CAL POLY
SAN LUIS OBISPO

EE 531: ADVANCED VLSI DESIGN

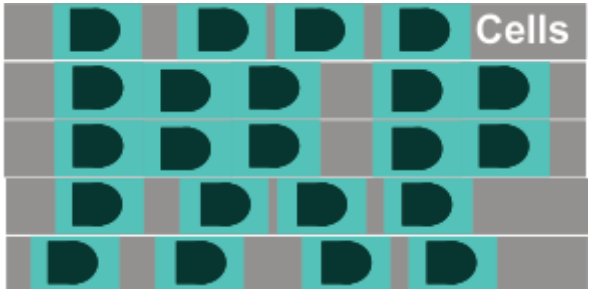
Placement

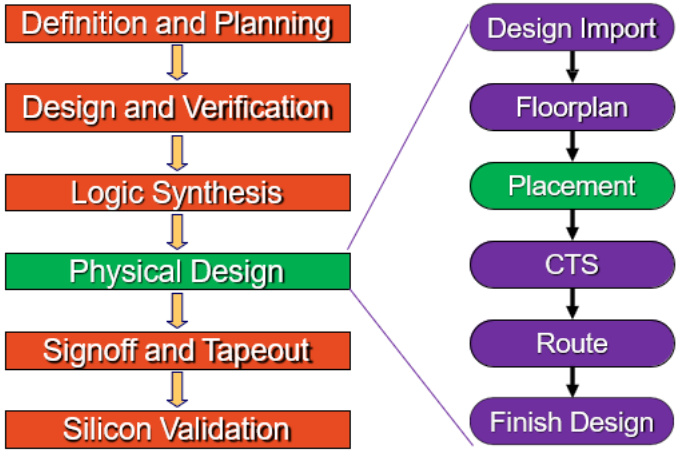
Nishith N. Chakraborty

February, 2025

DESIGN FLOW: OUR PROGRESS

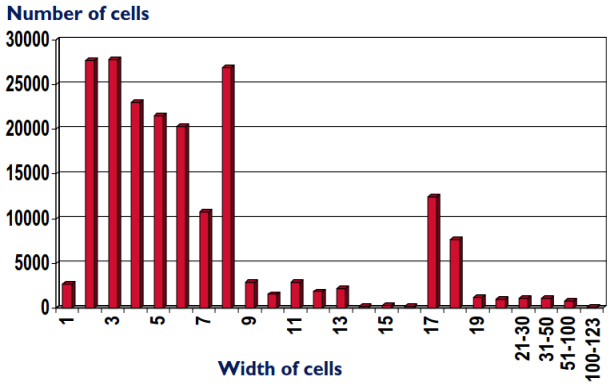
- We have reviewed the synthesis of our design into a technology mapped gate-level netlist.
- We have seen how to design a floorplan with pre-placed blocks.
- Now we will move into detailed placement of the standard cells.





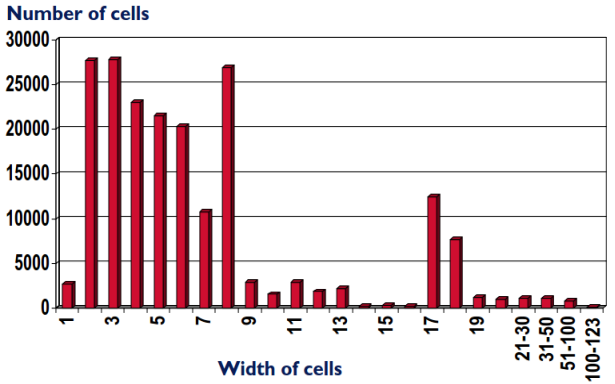
CHIP SIZE AND SMALL GATES

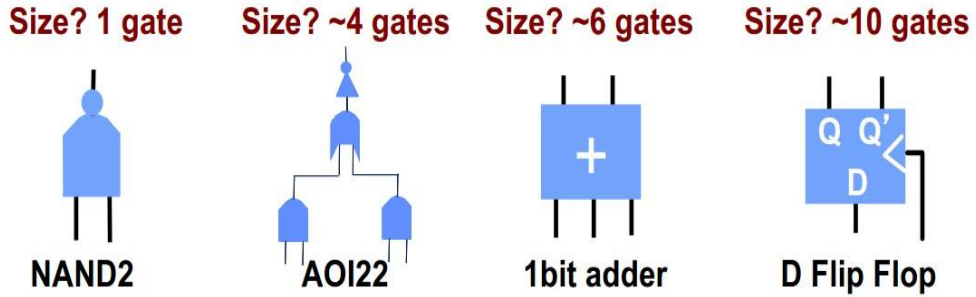
- Before we start discussing how to place the standard cells, we would like to make sure we understand how big a problem it is.
  - How big is a “100 Million gate ASIC”?
- In real designs, small cells dominate.
  - Therefore, numbers are usually given as “equivalent small gates”
  - May also consider a hard macro to be many, many small gates.



CHIP SIZE AND SMALL GATES

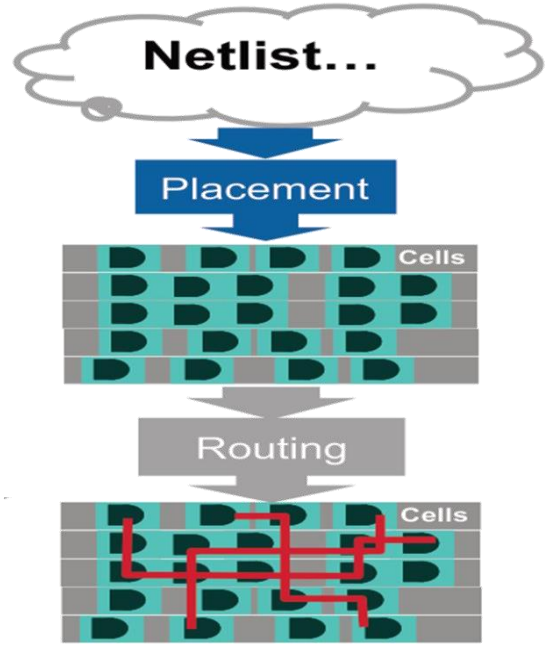
- Before we start discussing how to place the standard cells, we would like to make sure we understand how big a problem it is.
  - How big is a “100 Million gate ASIC”?
- In real designs, small cells dominate.
  - Therefore, numbers are usually given as “equivalent small gates”
  - May also consider a hard macro to be many, many small gates.
- Conclusion:
  - Gates==Equivalent NAND2 gates
  - Instances: # of things placed!
  - Rule of thumb: Instances=Gates/4..5





PLACEMENT

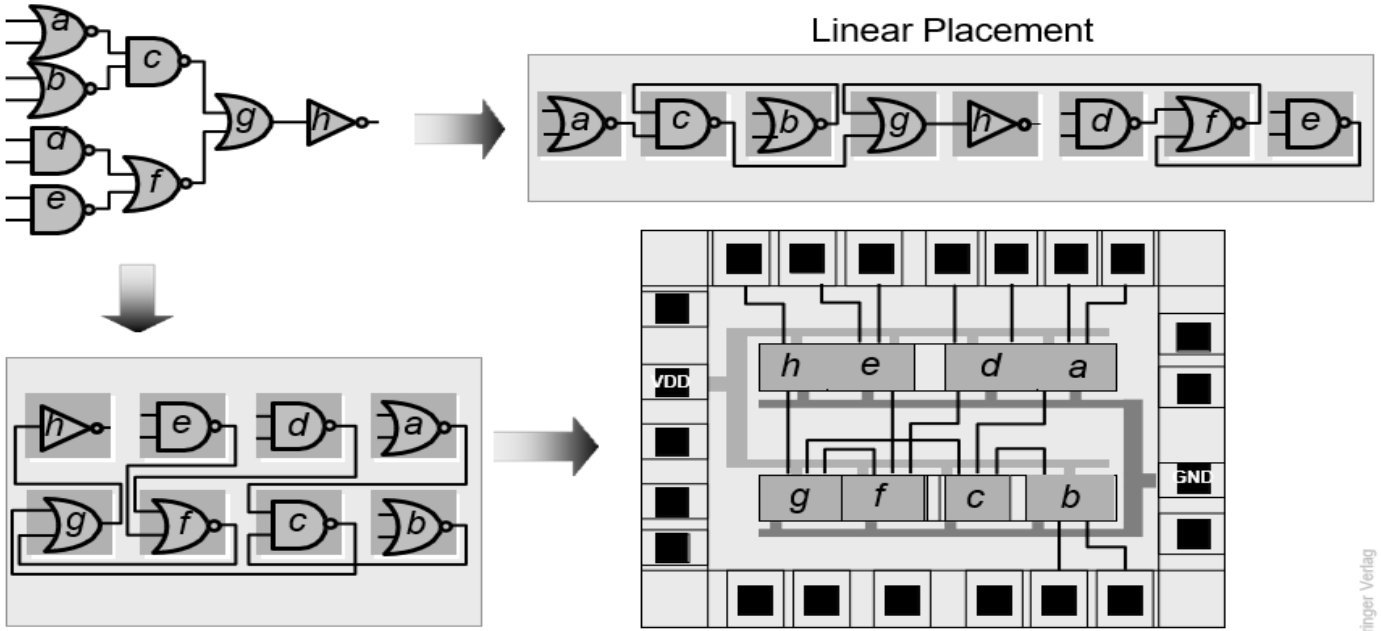
- Placement is the stage of the design flow, during which each instance (standard cell) is given an exact location.
- Inputs:
  - Netlist of gates and wires.
  - Floorplan and Technology constraints
- Output:
  - All cells located in the floorplan.
- Goal
  - Provide legal location of entire netlist
  - Enable detailed route of all nets
  - Meet timing, area, and power targets



PLACEMENT

- Key is use of constant height, variable-width standard cells arrayed into rows across the chip
- Can also add application-specific custom blocks
- Typically, there is no separation between std. cell rows as routing occurs over the cells using multiple metal layers
- Placement makes use of LEF input

