ECE6703J

Computer-Aided Design of Integrated Circuits

Analytical Placement

Outline

- Analytical Placement
  - Quadratic Placement
  - Recursive Partitioning

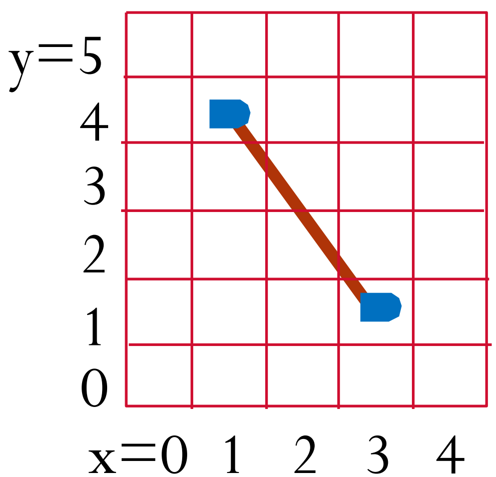
Analytical Placers: The Problem

- **<u>Goal</u>**: Write an **equation** whose **minimum** is the final placement.
  - If you have a million gates, need a million (x_i, y_i) values as result.
  - Formulate an appropriate **cost function** for all (x_i, y_i)'s: $F(x_1, x_2, \dots, x_M, y_1, y_2, \dots, y_M)$.
  - …then solve **analytically** for $X^* = (x_1, x_2, \dots, x_M)$, $Y^* = (y_1, y_2, \dots, y_M)$ to minimize F.
  - The resulting set of values of X^*, Y^* give you the placement of all 1M gates.
- This sounds sort of **<u>crazy</u>**… (an optimization problem with 2 million variables!) but it works **great**.
  - All modern placers for big ASICs and SOCs are "**analytical**".
  - Big trick is to write the wirelength in mathematically "**friendly**" form so that we can **optimize**.

Idea: Optimize Quadratic Wirelength Model

- For **2-point** net: squared length of the **straight line** between points.
  - Quadratic length$= (x_1 - x_2)^2 + (y_1 - y_2)^2$
- Why? works **nice mathematically**.

A "2-point" net



Quadratic wirelength=
$(3-1)^2 + (4-1)^2 = 13$

BUT… what happens if your net has more than 2 points in it?

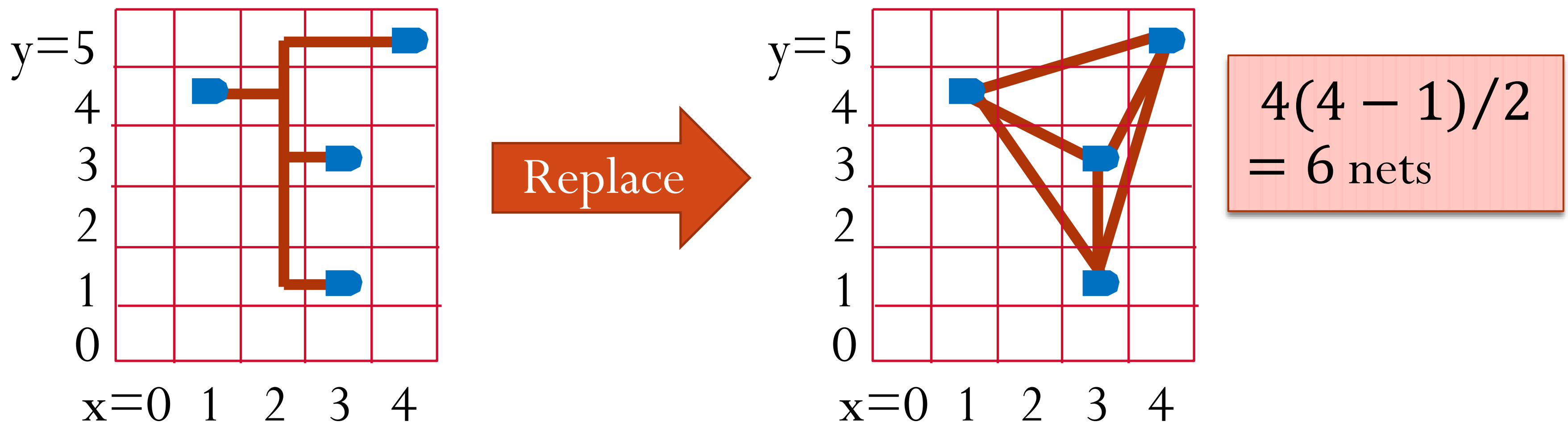
What About k-point Net, k>2?

1. **Replace** one "real" net with $k(k-1)/2$ 2-point nets.
   - Add a new net between **every pair of points**. Called a **fully-connected clique model**.
   - We use this model for all our subsequent examples.

2. Do a **weighted** sum of quadratic wirelengths over all new 2-point nets with weight $= 1/(k-1)$.
   - Why? 1 net became $k(k-1)/2$ nets. Need to **compensate** so that we don't "overestimate".
   - **Note also**: when $k = 2$, this weight is just 1, so **consistent with** 2-point nets.

Example

A “4-point” net **Clique model**



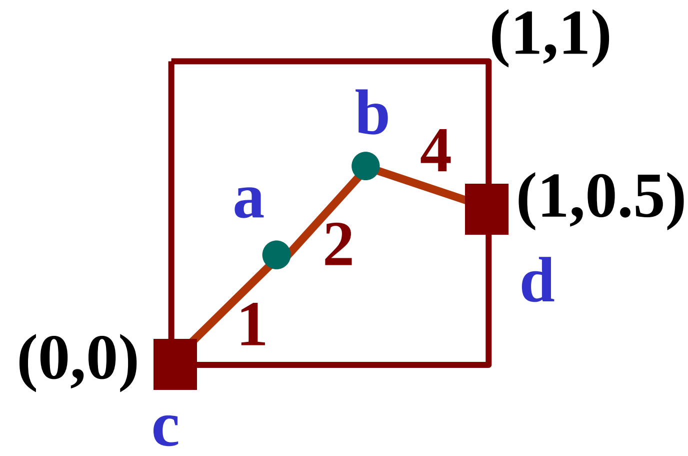
- Quadratic estimate:
  - Sum of 6 weighted 2-point net lengths, with weight = 1/3.

