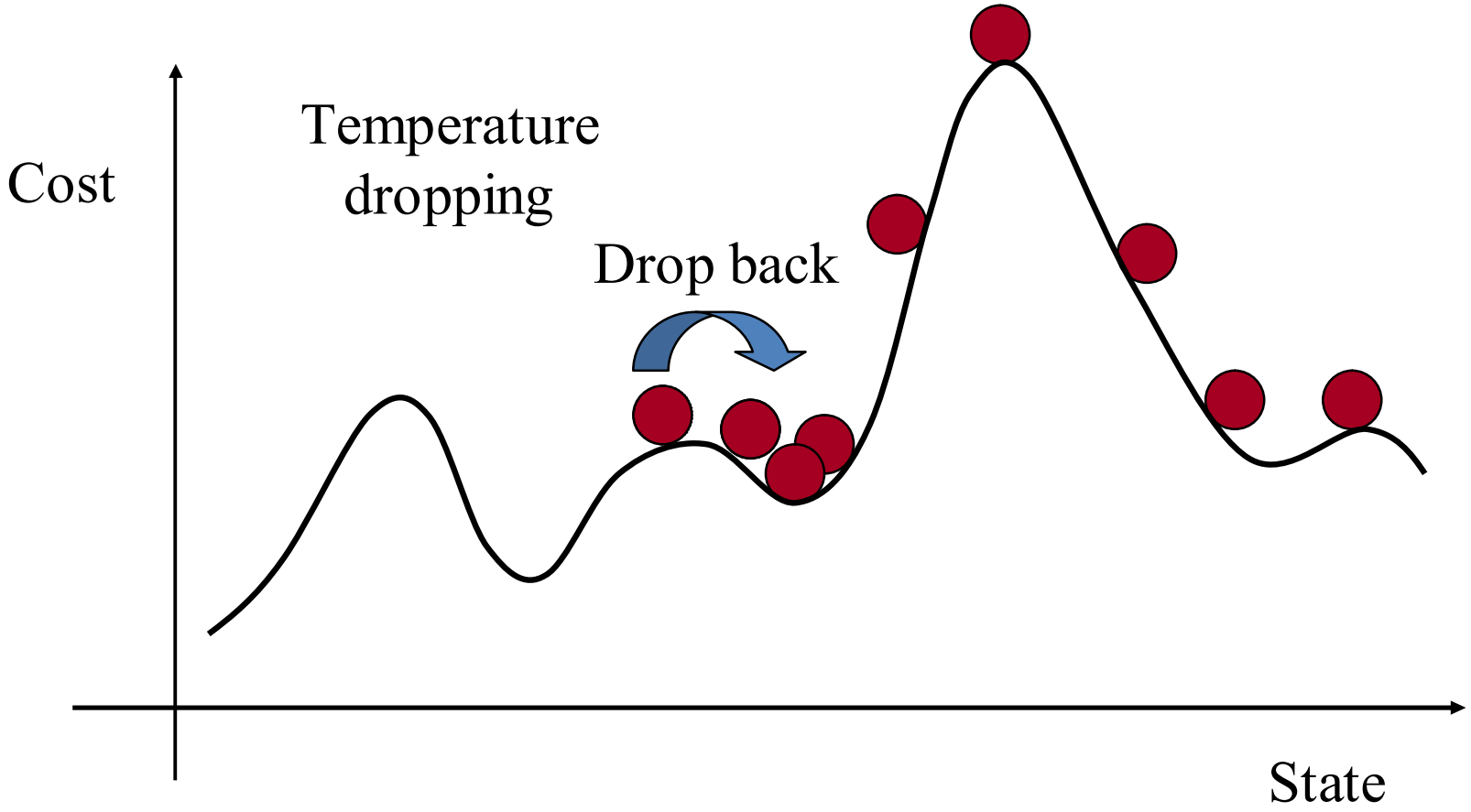
Simulated Annealing

- Very general search technique.
- Try to avoid being trapped in local minimum by making probabilistic moves.
- Popularize as a heuristic for optimization by:
  - Kirkpatrick, Gelatt and Vecchi, “Optimization by Simulated Annealing”, Science, 220(4598):498-516, May 1983.

Basic Idea of Simulated Annealing

- Inspired by the *Annealing Process*:
  - The process of carefully cooling molten metals in order to obtain a good crystal structure.
  - First, metal is heated to a very high temperature.
  - Then slowly cooled.
  - By cooling at a proper rate, atoms will have an increased chance to regain proper crystal structure.
- Attaining a min cost state in simulated annealing is analogous to attaining a good crystal structure in annealing.

Simulated Annealing



The Simulated Annealing Procedure

Let t be the initial temperature.