PLACEMENT EXAMPLE



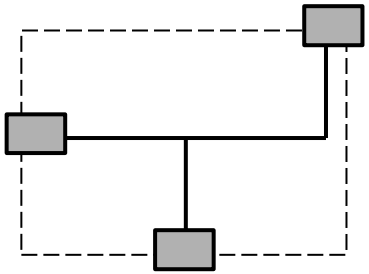
OPTIMIZATION OBJECTIVES

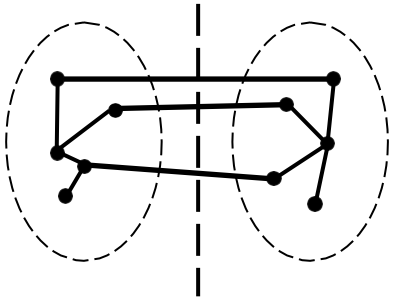
Total Wirelength

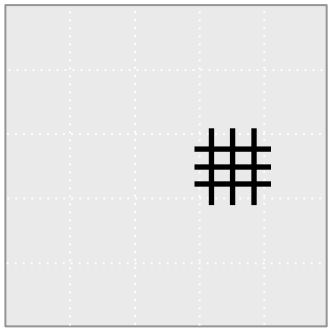
Number of Cut Nets

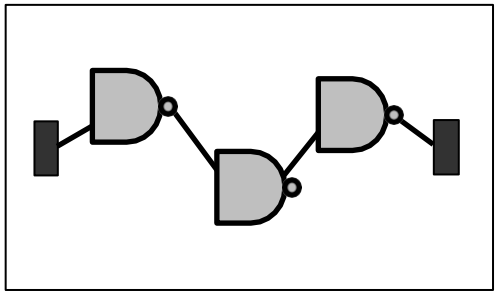
Wire Congestion

Signal Delay



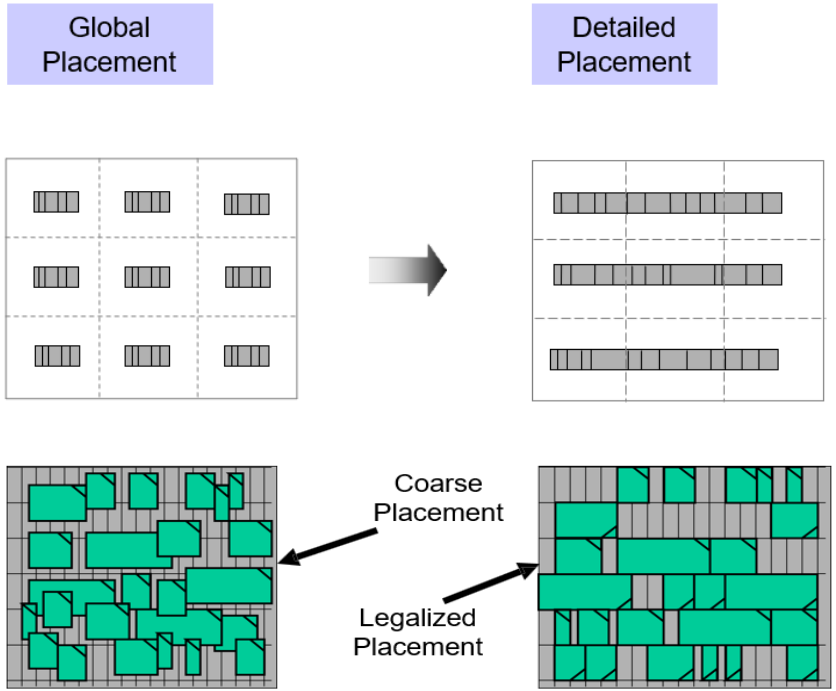




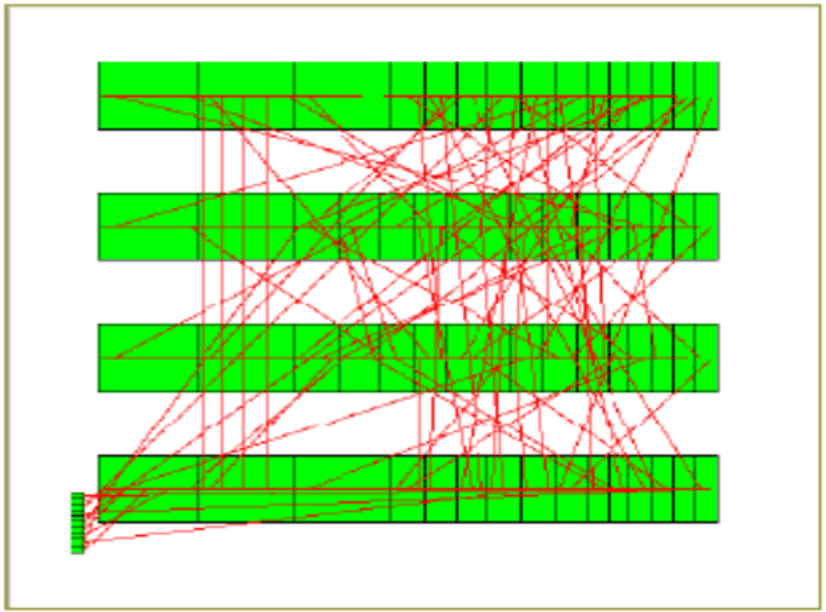


PLACEMENT FLOW

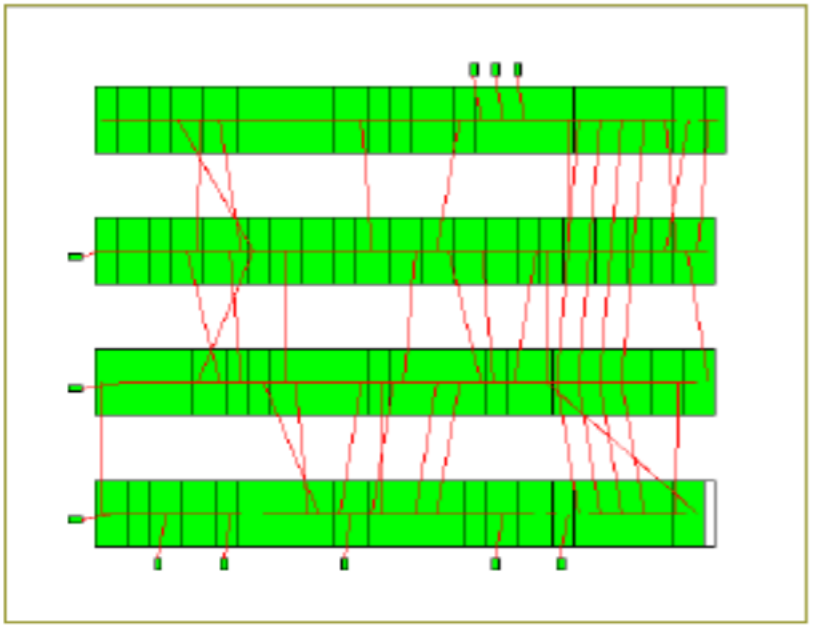
- In general, most tools partition the placement task into two stages:
  - Global placement:
    - Quickly divide each cell into “bins” to try and minimize the number of connections between groups.
  - Detailed placement:
    - Provide a legal placement for each instance
    - Try and minimize wirelength (or other cost metrics)
    - Try to finish with uncongested design.



RESULTS OF PLACEMENT



Bad Placement



Good Placement

RESULTS OF PLACEMENT

- Bad placement causes routing congestion:
  - Increases circuit area and cost
  - Longer wires leading to more capacitance
  - Longer delay and higher dynamic power
- Good placement:
  - Circuit area and wiring decreased
  - Shorter wires with less capacitance
  - Shorter delay and less dynamic power

PLACEMENT ALGORITHMS

- Partitioning-based algorithms:
  - The netlist and the layout are divided into smaller sub-netlists and sub-regions, respectively
  - Process is repeated until each sub-netlist and sub-region is small enough to be handled optimally
  - Detailed placement often performed by optimal solvers, facilitating a natural transition from global placement to detailed placement
  - Example: min-cut placement

PLACEMENT ALGORITHMS

- Stochastic algorithms:
  - Randomized moves that allow hill-climbing are used to optimize the cost function
  - Example: simulated annealing

- Analytic techniques:
  - Model the placement problem using an objective (cost) function, which can be optimized via numerical analysis
  - Examples: quadratic placement and force-directed placement

PLACEMENT ALGORITHMS

Partitioning-based

Analytic

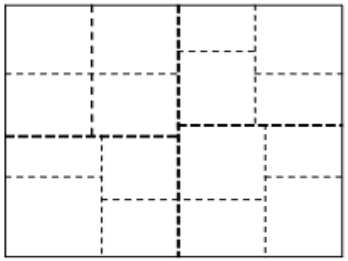
Stochastic

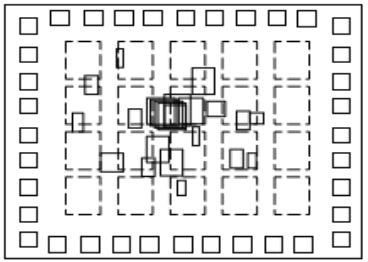
Min-cut placement

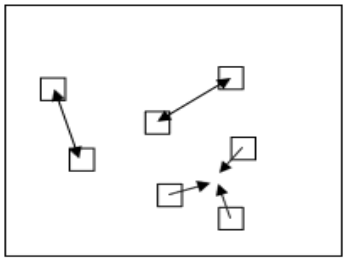
Quadratic placement

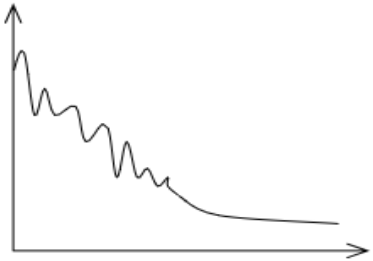
Force-directed placement

Simulated annealing







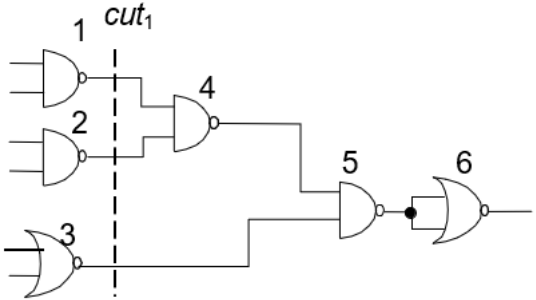


MIN-CUT ALGORITHM

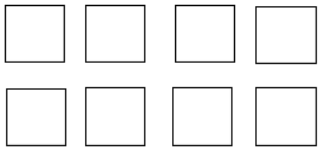
- Uses partitioning algorithms to divide (1) the netlist and (2) the layout region into smaller sub-netlists and sub-regions
- Conceptually, each sub-region is assigned a portion of the original netlist
- Each cut heuristically minimizes the number of cut nets using, for example,
  - Kernighan-Lin (KL) algorithm
  - Fiduccia-Mattheyses (FM) algorithm

MIN-CUT ALGORITHM: EXAMPLE

- Given:

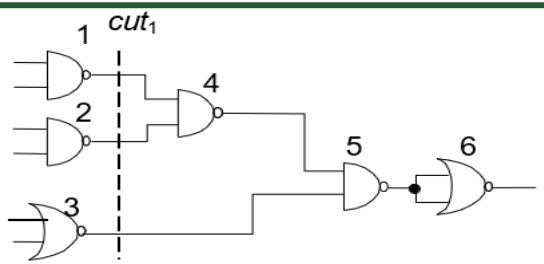


- Task: 4 x 2 placement with minimum wirelength using alternative cutline directions and the KL algorithm

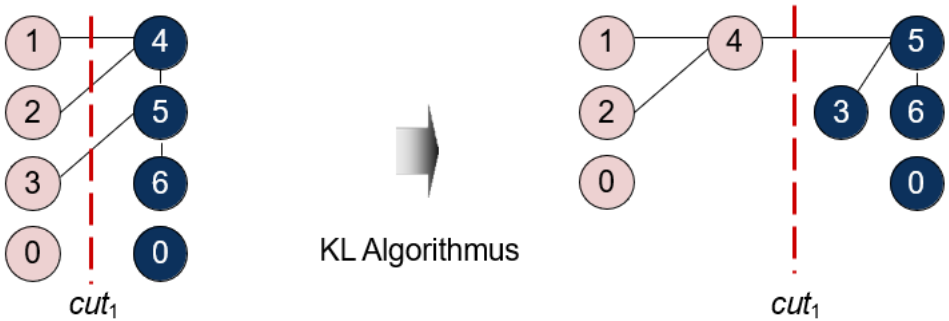


MIN-CUT ALGORITHM: EXAMPLE

- Given:

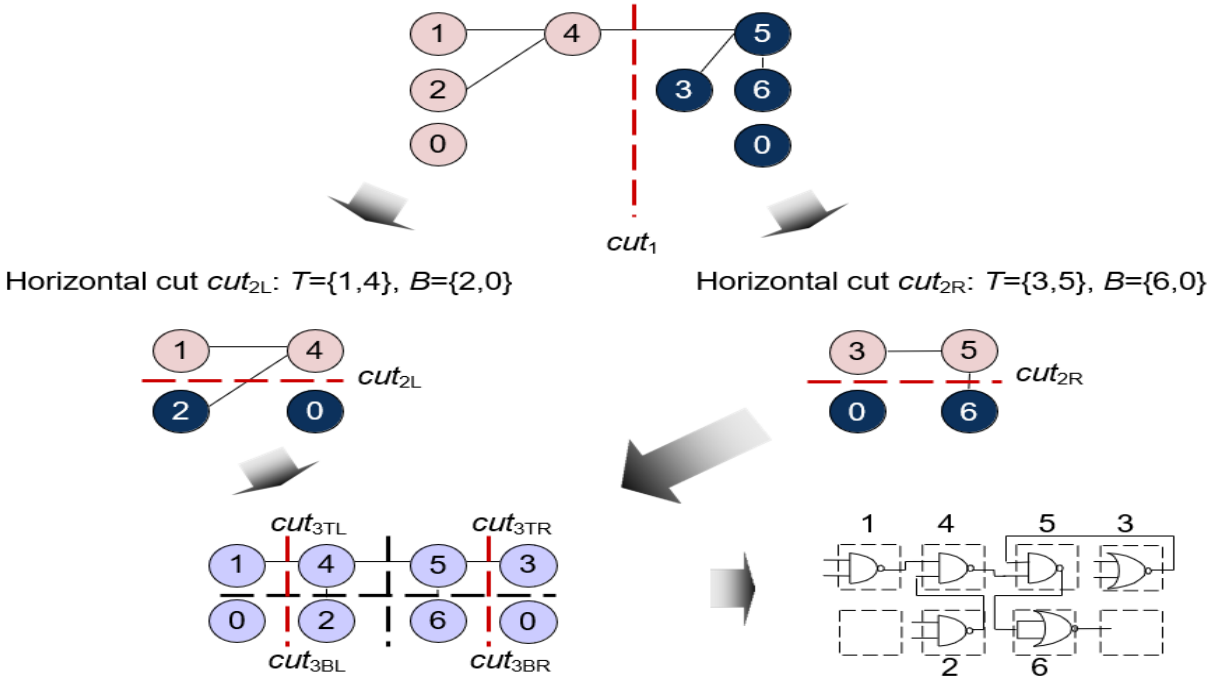


- Vertical cut cut_1: L={1,2,3}, R={4,5,6}
- Let's run KL algorithm



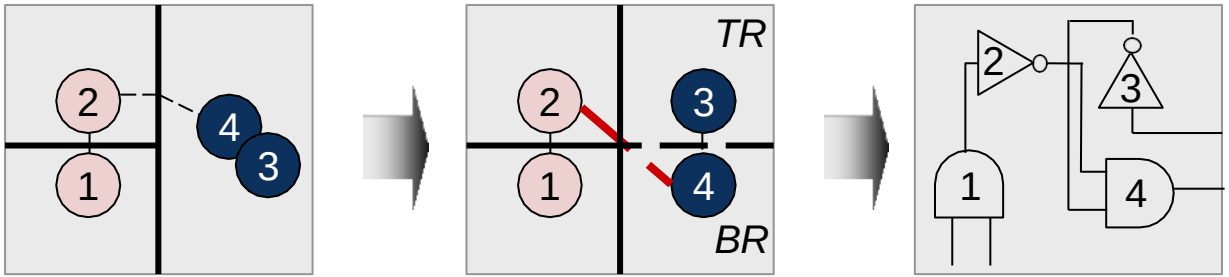
MIN-CUT ALGORITHM: EXAMPLE

- Now partition each partition:



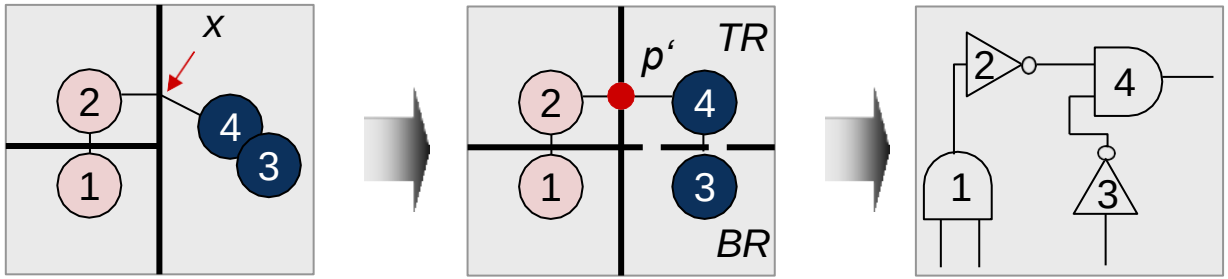
MIN-CUT ALGORITHM: TERMINAL PROPAGATION

- Min-cut didn't take external conditions into account



MIN-CUT ALGORITHM: TERMINAL PROPAGATION

- Min-cut didn't take external conditions into account
- Modified Min-cut to include terminal propagation
  - External connections are represented by artificial connection points on the cutline
  - Dummy nodes in hypergraphs



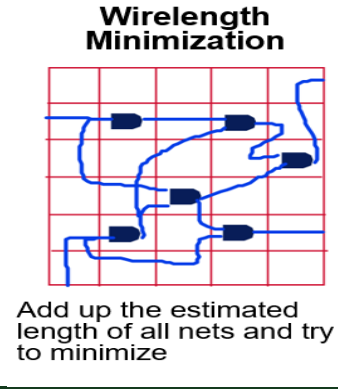
MIN-CUT ALGORITHM: PROS AND CONS

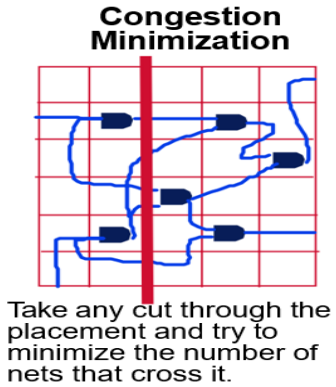
- Advantages:
  - Reasonably fast
  - Objective function can be adjusted, e.g., to perform timing-driven placement
  - Hierarchical strategy applicable to large circuits

- Disadvantages:
  - Randomized, chaotic algorithms – small changes in input lead to large changes in output
  - Optimizing one cutline at a time may result in routing congestion elsewhere

RANDOM PLACEMENT

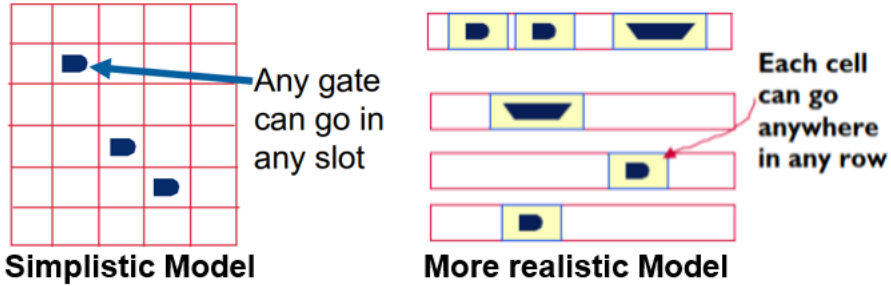
- Problem Formulation: Given a netlist, and fixed-shape cells (small, standard cell), find the exact location of the cells to minimize area and wire-length
  - Consistent with the standard-cell design methodology
    - Row-based, no hard-macros
  - Modules
    - Usually fixed, equal height (exception: double height cells)
    - Some fixed (I/O pads)
    - Connected by edges or hyperedges
- Objectives:
  - Cost components: area, wire length
  - Additional cost components: timing, congestion





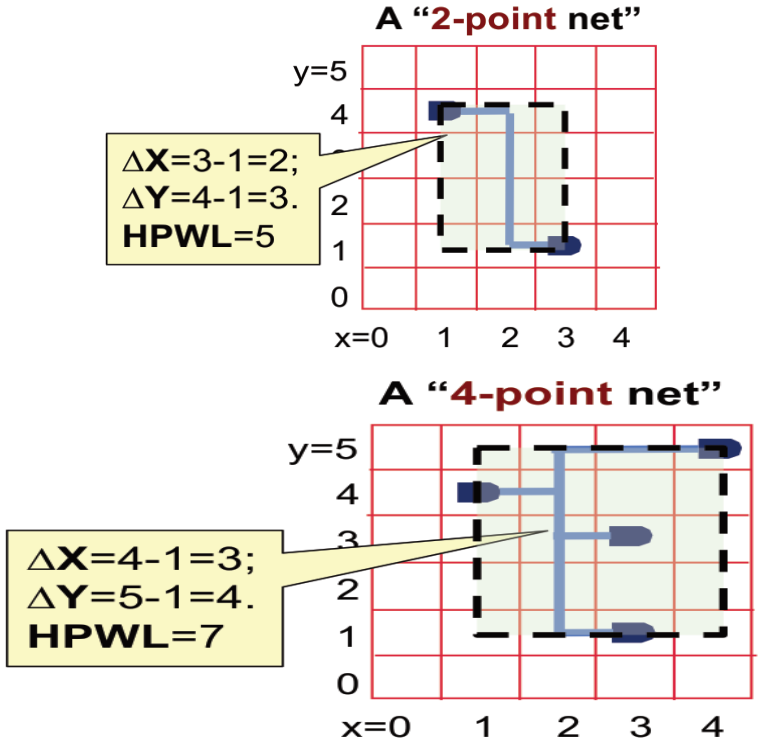
SIMPLE PLACER

- Assume a very simple chip model:
  - Simple grid – cells go in squares
  - Pins fixed at edges
- Assume simple gate model:
  - All gates same size
  - Each grid slot can hold one gate.



SIMPLE PLACER

- Assume a very simple chip model:
  - Simple grid – cells go in squares
  - Pins fixed at edges
- Assume simple gate model:
  - All gates same size
  - Each grid slot can hold one gate.
- Simple bounding box wirelength estimator:
  - "Half-Perimeter Wirelength" (HPWL)
  - Put the smallest box around all gates on net.
  - Measure Width (ΔX) and Height (ΔY) of box.
  - HPWL= ΔX+ΔY



SIMPLE PLACER

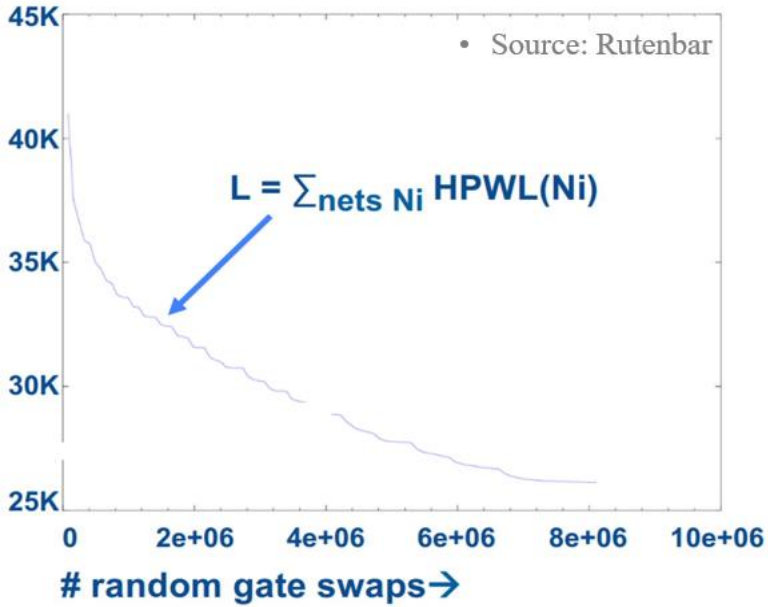
Let's define a simple algorithm:

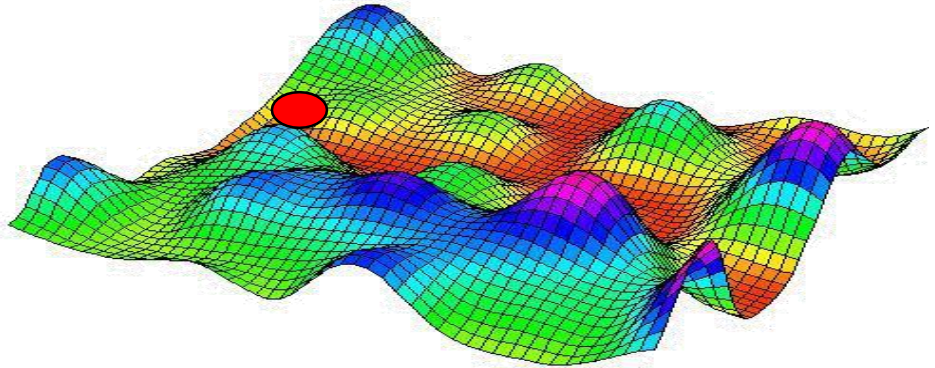
- Start with a random placement:
  - Randomly assign each gate to a grid location.
- Apply random iterative improvement:
  - Pick a random pair of gates.
  - Swap their locations and evaluate the change in total wirelength.
  - If wirelength got smaller – accept the swap.
  - If wirelength got bigger – undo the swap.
  - Repeat until wirelength stops improving.

```
// Random initial placement
foreach (gate Gi in netlist)
  place Gi in random (x,y) not occupied.

// calculate initial HPWL
L=0
foreach (net Ni in netlist)
  L = L + HPWL(Ni)

// random iterative improvement
while (overall HPWL is improving)
  pick two random gates Gi, Gj
  swap Gi and Gj
  evaluate ΔL = new HPWL - old HPWL
  if (ΔL < 0) then keep the swap
  if (ΔL > 0) undo the swap
```