$$\frac{1}{3}[(1-3)^2+(4-3)^2]+\frac{1}{3}[(1-3)^2+(4-1)^2]$$
$$+\frac{1}{3}[(1-4)^2+(4-5)^2]+\frac{1}{3}[(3-3)^2+(3-1)^2]$$
$$+\frac{1}{3}[(3-4)^2+(3-5)^2]+\frac{1}{3}[(3-4)^2+(1-5)^2]=18$$

One More Big Idea: Gates as "Points"

- To make the math work out easily, one more simplification:
  - Ignore the physical size of all the gate – **pretend** gates are **dimensionless points**.
  - And, we will **ignore** (for now…) constraint that **gates cannot overlap**.
- Why…?
  - Allows us to write a very simple, very elegant "equation" for the placement.
  - We can solve it, quickly and effectively.
  - We can then use another set of methods – later in this lecture – to repair this.

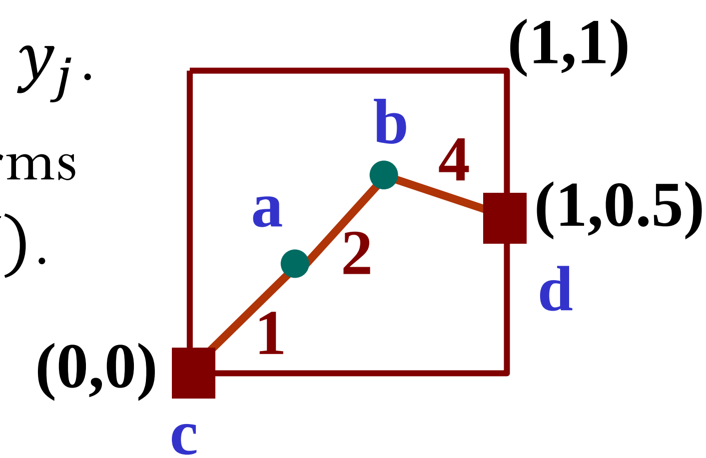
Example



- Chip surface is a rectangle.
  - X from 0 to 1; Y from 0 to 1.
  - This is totally arbitrary.
- 2 gate "points", index a and b
- 2 pads, index c and d
  - Pad = **fixed** pin (red square) on the edge of the chip. These do not move.
- 3 nets, each with a **weight**.
  - We will minimize the **total quadratic wirelengths** for these 3 nets.
  - Weight gives us "**control**" or "**freedom**".
  - Each net has 2 points to keep example simple.

Easy to Write the Quadratic Wirelength

- Assume the location for gate "a" is (x_1, y_1) and for gate "b" is (x_2, y_2).
- Quadratic wirelength for:
  - net (a, c): $l_1 = 1 \cdot (x_1 - 0)^2 + 1 \cdot (y_1 - 0)^2$
  - net (a, b): $l_2 = 2 \cdot (x_1 - x_2)^2 + 2 \cdot (y_1 - y_2)^2$
  - net (b, d): $l_3 = 4 \cdot (x_2 - 1)^2 + 4 \cdot (y_2 - 0.5)^2$
- The total quadratic wirelength is $Q = l_1 + l_2 + l_3$.
- **Note**: Sum Q has no terms like $x_i \cdot y_j$.
- **Claim**: We can separate x and y terms in the sum Q as $Q = Q(X) + Q(Y)$.

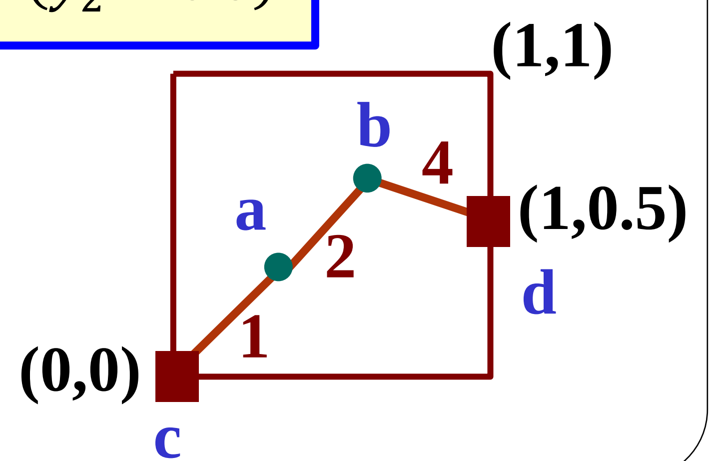


Easy to Write the Quadratic Wirelength

- Quadratic wirelength for:
  - net (a, c): $l_1 = 1 \cdot (x_1 - 0)^2 + 1 \cdot (y_1 - 0)^2$
  - net (a, b): $l_2 = 2 \cdot (x_1 - x_2)^2 + 2 \cdot (y_1 - y_2)^2$
  - net (b, d): $l_3 = 4 \cdot (x_2 - 1)^2 + 4 \cdot (y_2 - 0.5)^2$
- $Q = l_1 + l_2 + l_3 = Q(X) + Q(Y)$.

$$Q(X) = 1(x_1 - 0)^2 + 2(x_1 - x_2)^2 + 4(x_2 - 1)^2$$

$$Q(Y) = 1(y_1 - 0)^2 + 2(y_1 - y_2)^2 + 4(y_2 - 0.5)^2$$



How Do We Minimize the Objective?

$$Q(X) = 1(x_1 - 0)^2 + 2(x_1 - x_2)^2 + 4(x_2 - 1)^2$$
$$Q(Y) = 1(y_1 - 0)^2 + 2(y_1 - y_2)^2 + 4(y_2 - 0.5)^2$$