Repeat

Repeat

- Pick a neighbor of the current state randomly.
- Let c = cost of current state.

  Let c' = cost of the neighbour picked.
- If c' < c, then move to the neighbour (downhill move).
- If c' > c, then move to the neighbour with probablility $e^{-(c'-c)/t}$ (uphill move).

Until equilibrium is reached.

Reduce t according to cooling schedule.

Until Freezing point is reached.

Things to decide while using Simulated Annealing

- When solving a combinatorial problem, we have to decide:
  - The state space
  - The neighborhood structure
  - The cost function
  - The initial state
  - The initial temperature
  - The cooling schedule (how to change t)
  - The freezing point

Common Cooling Schedules

- Initial temperature, Cooling schedule, and freezing point are usually experimentally determined.
- Some common cooling schedules:
  - $t = \alpha t$, where α is typically around 0.95
  - $t = e^{-\beta t}\, t$, where β is typically around 0.7
  - ……

Floorplanning

The floorplanning problem is to plan the *positions* and *shapes* of the modules at the beginning of the design cycle to optimize the circuit performance:

- chip area
- total wirelength
- delay of critical path
- routability
- others, e.g., noise, heat dissipation, etc.

Floorplanning v.s. Placement

- Both determines block positions to optimize the circuit performance.
- Floorplanning:
  - Details like shapes of blocks, I/O pin positions, etc. are not yet fixed (blocks with flexible shape are called soft blocks).
- Placement:
  - Details like module shapes and I/O pin positions are fixed (blocks with no flexibility in shape are called hard blocks).

Floorplanning Problem

- Input:
  - n Blocks with areas $A_1, \ldots, A_n$
  - Bounds r_i and s_i on the aspect ratio of block B_i
- Output:
  - Coordinates (x_i, y_i), width w_i and height h_i for each block such that $h_i w_i = A_i$ and

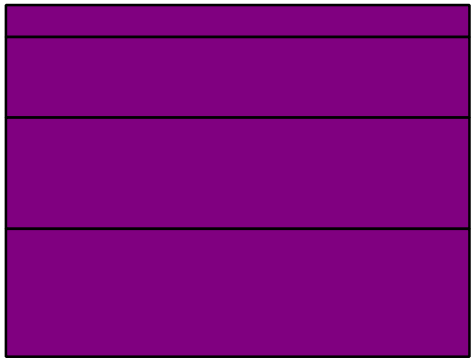
$$r_i \leq h_i/w_i \leq s_i$$

- Objective:
  - To optimize the circuit performance.

Bounds on Aspect Ratios

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

- Sure!



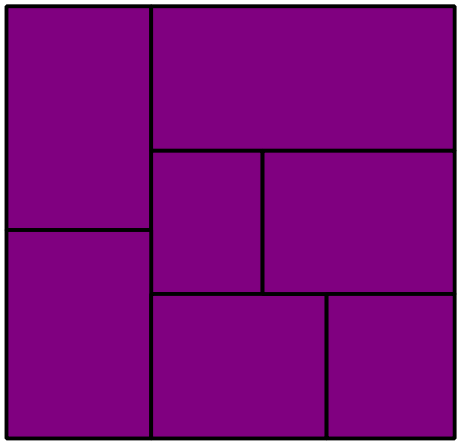
But we don't want to layout blocks as long strips, so we require

$$r_i \leq h_i/w_i \leq s_i \text{ for each i.}$$

Slicing and Non-Slicing Floorplan

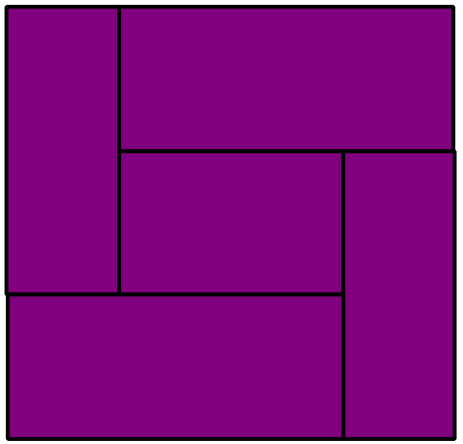
- Slicing Floorplan:

  One that can be obtained by repetitively subdividing (slicing) rectangles horizontally or vertically.



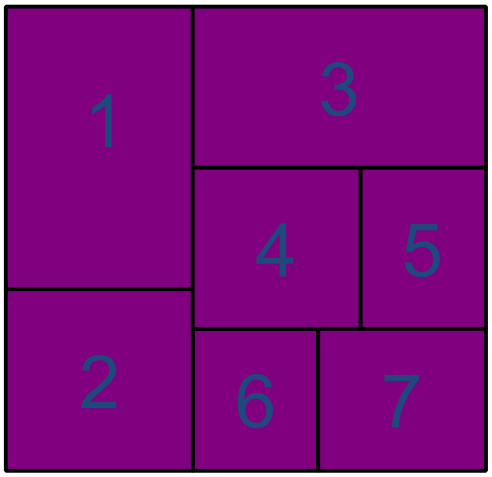
- Non-Slicing Floorplan:

  One that may not be obtained by repetitively subdividing alone.