SIMPLE PLACER

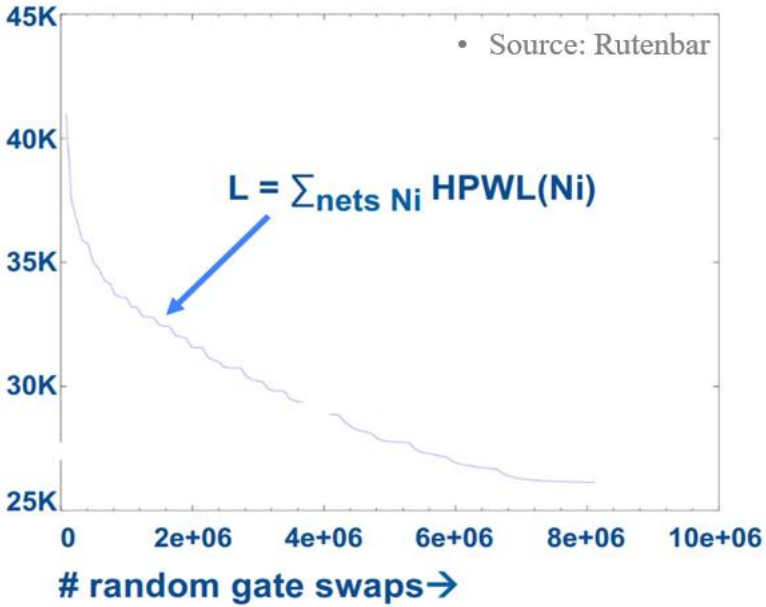
- Was this any good?
  - Well, we quickly get stuck in a local minimum.

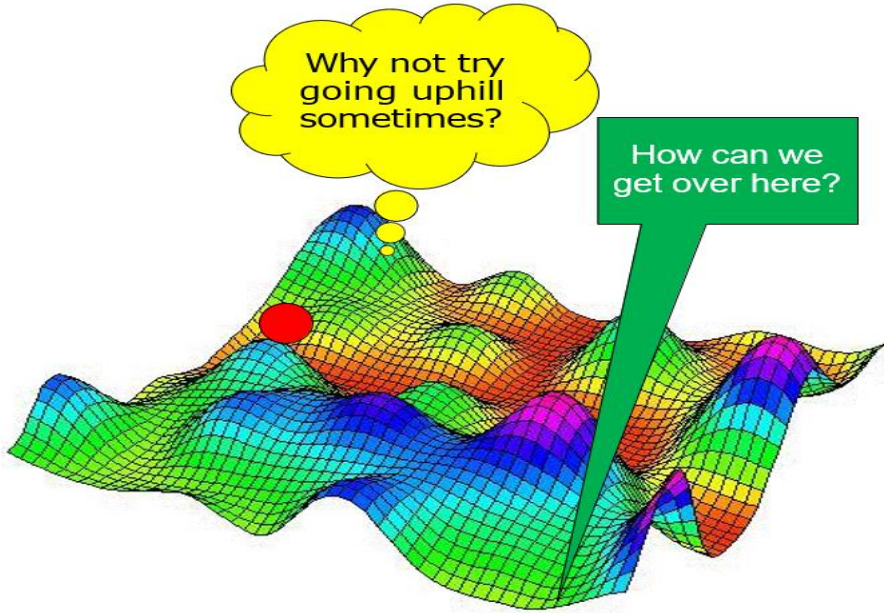




SIMPLE PLACER

- Was this any good?
  - Well, we quickly get stuck in a local minimum.

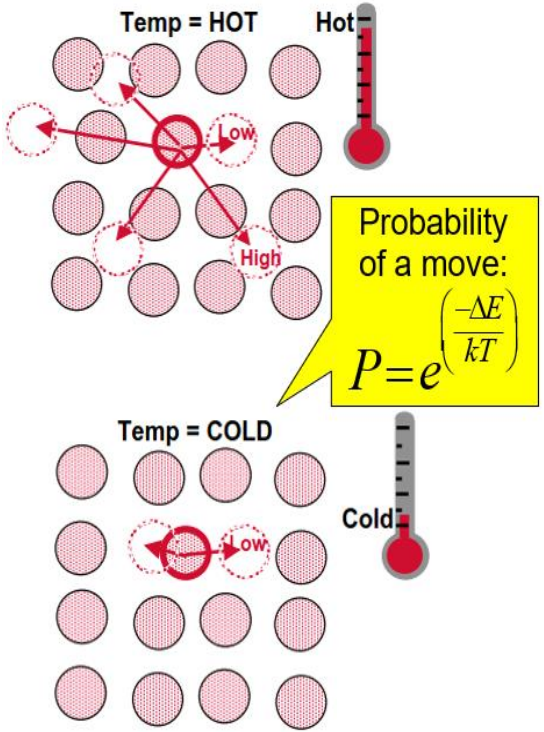




SIMULATED ANNEALING

In semiconductors, there is "annealing":

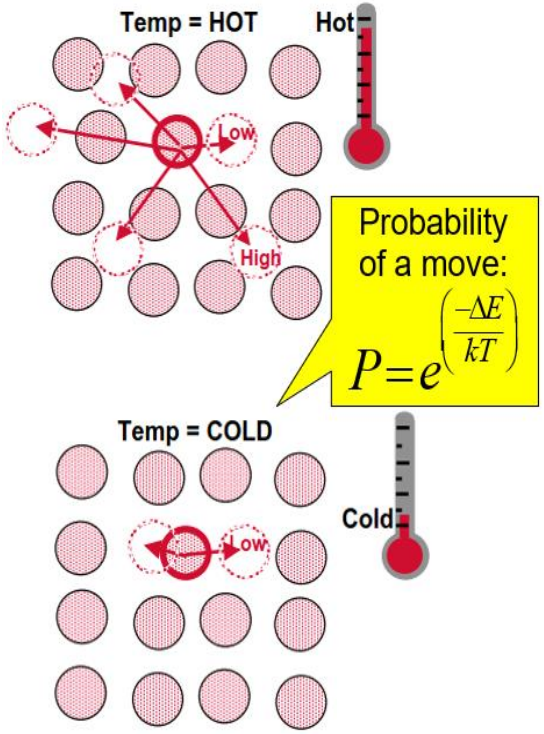
- The lowest energy state of a crystal lattice is when all atoms are lined up.
- So if we have a messy crystal:
  - Heat it up – give atoms energy to move around.
  - Cool it slowly –
    - At first, atoms will move a lot.
    - But as the crystal cools, the movement will be restricted.



SIMULATED ANNEALING

In semiconductors, there is "annealing":

- The lowest energy state of a crystal lattice is when all atoms are lined up.
- So if we have a messy crystal:
  - Heat it up – give atoms energy to move around.
  - Cool it slowly –
    - At first, atoms will move a lot.
    - But as the crystal cools, the movement will be restricted.
- What if we apply this idea to random hill climbing?
  - At first ("hot temperature") we'll "climb a lot of hills".
  - As we progress ("cool down") we will make fewer big jumps.
- This very famous idea is called "Simulated Annealing"



SIMULATED ANNEALING ALGORITHM

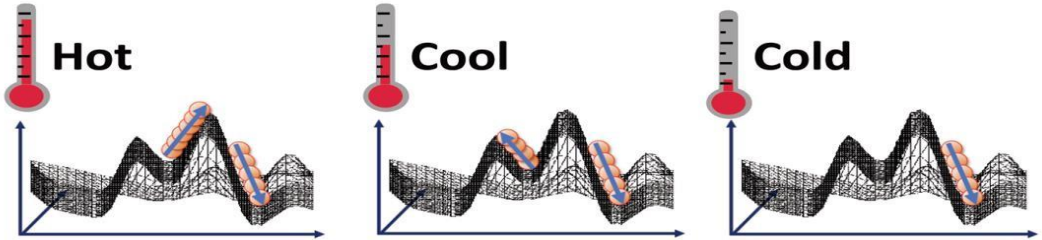
- Start with the same basic algorithm:
  - Random initial placement
  - Swap two Random gates
  - Evaluate change in HPWL (ΔL)
  - If wirelength improves, accept the change.

```
T=HOT; frozen = false
while (!frozen)
  swap two random gates Gi, Gj
  evaluate ΔL
  if (ΔL < 0) then
    keep the swap
  else
    if (random() < exp(-ΔL/T))
      accept swap
    else
      undo swap
  if (HPWL still decreasing)
    T = 0.9*T
  else
    frozen=true
```

SIMULATED ANNEALING ALGORITHM

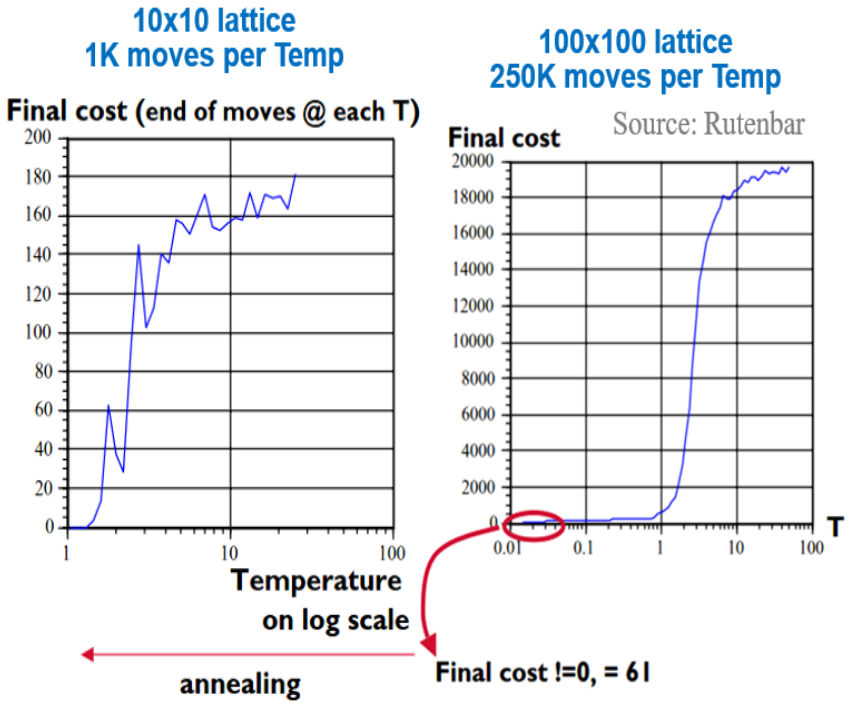
- Start with the same basic algorithm:
  - Random initial placement
  - Swap two Random gates
  - Evaluate change in HPWL (ΔL)
  - If wirelength improves, accept the change.
- But what if wirelength increases (ΔL>0)?
  - Evaluate annealing probability function: $P = \exp\left(\frac{-\Delta L}{T}\right)$
  - Choose a uniform random number (*R*) between 0 and 1
  - If R<P then keep the swap.
- What is T?
  - *T* is the simulated temperature.
  - Start hot. Cool down.

```
T=HOT; frozen = false
while (!frozen)
  swap two random gates Gi, Gj
  evaluate ΔL
  if (ΔL < 0) then
    keep the swap
  else
    if (random() < exp(-ΔL/T))
      accept swap
    else
      undo swap
  if (HPWL still decreasing)
    T = 0.9*T
  else
    frozen=true
```



SIMULATED ANNEALING

- How well does this work?
  - Really well!
  - Many EDA algorithms use Simulated Annealing.
- Does it find an optimal solution?
  - No. But it's good at avoiding local minima.
- What happens if I run it again?
  - I will get a different answer each time!
- NOT how placers work today!