- Basic calculus!
  - Differentiate, set derivative to 0, then solve!
  - But this is multiple variables! So, we do **partial derivatives**, set each to 0, solve.

$$\partial Q(X)/\partial x_1 = 2x_1 + 4(x_1 - x_2) + 0 = 6x_1 - 4x_2 = 0$$

$$\partial Q(X)/\partial x_2 = 0 - 4(x_1 - x_2) + 8(x_2 - 1) = -4x_1 + 12x_2 - 8 = 0$$

$$\partial Q(Y)/\partial y_1 = 2y_1 + 4(y_1 - y_2) + 0 = 6y_1 - 4y_2 = 0$$

$$\partial Q(Y)/\partial y_2 = 0 - 4(y_1 - y_2) + 8(y_2 - 0.5) = -4y_1 + 12y_2 - 4 = 0$$

How Do We Minimize the Objective?

$$\partial Q(X)/\partial x_1 = 6x_1 - 4x_2 = 0$$
$$\partial Q(X)/\partial x_2 = -4x_1 + 12x_2 - 8 = 0$$

$$\begin{bmatrix} 6 & -4 \\ -4 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \end{bmatrix}$$

$$x_1 = 4/7, x_2 = 6/7$$

$$\partial Q(Y)/\partial y_1 = 6y_1 - 4y_2 = 0$$
$$\partial Q(Y)/\partial y_2 = -4y_1 + 12y_2 - 4 = 0$$

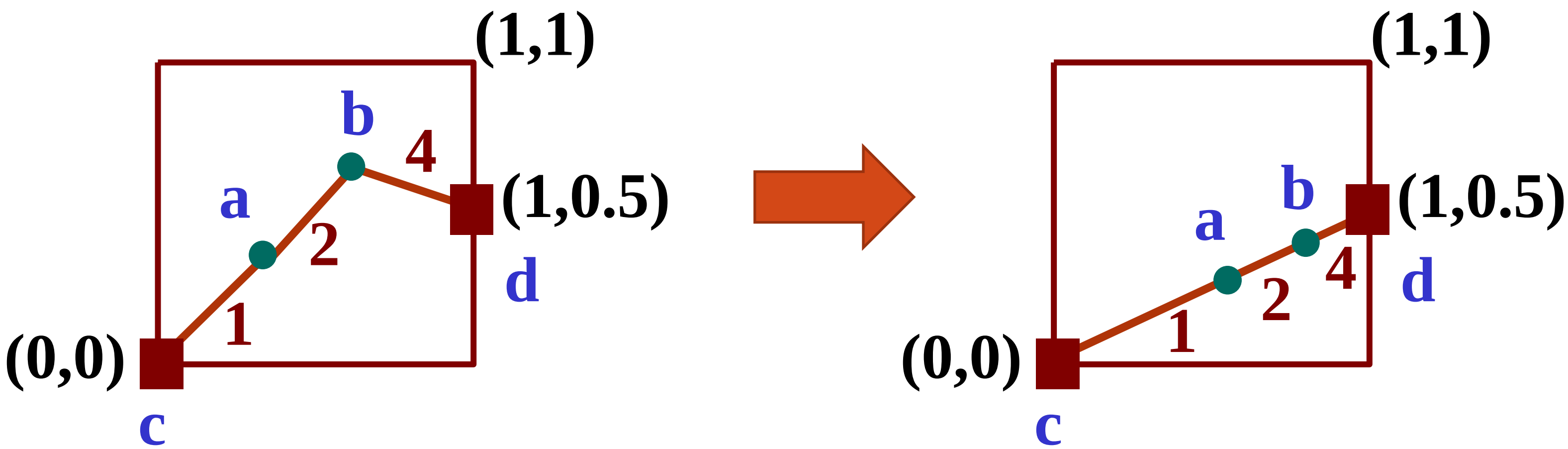
$$\begin{bmatrix} 6 & -4 \\ -4 & 12 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

$$y_1 = 2/7, y_2 = 3/7$$

- **Observations**
  - Two matrix equations: $AX = b_X$ and $AY = b_Y$.
  - Same matrix for X, Y, but different b vectors.
  - If you have N gates, matrix A is $N \times N$ and vectors X, Y, b_X, b_Y have N elements.

Placement Result

a: $(x_1, y_1) = (4/7, 2/7)$
b: $(x_2, y_2) = (6/7, 3/7)$

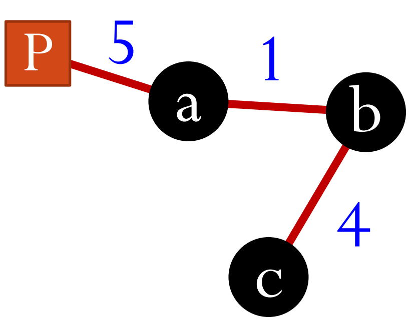


- **Observations**
  - Placement makes visual sense. All points on a **straight line** between the pads.
    - **Analogy**: each 2-point wire is like a **spring** (with different strength). Placement minimizes total spring **energy**.
    - When is spring energy minimized? At their equilibrium state!
  - **Bigger** weight on the wire → **shorter** wire. Weight gives us lots of control over placement.

Quadratic Placement: What is Matrix A?

- Surprisingly simple recipe to build the required A matrix.
  - First, build the $N \times N$ **connectivity** matrix, called C.
  - If gate i has a 2-point wire to gate j with weight w, then let $c[i,j] = c[j,i] = w$, else let them be 0.

- New (bigger) example, with 3 gates, 1 pad (P), and 3 wires (with weights).



$$C = \begin{array}{c} \begin{matrix} \text{a} & \text{b} & \text{c} \end{matrix} \\ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 4 \\ 0 & 4 & 0 \end{bmatrix} \end{array} \begin{matrix} \text{a} \\ \text{b} \\ \text{c} \end{matrix}$$

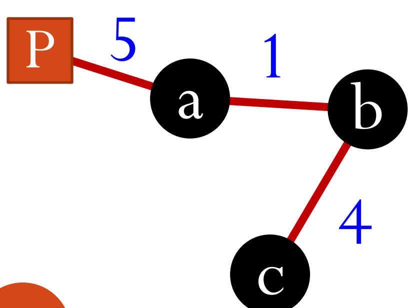
Note:
- C matrix **ignores** pads.
- Diagonal entries are 0.