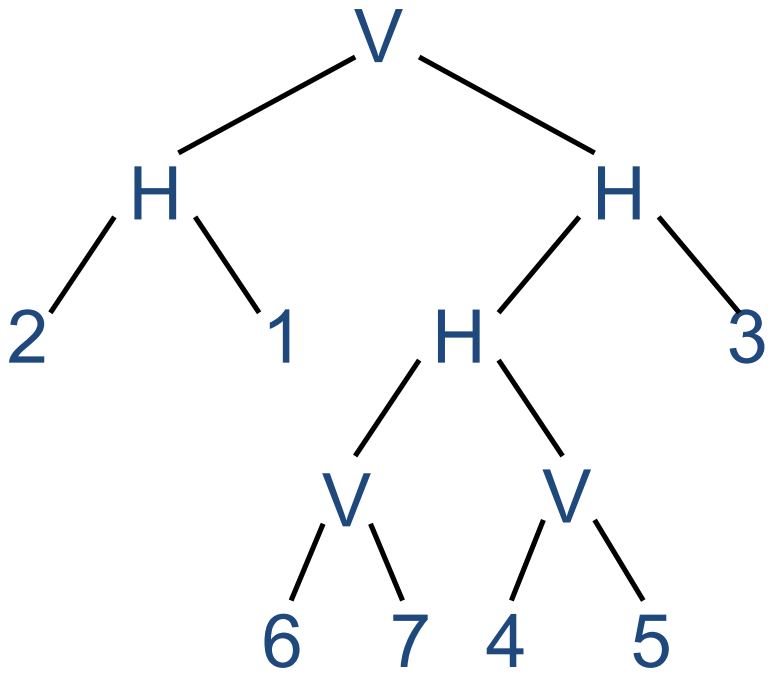


Representation of Slicing Floorplan

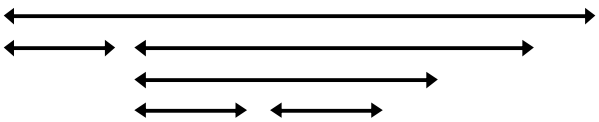
Slicing Floorplan



Slicing Tree



Polish Expression
(postorder traversal of slicing tree)



21H67V45VH3HV

Annealing Schedule

- $T_i = \alpha T_{i-1}$ where α=0.85
- At each temperature, try k x n moves
  (k is around 5 to 10)
- Terminate the annealing process if
  - either # of accepted moves < 5%
  - or the temperate is low enough