ANALYTICAL PLACEMENT APPROACH

- Question:
  - Can we write an equation whose minimum is the placement?
  - If we have a cost function (such as wirelength) that is a function of the gate coordinates (x_i, y_i), i.e.: $L_{\text{wire}} = f(x_1, x_2, \ldots, x_N, y_1, y_2, \ldots, y_N)$
  - Then we could find the minimum of f and this would be our optimal placement!

ANALYTICAL PLACEMENT APPROACH

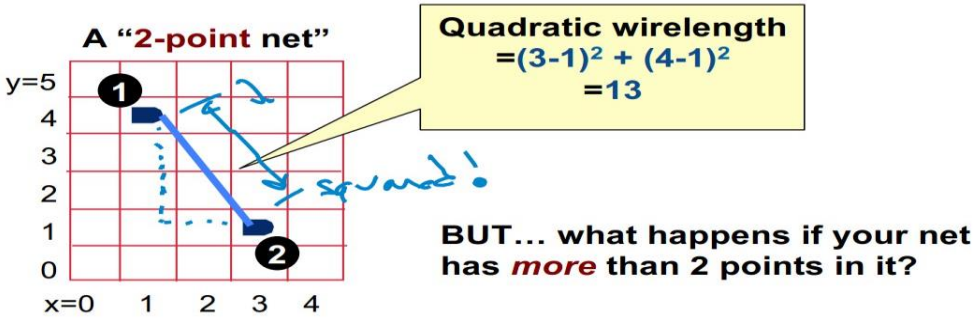
- Question:
  - Can we write an equation whose minimum is the placement?
  - If we have a cost function (such as wirelength) that is a function of the gate coordinates (x_i, y_i), i.e.: $L_{\text{wire}} = f(x_1, x_2, \ldots, x_N, y_1, y_2, \ldots, y_N)$
  - Then we could find the minimum of f and this would be our optimal placement!

- Sounds crazy, but the answer is YES!
  - All modern placers are based on analytical placement.
  - We need to write the cost function in a mathematically friendly way.
  - Then, we can just differentiate and equate to 0!

$$\frac{\partial f}{\partial x} = 0, \frac{\partial f}{\partial y} = 0$$

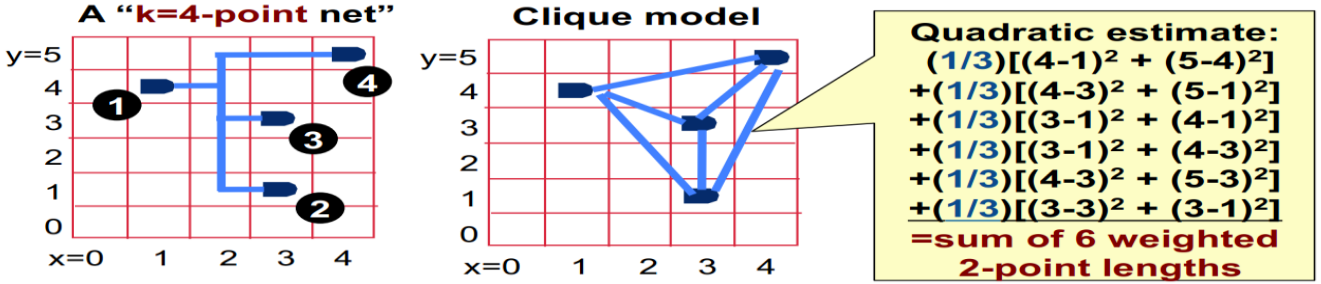
QUADRATIC PLACEMENT COST FUNCTION

- Instead of HPWL, let's define a new wirelength model:
  - Quadratic wirelength: $L = (x_1 - x_2)^2 + (y_1 - y_2)^2$



QUADRATIC PLACEMENT COST FUNCTION

- What about a k-point net (k>2)?
  - Instead of one "real" net, replace with a fully connected clique model.
  - So each gate on the net has a one-to-one connection with the other.
  - Altogether k(k-1)/2 nets
  - Compensate by weighting each new net by 1/(k-1)
- One last point:
  - Assume that gates are dimensionless points.

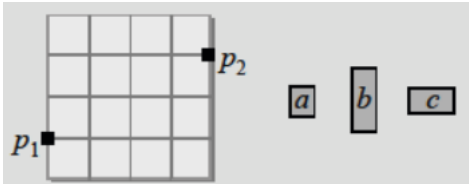


QUADRATIC PLACEMENT CALCULATION

- Example of quadratic wirelength calculation:
  - Each point has an unknown (x_i, y_i) coordinate.
  - Each net has a pre-assigned weight.

$$L(P) = \sum_{i=1, j=1}^{n} c(i,j)\left((x_i - x_j)^2 + (y_i - y_j)^2\right)$$

  - Pads are fixed pins on the edge.



QUADRATIC PLACEMENT CALCULATION

- Example of quadratic wirelength calculation:
  - Each point has an unknown (x_i, y_i) coordinate.
  - Each net has a pre-assigned weight.

$$L(P) = \sum_{i=1, j=1}^{n} c(i,j)\left((x_i - x_j)^2 + (y_i - y_j)^2\right)$$

  - Pads are fixed pins on the edge.
  - Each dimension can be considered independently:

$$L_x(P) = \sum_{i=1, j=1}^{n} c(i,j)(x_i - x_j)^2 \qquad L_y(P) = \sum_{i=1, j=1}^{n} c(i,j)(y_i - y_j)^2$$

  - Convex quadratic optimization problem: any local minimum solution is also a global minimum
  - Optimal x- and y--coordinates can be found by setting the partial derivatives of $L_x(P)$ and $L_y(P)$ to zero

QUADRATIC PLACEMENT CALCULATION

$$L_x(P) = \sum_{i=1,j=1}^{n} c(i,j)(x_i - x_j)^2 \qquad L_y(P) = \sum_{i=1,j=1}^{n} c(i,j)(y_i - y_j)^2$$

$$\frac{\partial L_x(P)}{\partial X} = AX - b_x = 0 \qquad \frac{\partial L_y(P)}{\partial Y} = AY - b_y = 0$$

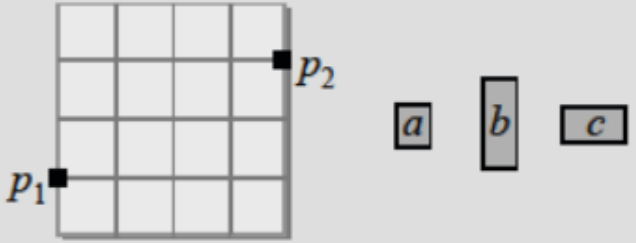
- Where A is a matrix with $A[i][j] = -c(i,j)$ when $i \neq j$, and $A[i][i]$ = the sum of incident connection weights of cell i.
- X is a vector of all the x-coordinates of the non-fixed cells, and b_x is a vector with $b_x[i]$ = the sum of x-coordinates of all fixed cells attached to i.
- Y is a vector of all the y-coordinates of the non-fixed cells, and b_y is a vector with $b_y[i]$ = the sum of y-coordinates of all fixed cells attached to i.

QUADRATIC PLACEMENT EXAMPLE

Example: Quadratic Placement

Given: (1) placement P with two fixed points p_1 (100,175) and p_2 (200,225), (3) three free blocks a-c and (4) nets N_1-N_4.

N_1 (P_1,a) N_2 (a,b) N_3 (b,c) N_4 (c,P_2)



Task: find the coordinates of blocks (x_a, y_a), (x_b, y_b) and (x_c, y_c).

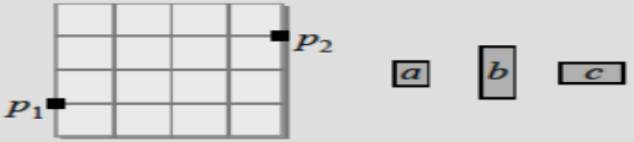
$C_{i,j} = 1$, as no weight specified.

QUADRATIC PLACEMENT EXAMPLE

Example: Quadratic Placement

Given: (1) placement P with two fixed points p_1 (100,175) and p_2 (200,225), (3) three free blocks a-c and (4) nets N_1-N_4.

N_1 (P_1,a) N_2 (a,b) N_3 (b,c) N_4 (c,P_2)



Task: find the coordinates of blocks (x_a, y_a), (x_b, y_b) and (x_c, y_c).

Solution:

Solve for x-coordinates.

$$L_x(P) = (100 - x_a)^2 + (x_a - x_b)^2 + (x_b - x_c)^2 + (x_c - 200)^2$$

$$\frac{\partial L_x(P)}{x_a} =$$

$$\frac{\partial L_x(P)}{x_b} =$$

$$\frac{\partial L_x(P)}{x_c} =$$

Solve for y-coordinates.

$$L_y(P) = (175 - y_a)^2 + (y_a - y_b)^2 + (y_b - y_c)^2 + (y_c - 225)^2$$

$$\frac{\partial L_y(P)}{y_a} = \cdot$$

$$\frac{\partial L_y(P)}{y_b} = \cdot$$

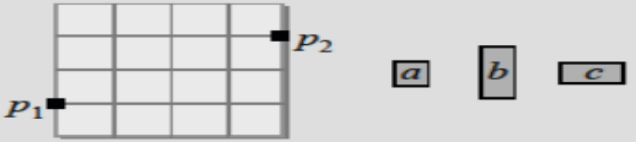
$$\frac{\partial L_y(P)}{y_c} = \cdot$$

QUADRATIC PLACEMENT EXAMPLE

Example: Quadratic Placement

Given: (1) placement P with two fixed points p_1 (100,175) and p_2 (200,225), (3) three free blocks a-c and (4) nets N_1-N_4.

N_1 (P_1,a) N_2 (a,b) N_3 (b,c) N_4 (c,P_2)



Task: find the coordinates of blocks (x_a, y_a), (x_b, y_b) and (x_c, y_c).

Solution:

Solve for x-coordinates.

$$L_x(P) = (100 - x_a)^2 + (x_a - x_b)^2 + (x_b - x_c)^2 + (x_c - 200)^2$$

$$\frac{\partial L_x(P)}{x_a} = -2(100 - x_a) + 2(x_a - x_b) = 4x_a - 2x_b - 200 = 0$$

$$\frac{\partial L_x(P)}{x_b} = -2(x_a - x_b) + 2(x_b - x_c) = -2x_a + 4x_b - 2x_c = 0$$

$$\frac{\partial L_x(P)}{x_c} = -2(x_b - x_c) + 2(x_c - 200) = -2x_b + 4x_c - 400 = 0$$

Solve for y-coordinates.

$$L_y(P) = (175 - y_a)^2 + (y_a - y_b)^2 + (y_b - y_c)^2 + (y_c - 225)^2$$

$$\frac{\partial L_y(P)}{y_a} = -2(175 - y_a) + 2(y_a - y_b) = 4y_a - 2y_b - 350 = 0$$

$$\frac{\partial L_y(P)}{y_b} = -2(y_a - y_b) + 2(y_b - y_c) = -2y_a + 4y_b - 2y_c = 0$$

$$\frac{\partial L_y(P)}{y_c} = -2(y_b - y_c) + 2(y_c - 225) = -2y_b + 4y_c - 450 = 0$$

QUADRATIC PLACEMENT EXAMPLE

Solution:

Solve for x-coordinates.

$$L_x(P) = (100 - x_a)^2 + (x_a - x_b)^2 + (x_b - x_c)^2 + (x_c - 200)^2$$

$$\frac{\partial L_x(P)}{x_a} = -2(100 - x_a) + 2(x_a - x_b) = 4x_a - 2x_b - 200 = 0$$

$$\frac{\partial L_x(P)}{x_b} = -2(x_a - x_b) + 2(x_b - x_c) = -2x_a + 4x_b - 2x_c = 0$$

$$\frac{\partial L_x(P)}{x_c} = -2(x_b - x_c) + 2(x_c - 200) = -2x_b + 4x_c - 400 = 0$$

Put in matrix form $AX = b_x$.

$$\begin{bmatrix} 4 & -2 & 0 \\ -2 & 4 & -2 \\ 0 & -2 & 4 \end{bmatrix} \begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} = \begin{bmatrix} 200 \\ 0 \\ 400 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} = \begin{bmatrix} 100 \\ 0 \\ 200 \end{bmatrix}$$

Solve for X: $x_a = 125, x_b = 150, x_c = 175$.