Quadratic Placement: What is Matrix A?

- Use the connectivity matrix C to build matrix A.
  - Elements $a[i,j]$ **not** on the matrix diagonal are just $a[i,j] = -c[i,j]$.
  - Elements **on the diagonal** are

$$a[i,i] = \left(\sum_{j=1}^{n} c[i,j]\right) + (\text{weights of any pad wire })$$

    - **<u>In words</u>**: **add up** the i-th row of C and then add in weights on (possible) wires connecting gate i to a pad.

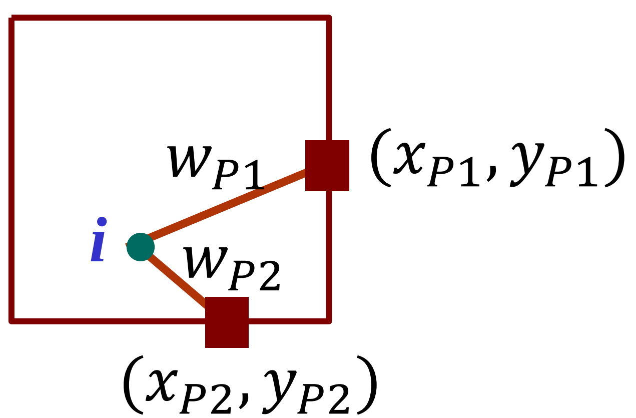


$$C = \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 4 \\ 0 & 4 & 0 \end{bmatrix} \end{array} \begin{array}{c} \\ a \\ b \\ c \end{array} \Rightarrow A = \begin{array}{c} \begin{array}{ccc} a & b & c \end{array} \\ \begin{bmatrix} 6 & -1 & 0 \\ -1 & 5 & -4 \\ 0 & -4 & 4 \end{bmatrix} \end{array} \begin{array}{c} \\ a \\ b \\ c \end{array}$$

How to Build b_X and b_Y Vectors?

$$AX = b_X, AY = b_Y$$

- If gate i connects to M pads at $(x_{P1}, y_{P1}), \ldots, (x_{PM}, y_{PM})$ with wires with weight $w_{P1}, \ldots, w_{PM}$, repectively.
  - Then set $b_X[i] = w_{P1}x_{P1} + \cdots + w_{PM}x_{PM}$ and $b_Y[i] = w_{P1}y_{P1} + \cdots + w_{PM}y_{PM}$.
- Otherwise (the gate connects to **no** pad), set $b_X[i] = b_Y[i] = 0$.



Method to Compute A and b Makes Sense!

- Suppose gate i $(i > k)$ connects to gates $1,2,\dots,k$ and one pad P.
- Then, the terms with x_i in $Q(X)$ are

$$w_1(x_i - x_1)^2 + w_2(x_i - x_2)^2 + \cdots$$
$$+ w_k(x_i - x_k)^2 + w_P(x_i - x_P)^2$$

  - **Note**: $x_i, x_1, x_2, \dots, x_k$ are **variables**; $x_P, w_1, \dots w_k, w_p$ are **constants**.

- Partial derivative

$$\partial Q(X)/\partial x_i = 2w_1(x_i - x_1) + 2w_2(x_i - x_2) + \cdots$$
$$+2w_k(x_i - x_k) + 2w_P(x_i - x_P) = 0$$

$$\underbrace{(w_1 + \cdots + w_k + w_P)}_{\boxed{A[i,i]}}x_i + \underbrace{(-w_1)}_{\boxed{A[i,1]}}x_1 + \cdots + \underbrace{(-w_k)}_{\boxed{A[i,k]}}x_k = \underbrace{w_P x_P}_{\boxed{b_X[i]}}$$

About the Ax=b Matrix Solving

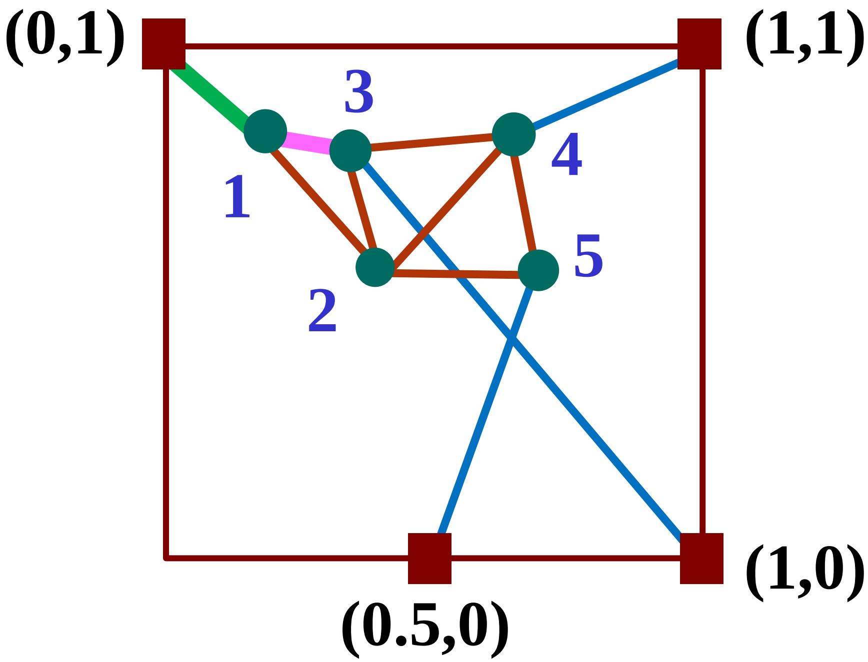
- Are the equation solving **difficult**, in practice?
  - If we have 1M gates, this is a 1M x 1M matrix A, with 1M element vectors x and b!