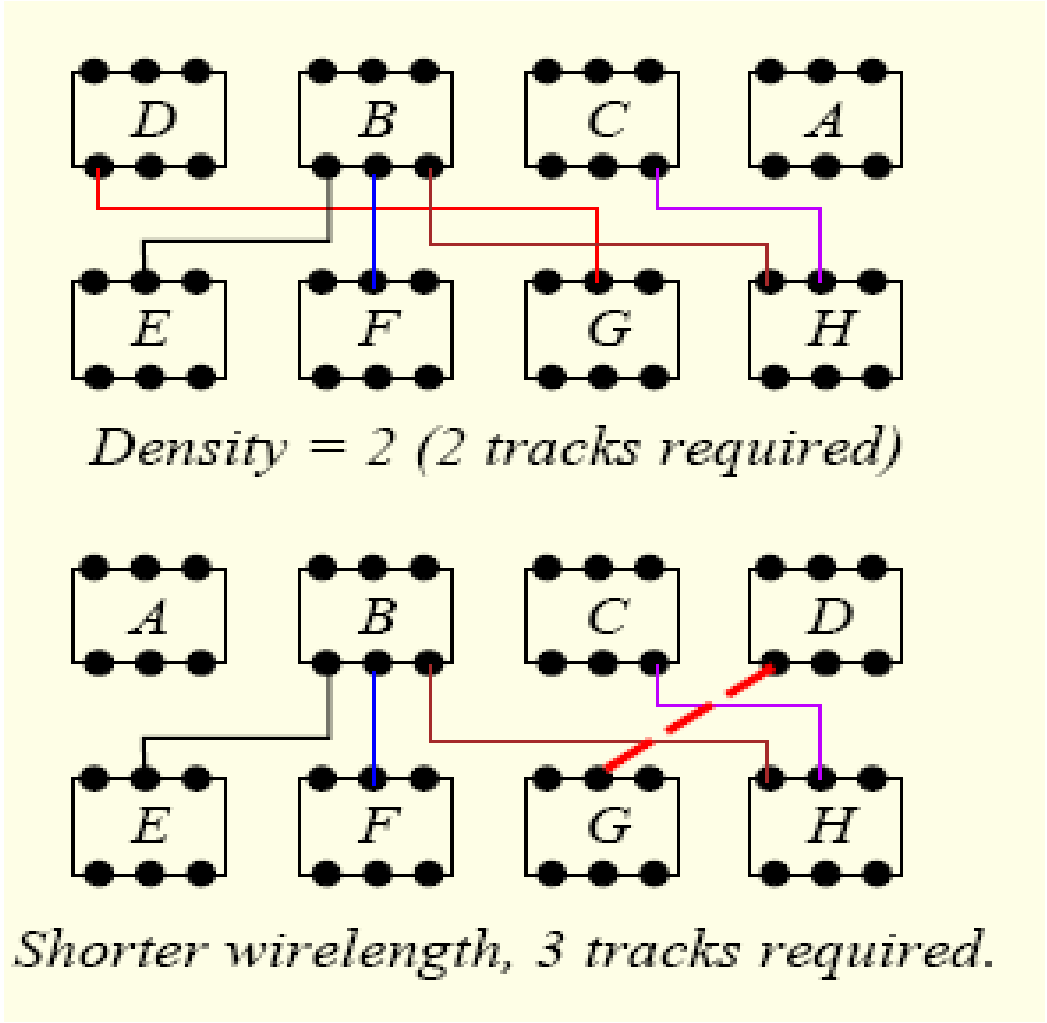
Placement Problem formulation

- Input:
  - Blocks (standard cells and macros) B_1, ... , B_n
  - Shapes and Pin Positions for each block B_i
  - Nets N_1, ... , N_m
- Output:
  - Coordinates (x_i , y_i) for block B_i.
  - No overlaps between blocks
  - The total wire length is minimized
  - The area of the resulting block is minimized or given a fixed die
- Other consideration: timing, routability, clock, buffering and interaction with physical synthesis