Solve for y-coordinates.

$$L_y(P) = (175 - y_a)^2 + (y_a - y_b)^2 + (y_b - y_c)^2 + (y_c - 225)^2$$

$$\frac{\partial L_y(P)}{y_a} = -2(175 - y_a) + 2(y_a - y_b) = 4y_a - 2y_b - 350 = 0$$

$$\frac{\partial L_y(P)}{y_b} = -2(y_a - y_b) + 2(y_b - y_c) = -2y_a + 4y_b - 2y_c = 0$$

$$\frac{\partial L_y(P)}{y_c} = -2(y_b - y_c) + 2(y_c - 225) = -2y_b + 4y_c - 450 = 0$$

Put in matrix form $AY = b_y$.

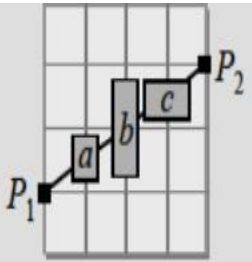
$$\begin{bmatrix} 4 & -2 & 0 \\ -2 & 4 & -2 \\ 0 & -2 & 4 \end{bmatrix} \begin{bmatrix} y_a \\ y_b \\ y_c \end{bmatrix} = \begin{bmatrix} 350 \\ 0 \\ 450 \end{bmatrix} \rightarrow \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} y_a \\ y_b \\ y_c \end{bmatrix} = \begin{bmatrix} 175 \\ 0 \\ 225 \end{bmatrix}$$

Solve for Y: $y_a = 187.5, y_b = 200, y_c = 212.5$.

QUADRATIC PLACEMENT EXAMPLE

- Final Solution:

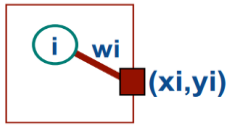
Final solution: a (125,187.5), b (150,200) and c (175,212.5).



BUILDING ANALYTICAL NETWORK

Recipe for success

- Build connectivity matrix (C) as follows.
  - C(i,j)=C(j,i)=w for a net with weight w connected between gates i and j.
  - If no net connects gates i and j, C(i,j)=0.
- Build A matrix:
  - Off diagonal, A(i,j)= −C(i,j)
  - A(i,i) is sum of row i + weight of net from i to pad.
- Build b vectors:
  - If gate i connects to a pad at (x_i,y_i) with weight w_i then $b_x(i)=w_i*x_i$, $b_y(i)=w_i*y_i$.



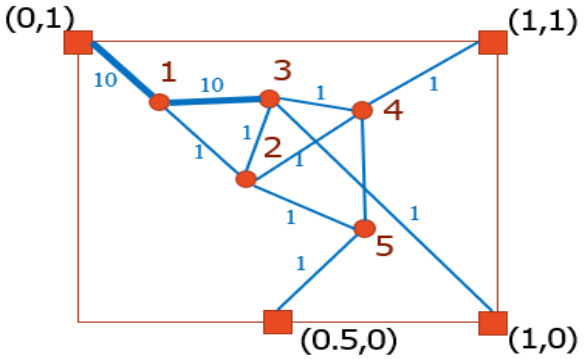
$$C_{i,j} = C_{j,i} = \begin{cases} 0 & i = j \\ 0 & \text{no net}_{i,j} \\ w_{i,j} & \text{net}_{i,j} \end{cases}$$

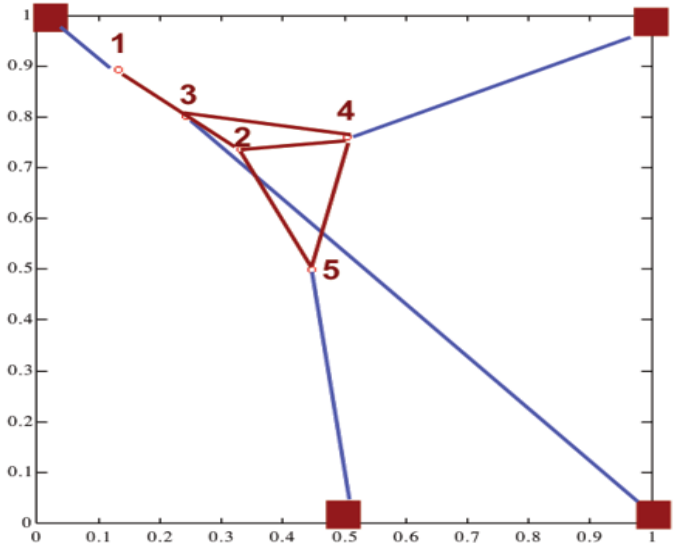
$$A_{i,j} = \begin{cases} -C_{i,j} & i \neq j \\ \sum_{j=1}^{N} C_{i,j} + w_{j,\text{pads}} & i = j \end{cases}$$

$$b_{x,i} = \begin{cases} 0 & \text{no pad} \\ w_i \cdot x_i & \text{pad } (x_i, y_i, w_i) \end{cases}$$

$$b_{y,i} = \begin{cases} 0 & \text{no pad} \\ w_i \cdot y_i & \text{pad } (x_i, y_i, w_i) \end{cases}$$

FIVE GATE EXAMPLE

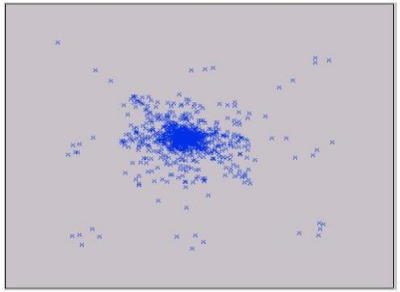


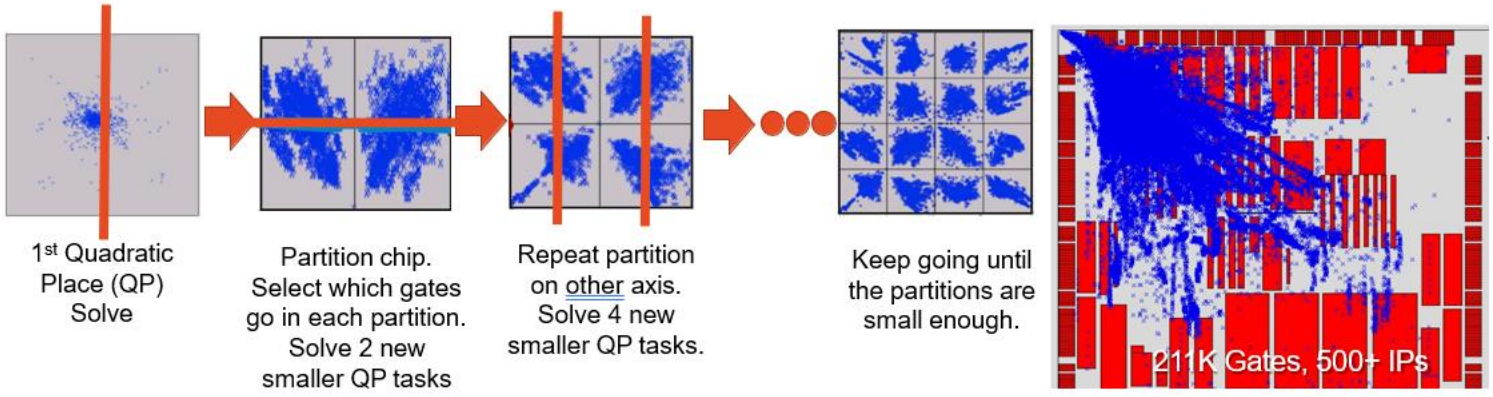


$$C = \begin{pmatrix} 0 & 1 & 10 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 10 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad A = \begin{pmatrix} 21 & -1 & -10 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -10 & -1 & 13 & -1 & 0 \\ 0 & -1 & -1 & 4 & -1 \\ 0 & -1 & 0 & -1 & 3 \end{pmatrix} \quad b_x = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0.5 \end{pmatrix} \quad b_y = \begin{pmatrix} 10 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

PROBLEM – GATE CLUSTERING

- What does a real quadratic placement look like?
  - All the gates want to be in the same place!
- How can we solve this?
  - Recursive Partitioning!

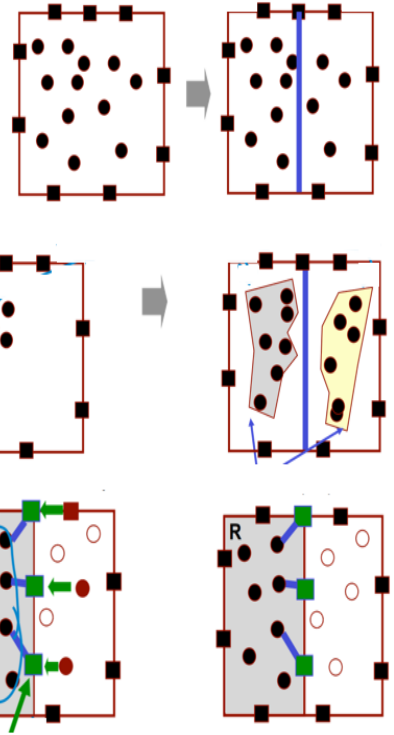




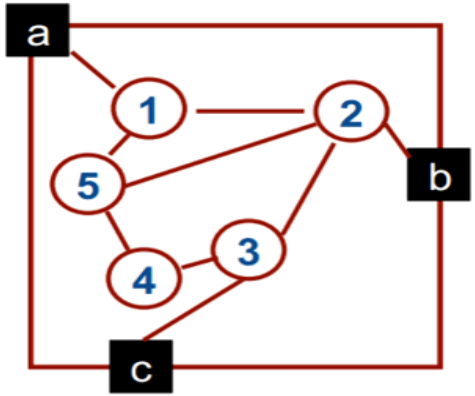
Source: Rutenbar, IBM

RECURSIVE PARTITIONING

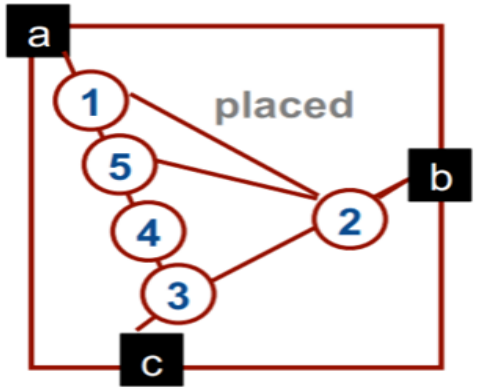
- Partition
  - Divide the chip into new, smaller placement tasks.
  - Divide it in half!
- Assignment
  - Assign gates into new, smaller region.
  - Sort the gates and distribute to each half.
- Containment
  - Formulate new QP matrix that keeps gates in new regions.
  - Create "pseudo pads" –
    - Every gate and pad NOT inside the partition R is modeled as a pad on the boundary of R.
    - Propagate the pseudo pads to their nearest point on R.