- No! The equation is very **easy** to solve, even when very large.
  - The matrix A has a special form. It is **sparse**, **symmetric**, and **diagonally dominant**.
  - Mathematically: A is **positive semi-definite**. Very simple to solve!

About the Ax=b Matrix Solving

- We use **iterative**, **approximate** solvers, in practice (i.e., not Gaussian elimination).
  - This means the solver converges gradually to the right answer.
  - But, also means that the answers can be a little bit "**off**", not quite **exact**.

Example: 4 Pads + 5-Gate Netlist



$$C = \begin{pmatrix} 0 & 1 & 10 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 10 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

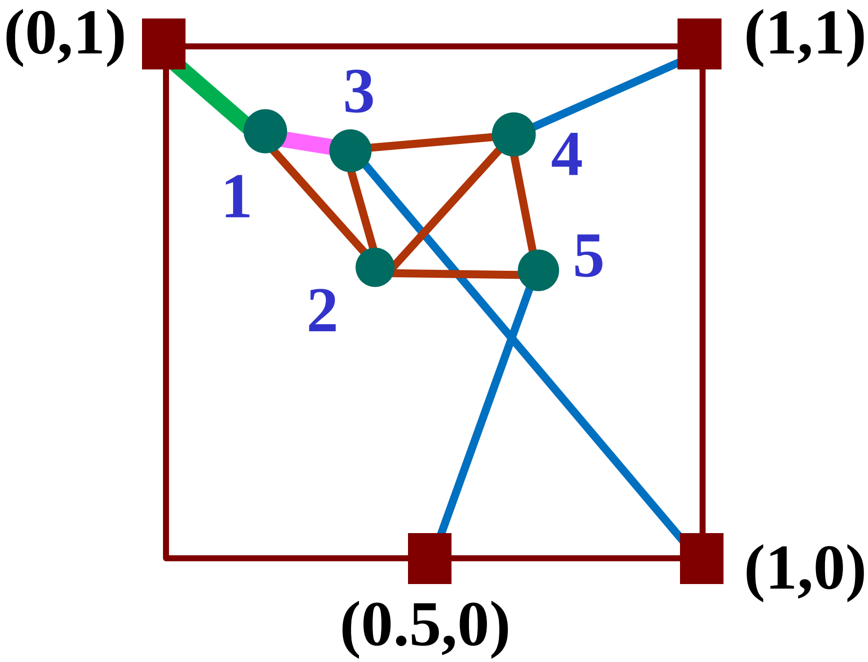
$$A = \begin{pmatrix} 21 & -1 & -10 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -10 & -1 & 13 & -1 & 0 \\ 0 & -1 & -1 & 4 & -1 \\ 0 & -1 & 0 & -1 & 3 \end{pmatrix}$$

All **blue** and **red** wire weights = 1;

Pink wire weight = 10;

Green wire weight = 10;

Example: 4 Pads + 5-Gate Netlist



$$b_X = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0.5 \end{pmatrix}$$

$$b_Y = \begin{pmatrix} 10 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

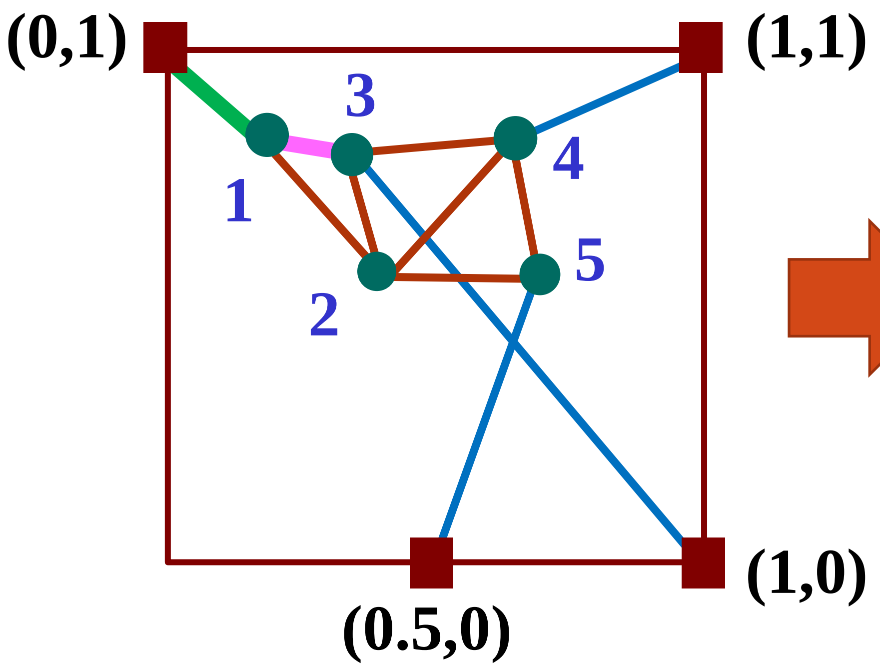
All **blue** and **red** wire weights = 1;