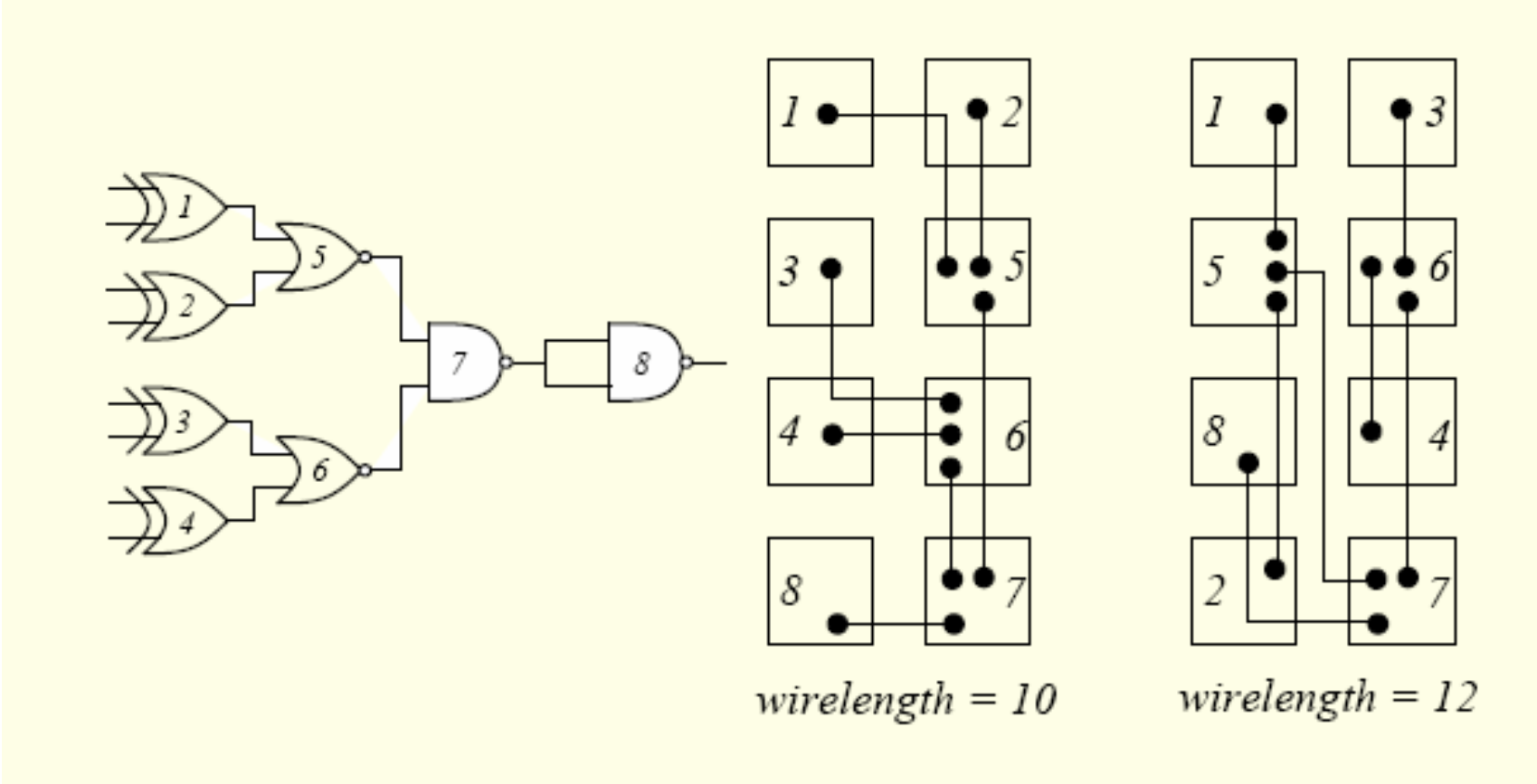
Importance of Placement

- Placement is a key step in physical design
- Poor placement consumes large area, leads to difficult/ impossible routing task
- Quality of placement:
  - Layout area
  - Routability
  - Performance (usually timing, measured by delay of critical/ longest net)

Placement affects chip area



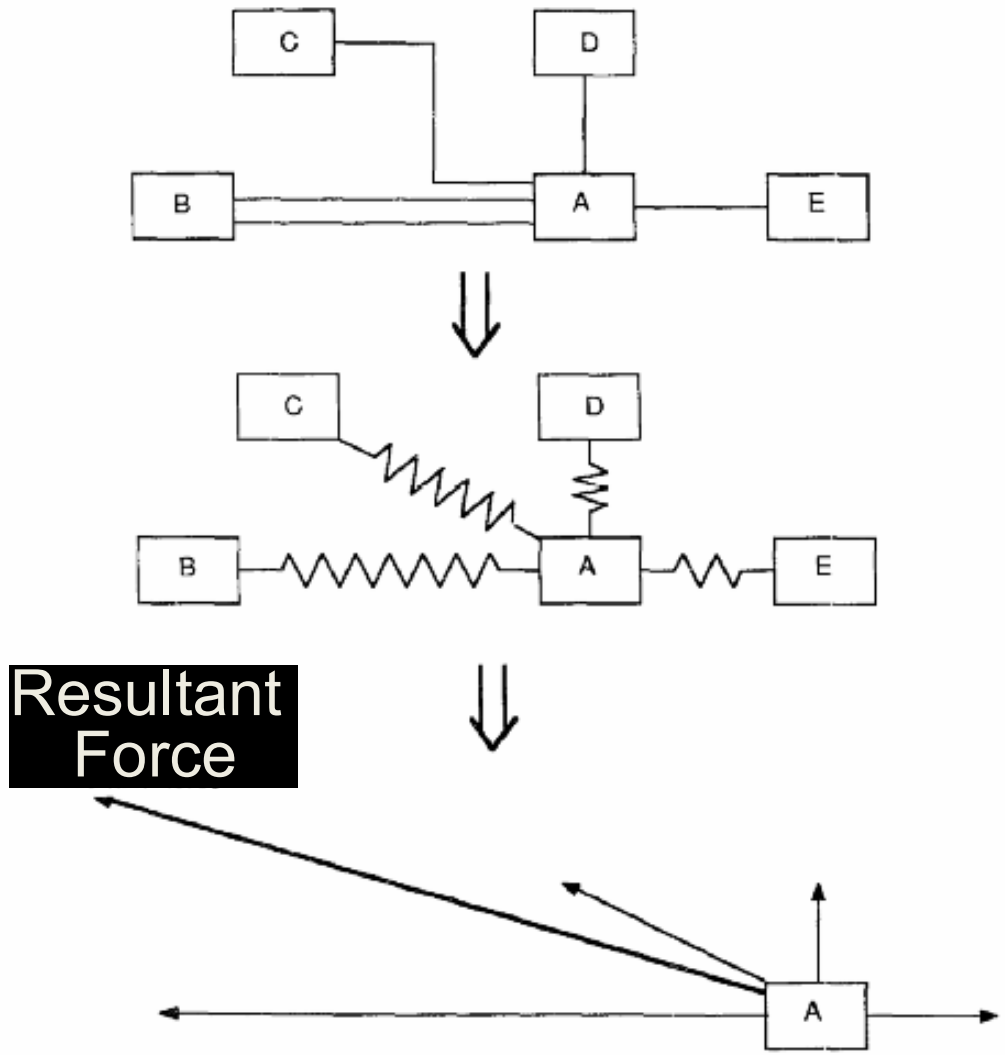
…And also Wire Length



Force Directed Approach

- Transform the placement problem to the classical mechanics problem of a system of objects attached to springs
- Analogies:
  - Module (Block/Cell/Gate) = Object
  - Net = Spring
  - Net weight = Spring constant
  - Optimal placement = Equilibrium configuration