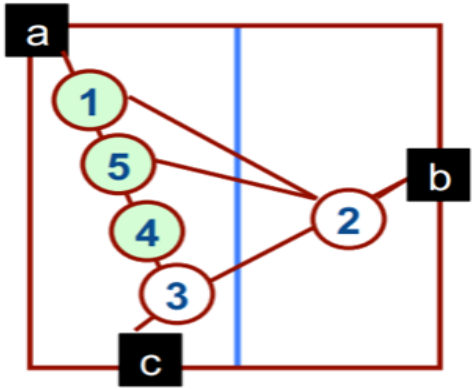
RECURSIVE PARTITIONING EXAMPLE



1. Initial netlist
5 gates (1,2,3,4,5)
9 wires
3 pads (a,b,c)

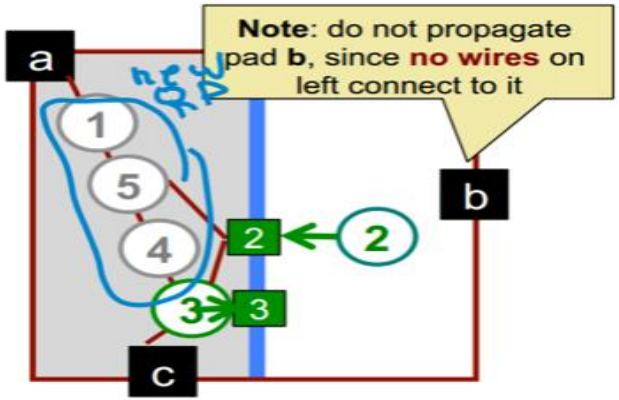


2. Initial QP

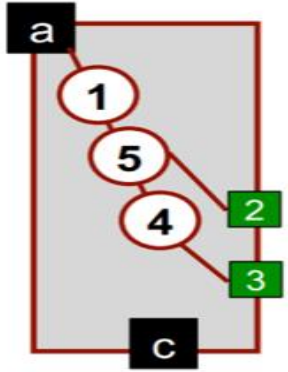


3. First partition
Sort on **X**:
Gate order 1 5 4 3 2
Pick: **1 5 4** on left

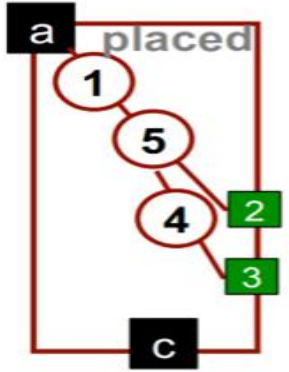
RECURSIVE PARTITIONING EXAMPLE



4. Propagate gates/pads
Right-side gates: 2,3
Right-side pads: b
Push to cut, using
y coordinates

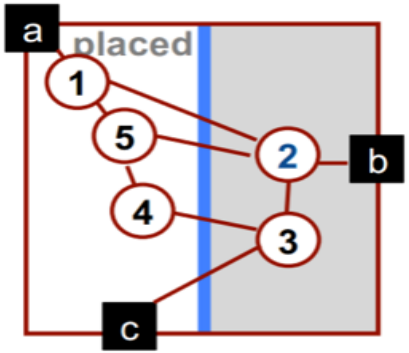


5. 2nd QP input
This is set up for this new smaller placement

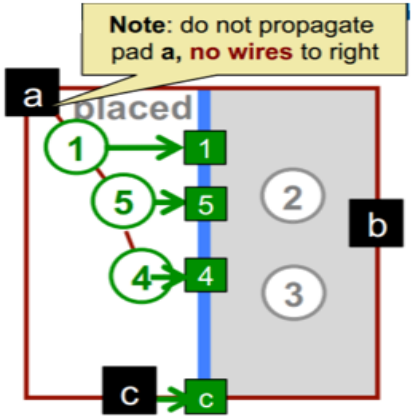


6. 2nd QP solved
New placement

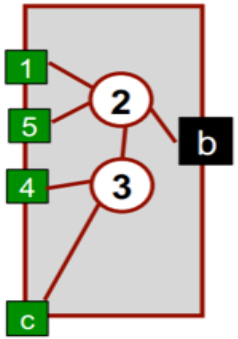
RECURSIVE PARTITIONING EXAMPLE



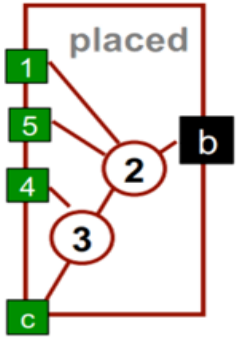
7. Left side placed.
Now, re-place right-side gates.



8. Propagate gates/pads
This is set up for next, new smaller placement



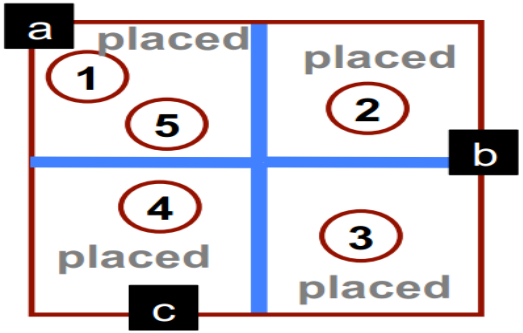
9. 3nd QP input
This is set up for this new smaller placement

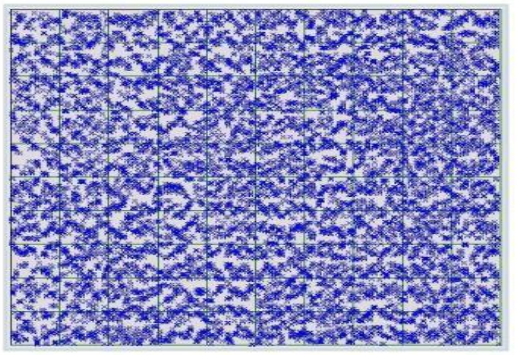


10. 3nd QP solve

PLACEMENT LEGALIZATION

- Keep recursively partitioning…
  - Usually, continue until you have a “small” number of gates in each region (i.e., 10-100 gates)
  - In these regions we will still have overlaps
- Still need to force gates in precise rows for final result
  - This is known as “legalization”
  - One easy way to do this is simulated annealing!
  - Just use a low temperature to start with, so you don’t make drastic moves.

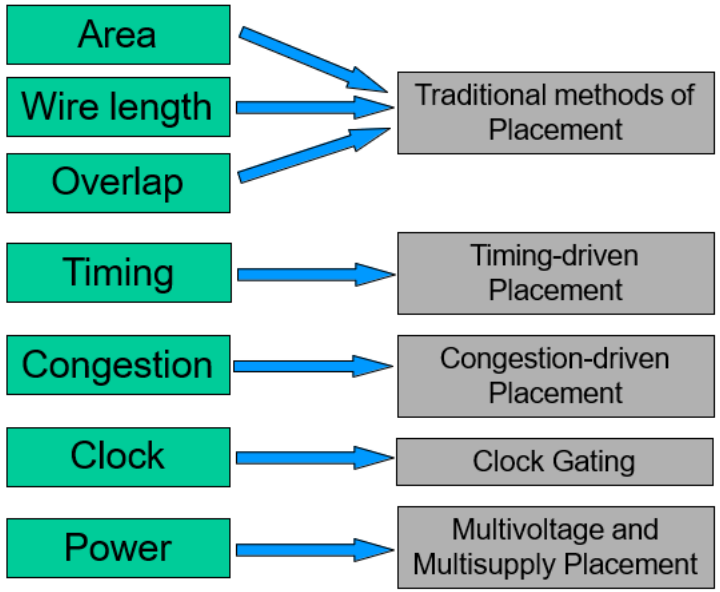




Placement in Practice

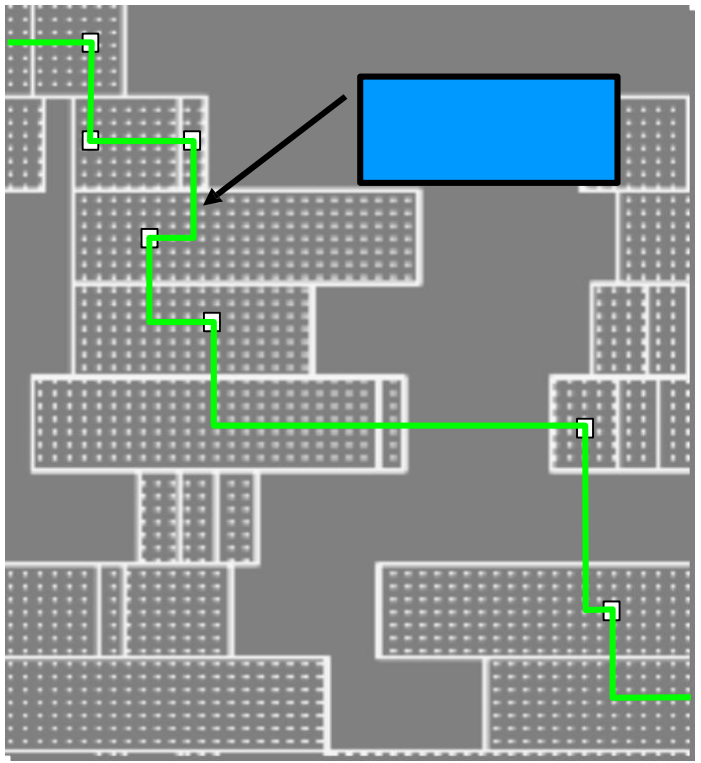
PLACEMENT STRATEGIES

- In addition to wire-length minimization, placement can be driven by two additional primary targets:
  - Timing Optimization (Timing-driven placement)
  - Congestion Minimization (Congestion-driven placement)
- Further targets include clock tree and power optimizations

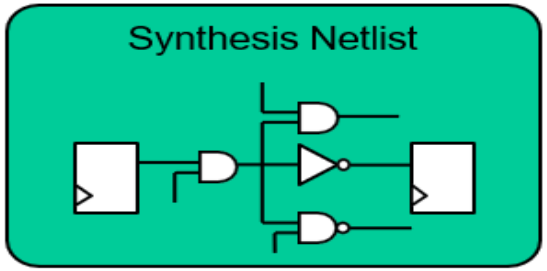


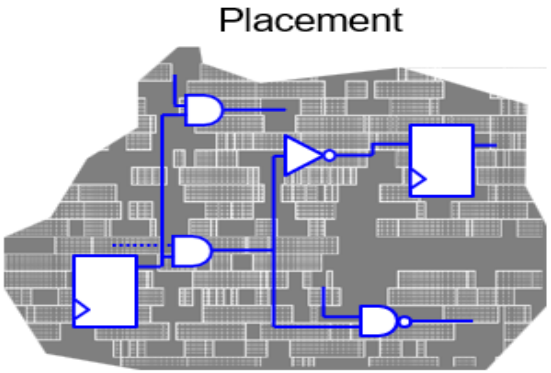
TIMING-DRIVEN PLACEMENT

- Timing-driven placement tries to place critical path cells close together to reduce net RCs and to meet setup timing
- RCs are based on Virtual Route (VR)
  - Layers are not taken into consideration
- Timing-driven placement based on Virtual Route
  - Tries to place cells along timing-critical paths close together to reduce net RCs and meet setup timing
  - Net RCs are based on Virtual Routing estimates



TIMING-DRIVEN PLACEMENT



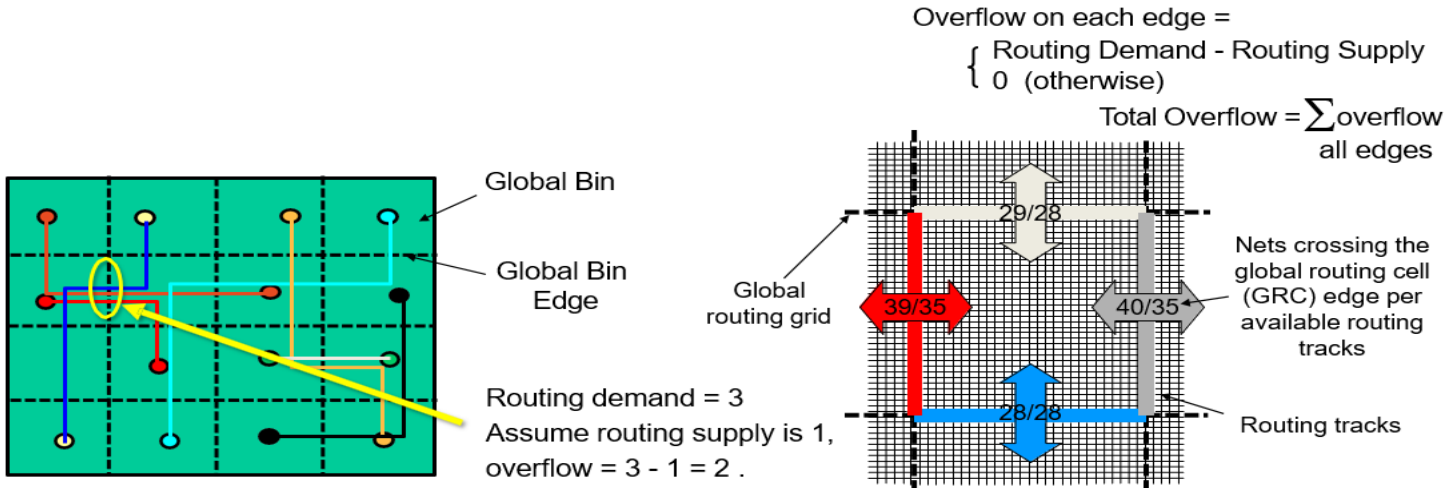


Statistically Based
Wire Load Model (WLM)