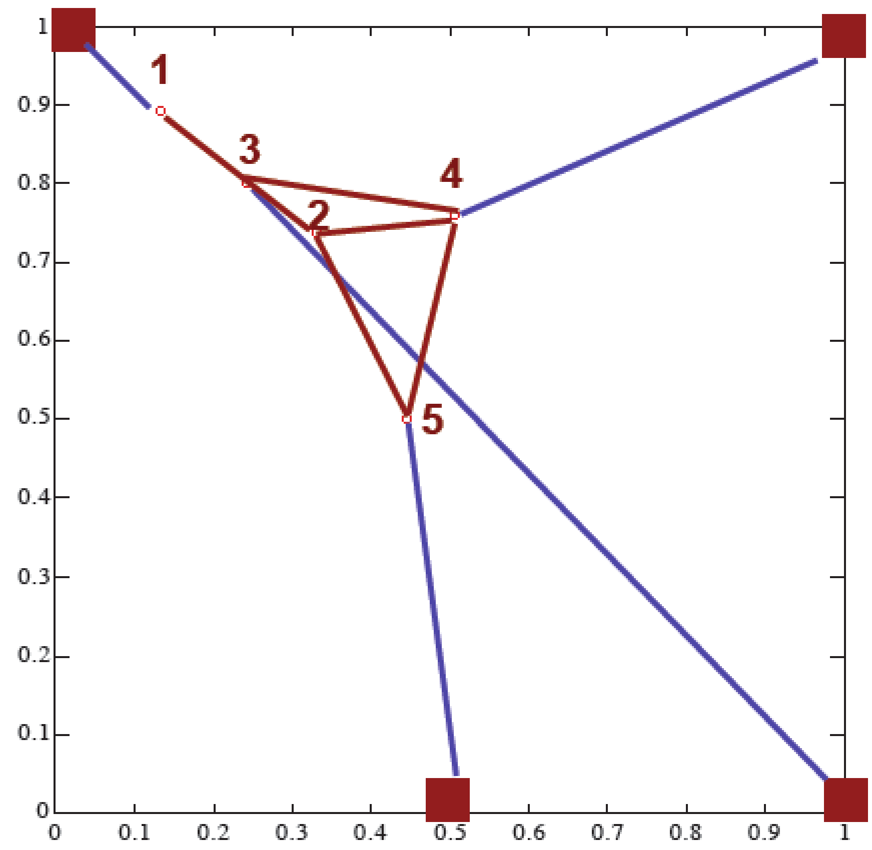
Pink wire weight = 10;

Green wire weight = 10;

Quadratic Placement Result

- Solving $AX = b_X$ and $AY = b_Y$, we get:



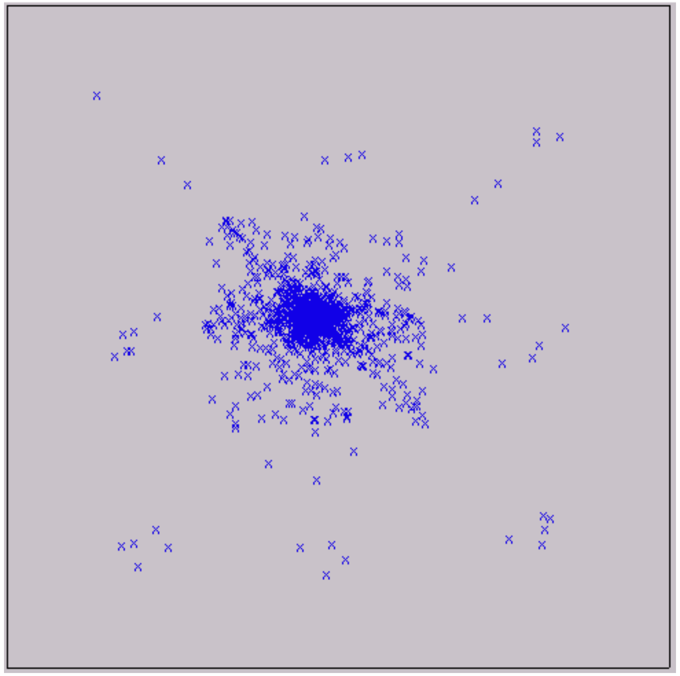


Outline

- Analytical Placement
  - Quadratic Placement
  - Recursive Partitioning

What Does A Real Quadratic Placement Look Like?

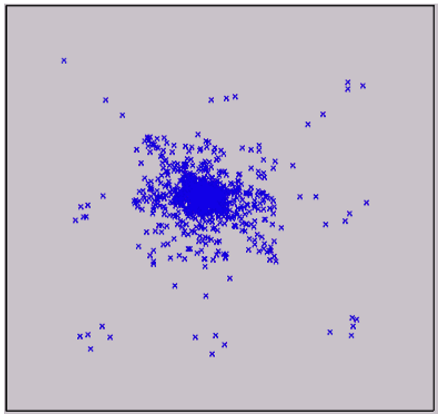
- Like this:
  - Small IBM ASIC, few thousand gates.



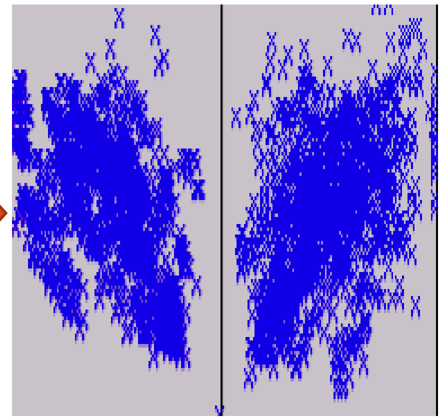
- New problem:
  - Quadratic model minimizes wirelength for **big** netlists, in a numerical way…
  - … but ignores that gates have **physical size**, cannot overlap.

- Now, we have to fix this…
  - Our solution: **recursive partitioning**

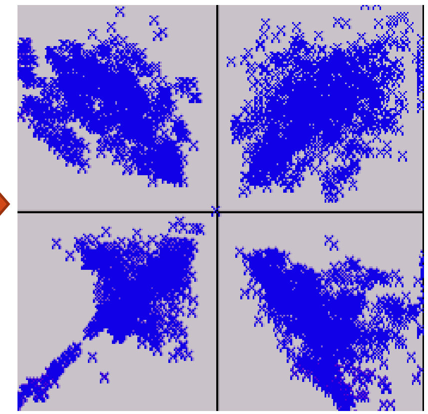
Big Idea: Recursive Partitioning



1st quadratic placement (**QP**) solving.



Partition chip into left/right. **Select** which gates on each side. **Solve** 2 new, smaller **QP** tasks.



Repeat. **Partition** each side top/bottom. **Select** which gates on each side. **Solve** 4 new, smaller **QP** tasks.