An Example



Force Calculation

- Hooke's Law:
  - Force = Spring Constant x Distance
- Can consider forces in x- and y-direction separately:

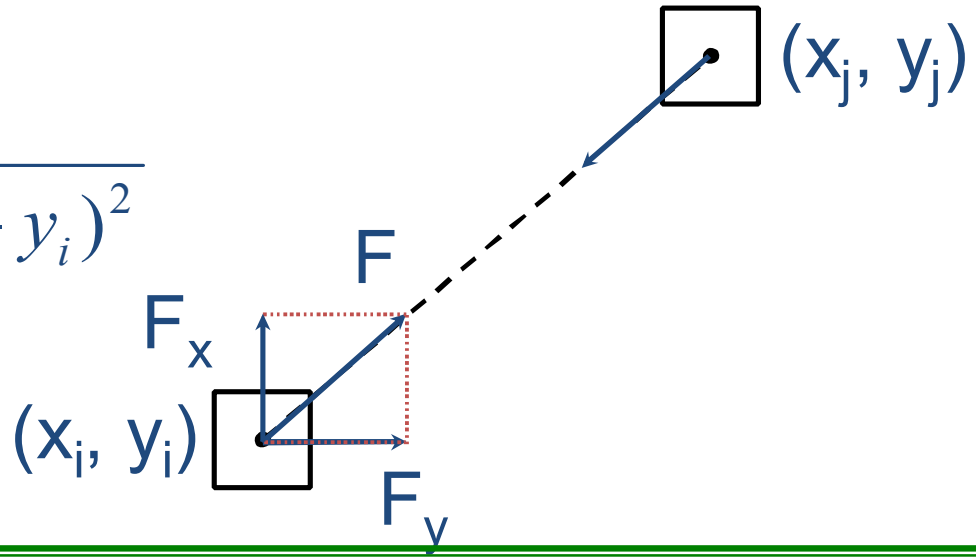
$$\text{Distance } d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

$$\text{Net Cost } c_{ij}$$

$$F = c_{ij}\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

$$F_x = c_{ij}(x_j - x_i)$$

$$F_y = c_{ij}(y_j - y_i)$$



Problem Formulation

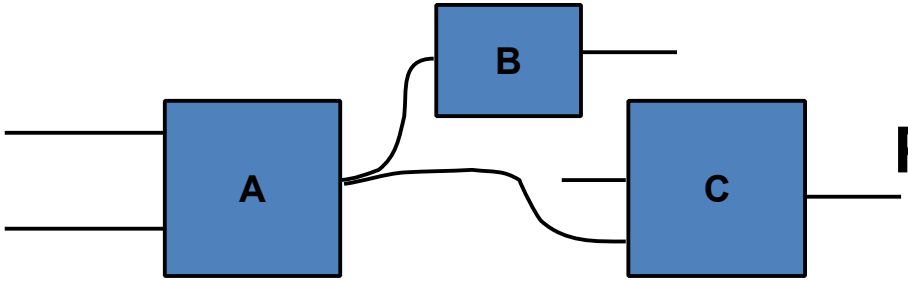
- Equilibrium: $\Sigma_j c_{ij} (x_j - x_i) = 0$ for all module i
- However, trivial solution: $x_j = x_i$ for all i, j. Everything placed on the same position!
- Need to have some way to avoid overlapping
- A method to avoid overlapping:
  - Add some repulsive force which is inversely proportional to distance (or distance squared)
- Solution of force equations correspond to the minimum potential energy of system
  -