Net Fanout	Resistance KΩ	Capacitance pF
1	0.0498	0.045
2	0.1295	0.0812
3	0.2092	0.1312
4	0.2888	0.1811

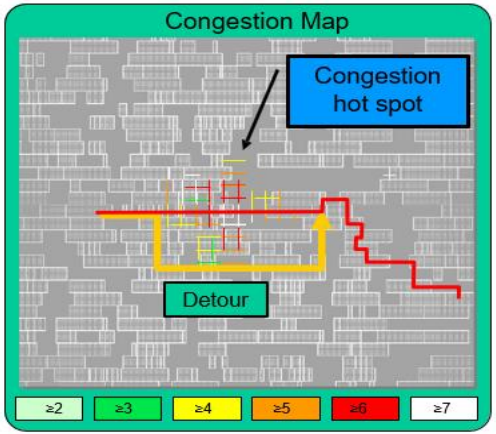
CONGESTION

- Congestion occurs when the number of required routing tracks exceeds the number of available tracks.
  - Congestion can be estimated from the results of a quick global route.
  - Global bins with routing overflow can be identified.



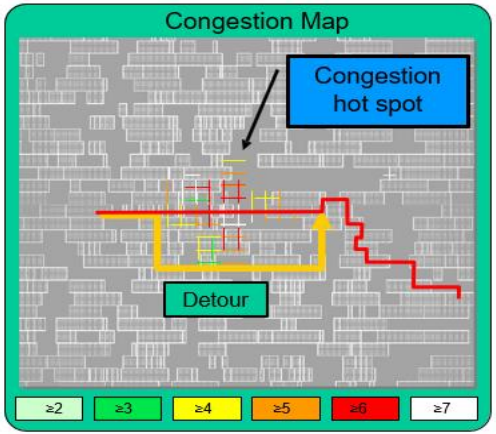
CONGESTION

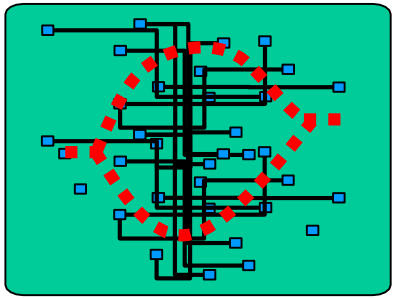
- Issues with Congestion
  - If congestion is not too severe, the actual route can be detoured around the congested area
  - The detoured nets will have worse RC delay compared to the VR estimates
  - In highly congested areas, delay estimates during placement will be optimistic.



CONGESTION

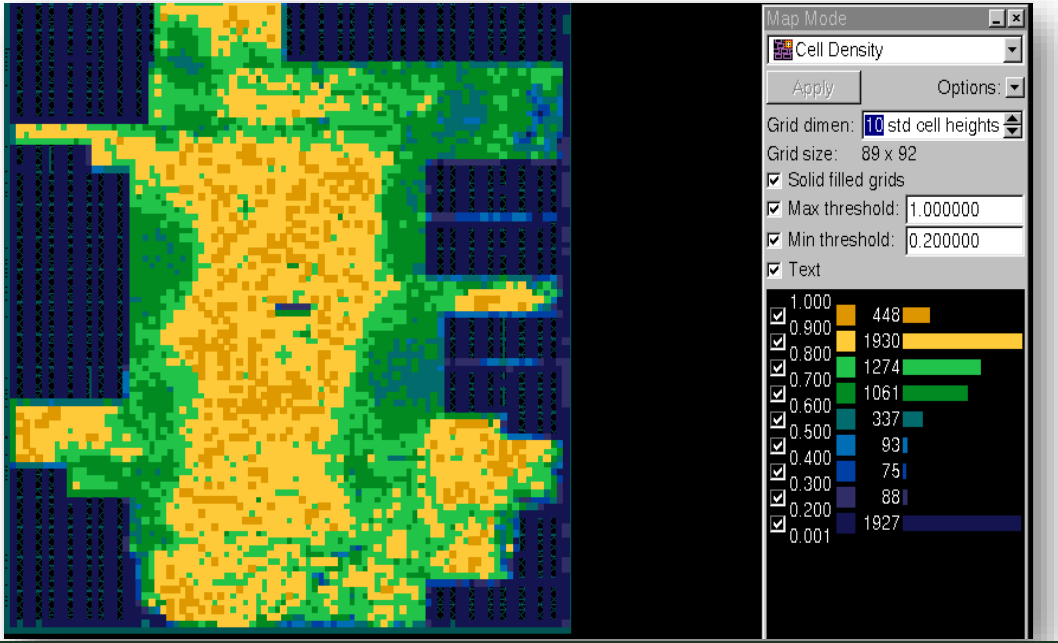
- Issues with Congestion
  - If congestion is not too severe, the actual route can be detoured around the congested area
  - The detoured nets will have worse RC delay compared to the VR estimates
  - In highly congested areas, delay estimates during placement will be optimistic.
- Not routable or severely congested design
  - It is important to minimize or eliminate congestion before continuing
  - Severe congestion can cause a design to be un-routable

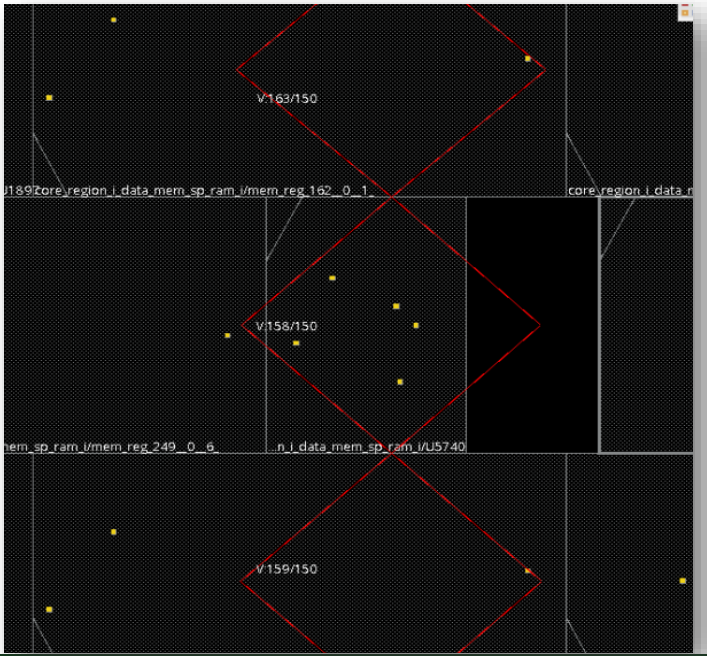




CONGESTION MAPS

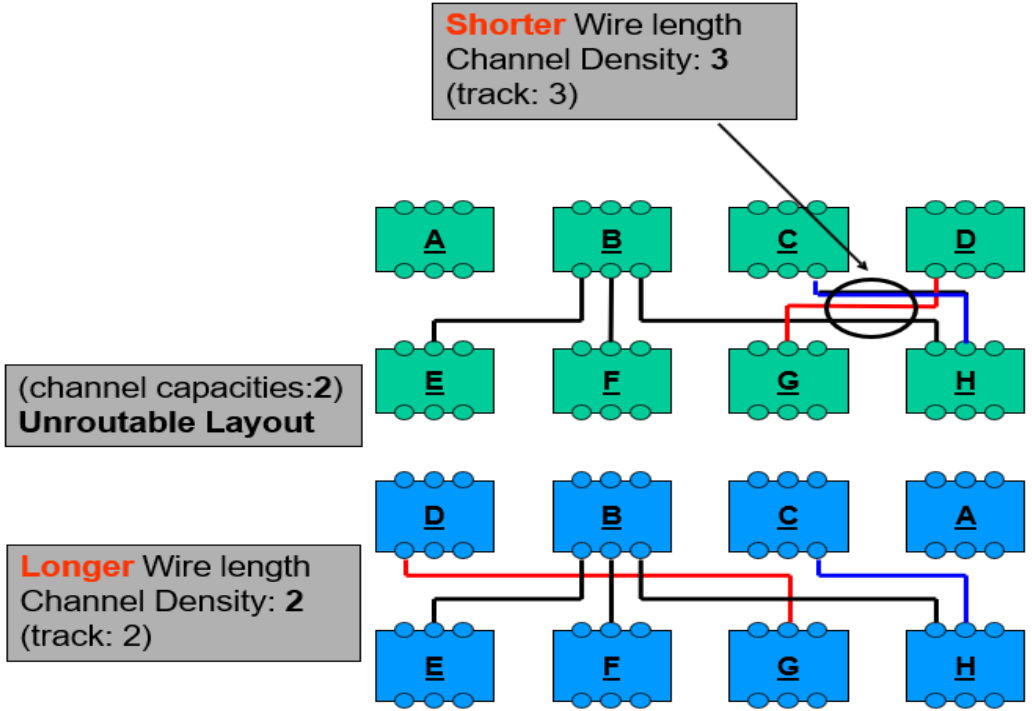
- Congestion maps are displayed by the backend tool to help us evaluate the total congestion, identify and fix congestion hot spots.





CONGESTION-DRIVEN PLACEMENT

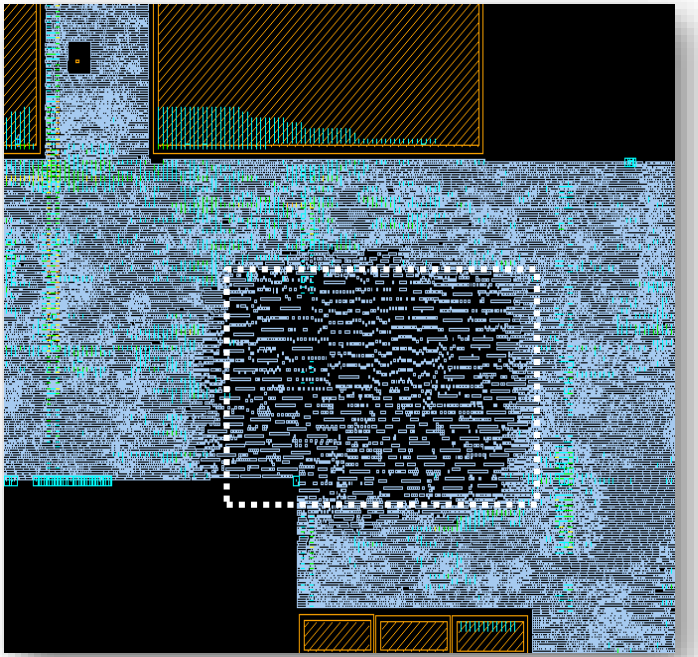
- Congestion Reduction
  - The tool tries to evaluate congestion hotspots and spread the cells (lower utilization) in the area to reduce congestion.
  - The tool can also choose cell location based on congestion, rather than wire-length.



STRATEGIES TO FIX CONGESTION

Modify the floorplan:

- Mark areas for low utilization.
- Top-level ports
  - Changing to a different metal layer
  - Spreading them out, re-ordering or moving to other sides
- Macro location or orientation
  - Alignment of bus signal pins
  - Increase of spacing between macros
  - Add blockages and halos



STRATEGIES TO FIX CONGESTION

Modify the floorplan:

- Core aspect ratio and size
  - Making block taller to add more horizontal routing resources
  - Increase of the block size to reduce overall congestion
- Power grid
  - Fixing any routed or non-preferred layers

Thank you!