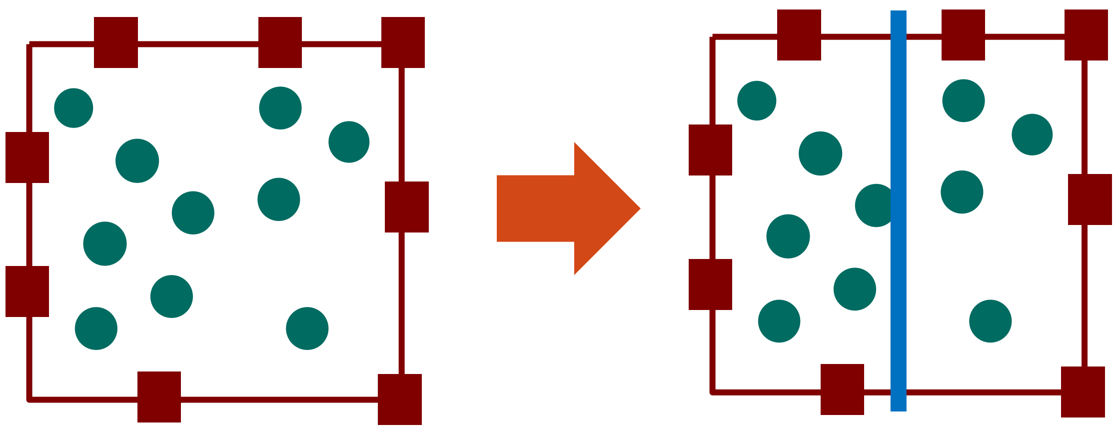
…
Keep going

Recursive Partitioning: Basic Steps

- **Partition**
  - How do we divide the chip into new, smaller placement tasks?
- **Assignment**
  - Which gates should go into each new, smaller region?
- **Containment**
  - Formulate new **QP** problems so that the gates **stay in new regions**, with short wirelength.
- Discuss one early strategy from a classical paper
  - Ren Song Tsay, Ernest Kuh, Chi Ping Hsu, "PROUD: A Sea-Of-gates Placement Algorithm," *IEEE Design & Test of Computers*, Dec 1988.

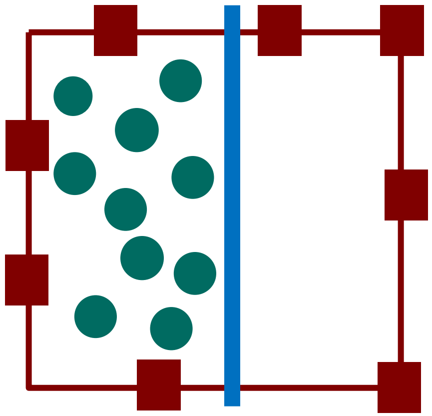
Recursive Partitioning: How to Partition

- Solution
  - After 1st quadratic placement (QP), divide chip area **exactly in half**, vertically.
    - Note: this is arbitrary. Horizontal is OK too.
  - We want **half** the gates on **each** side.
    - But, how do we achieve this?



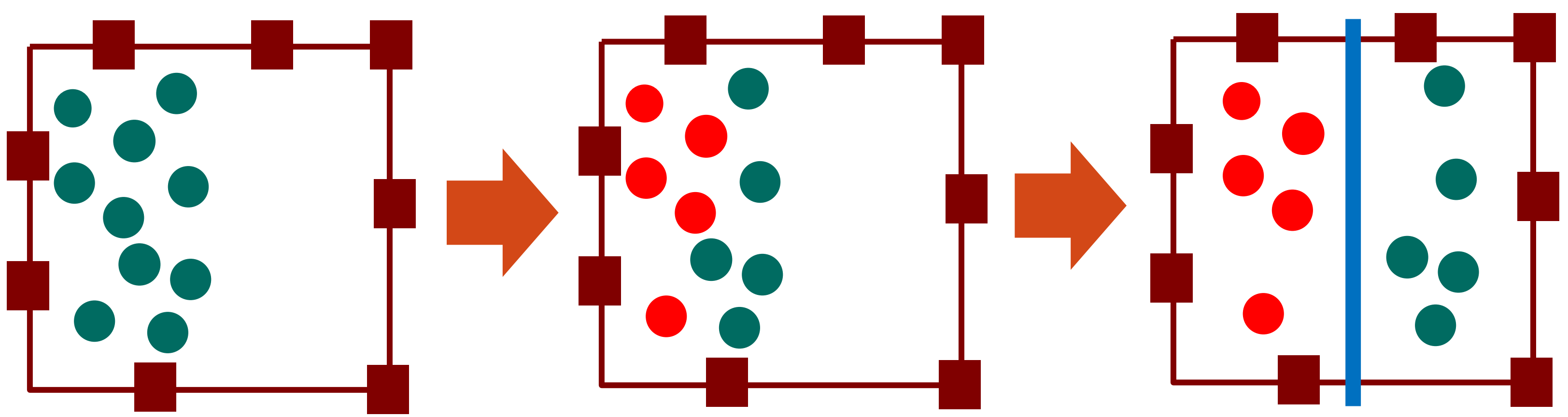
Recursive Partitioning: How to Assign

- **<u>Problem</u>**: What if QP does not spread gates evenly between halves?
  - Then, how do we know which gates to put **left/right** if this is initial QP?



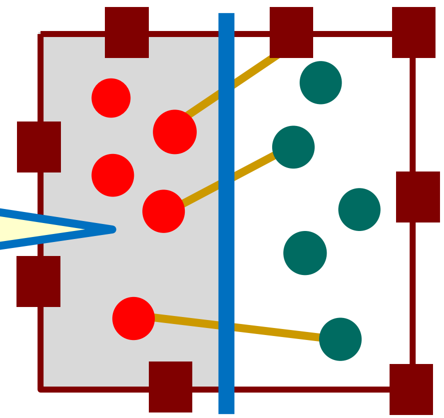
Recursive Partitioning: How to Assign

- Solution: **Sort the gates**
  - For **vertical** cut, sort gates on X coordinate first, then on Y coordinate if there is a tie.
    - For **horizontal** cut, sort on Y first, then X.
  - If N total gates, then assign **first** N/2 gates in sorted list to left. Others to right.