$$PE = \sum_{i=1}^{n} [(F_x^i)^2 + (F_y^i)^2]$$

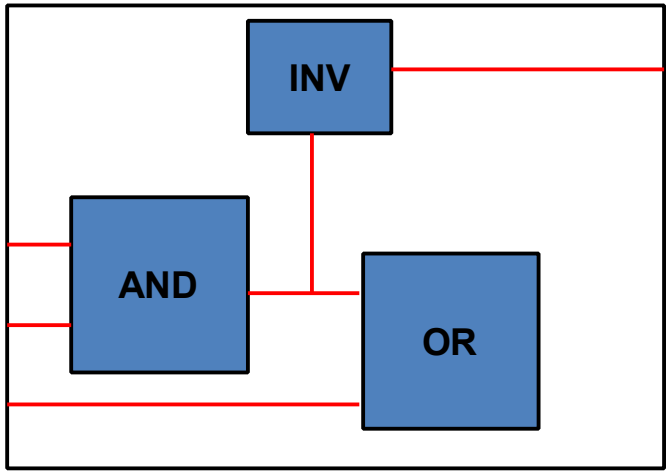
Comments on Force-Directed Placement

- ✓ Use directions of forces to guide the search
- ✓ Usually much faster than simulated annealing
- x Focus on connections, not shapes of blocks
- x Only a heuristic; an equilibrium configuration does not necessarily give a good placement
- ? Successful or not depends on the way to eliminate overlapping

Routing in design flow



Post Placed Netlist



Floorplan/Placement

Routing

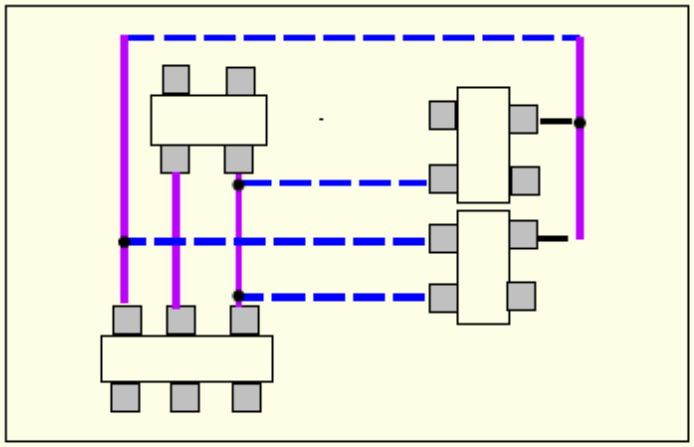
Process of finding geometric layouts of the net

The Routing Problem

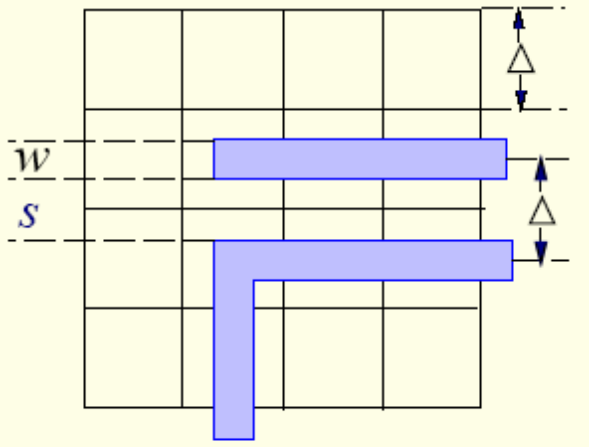
- Apply it after Placement
- Input:
  - Netlist
  - Timing budget for, typically, critical nets
  - Locations of blocks and locations of pins
- Output:
  - Geometric layouts of all nets
- Objective:
  - Minimize the total wire length, the number of vias, or just completing all connections without increasing the chip area.
  - Each net meets its timing budget.

The Routing Constraints

- Examples:
  - Placement constraint
  - Number of routing layers
  - Delay constraint
  - Meet all geometrical constraints (design rules)
  - Physical/Electrical/Manufacturing constraints:
    - Crosstalk



Two−layer routing



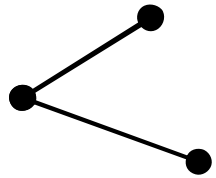
Geometrical constraint

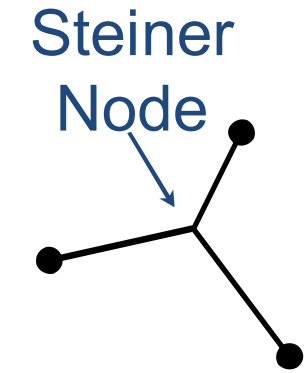
Steiner Tree

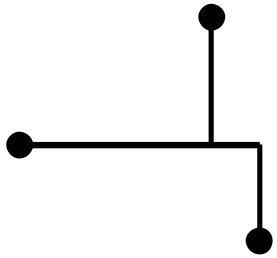
- For a multi-terminal net, we can construct a spanning tree to connect all the terminals together.
- But the wire length will be large.
- Better use Steiner Tree:

  A tree connecting all terminals and some additional nodes (Steiner nodes).
- Rectilinear Steiner Tree:

  Steiner tree in which all the edges run horizontally and vertically.







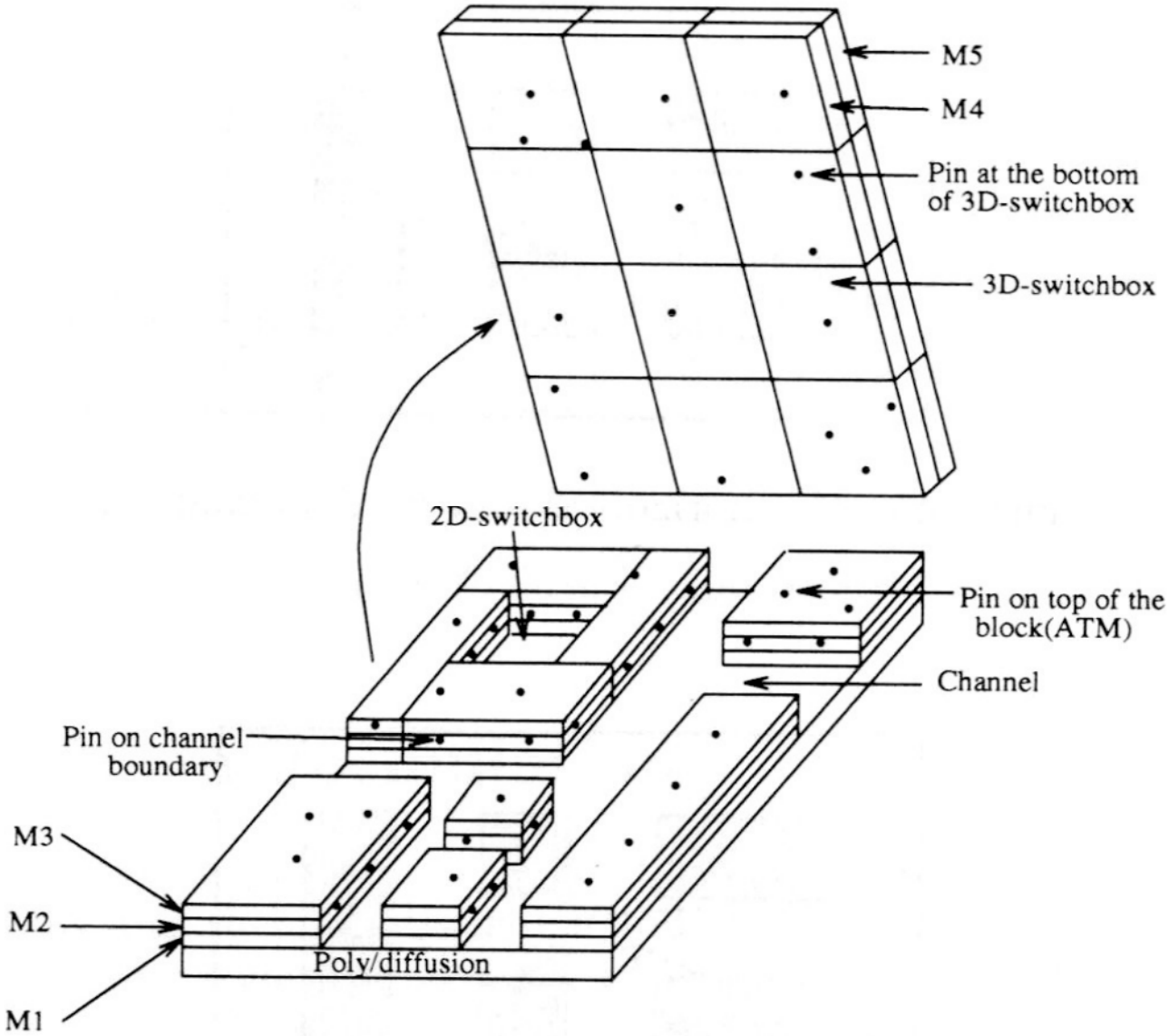
Routing Problem is Very Hard

- Minimum Steiner Tree Problem:
  - Given a net, find the Steiner tree with the minimum length.
  - Input :An edge weighted graph G=(V,E) and a subset D (demand points)
  - Output: A subset of vertices V'(such that D is covered) and induces a tree of minimum cost over all such trees
  - This problem is NP-Complete!

Kinds of Routing

- Global Routing
- Detailed Routing
  - Channel
  - Switchbox
- Others:
  - Maze routing
  - Over the cell routing
  - Clock routing

Routing Regions



Global Routing

Global routing is divided into 3 phases:

1. Region definition
2. Region assignment
3. Pin assignment to routing regions

Detailed routing

- Global routing do not define wires
- They define routing regions
- Detailed router places actual wires within regions, indicated by the global router
- We consider the channel routing problem here…

THANK YOU