Recursive Partitioning: How to Contain

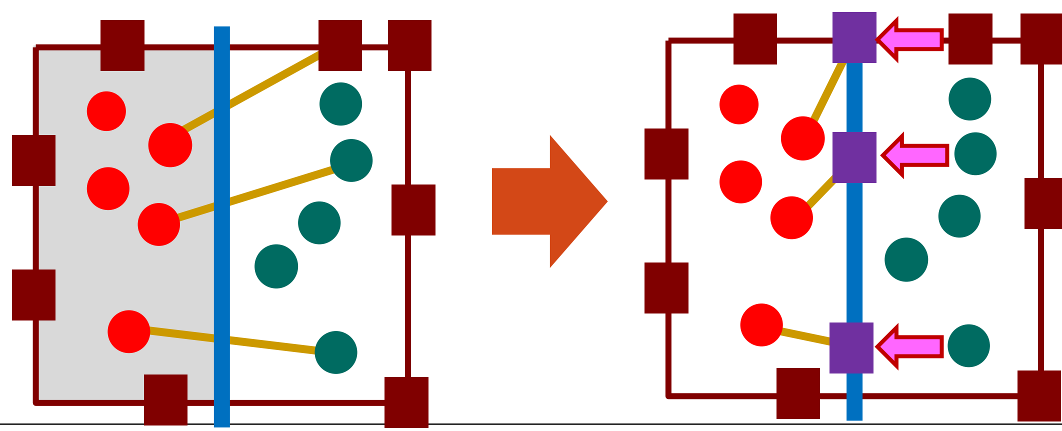
Focus on the gates **assigned to** the left side.



- Issues:
  - Some wires **connect** gates assigned to the left side to gates/pads on **right**. We can't ignore these!
  - However, if we keep these wires, the gates assigned to left side may be pulled **outside** the left region after the new QP.
    - Think this as a spring-mass system.
  - How do we solve this problem?

Recursive Partitioning: How to Contain

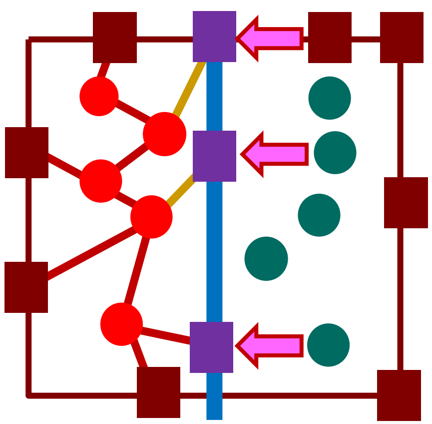
- **<u>Idea</u>: Pseudo-pads**
  - Every gate and pad **NOT assigned to** left half is modeled as a **pad on boundary** of left region.
- Details:
  - **Propagate** these outside gates and pads using their current (x, y) location to **<u>nearest</u> point** on left region.
  - For this simple first cut, we just take the y coordinate, and put pseudo-pad on the **center cut line**.



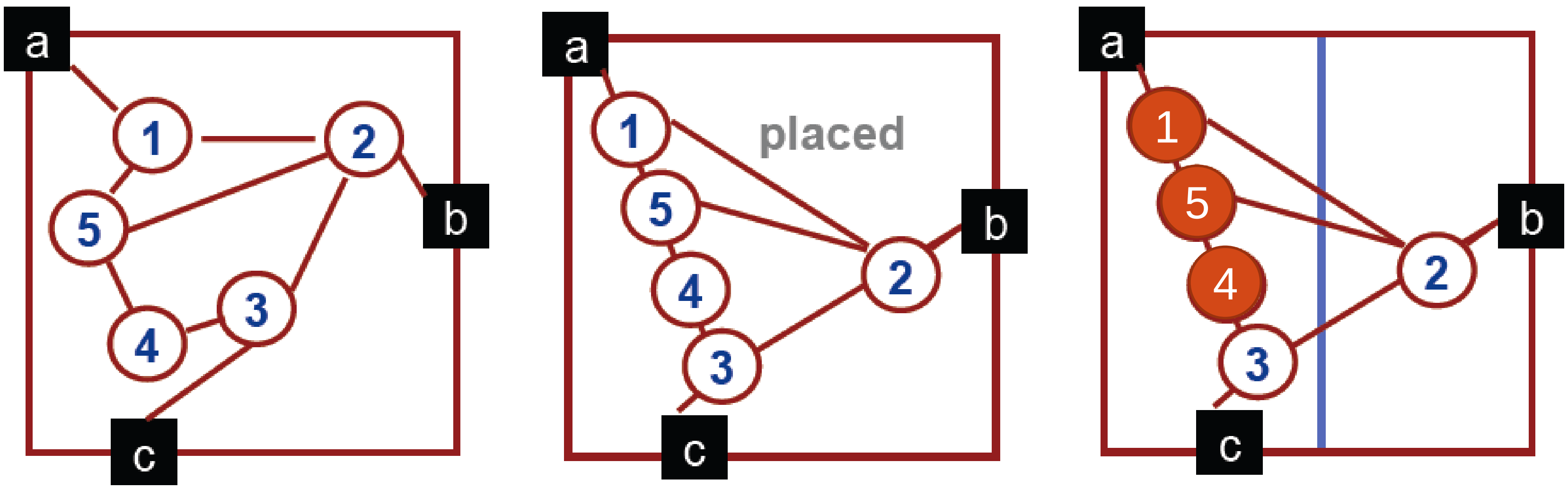
Result: new **QP** problem for gates in **left region**.

Pseudo-pads Achieves Containment

- Pseudo-pads guarantee all gates re-locate **inside** the region after the new QP.
  - Think of wires as "springs" that each pull gates **toward** other gates or pads.
  - Since pads (real & pseudo) are on edges of region, then **QP keeps gates inside**!



Partitioning Example



1. Initial netlist

5 gates (1,2,3,4,5)

9 wires

3 pads (a,b,c)

2. Initial QP result

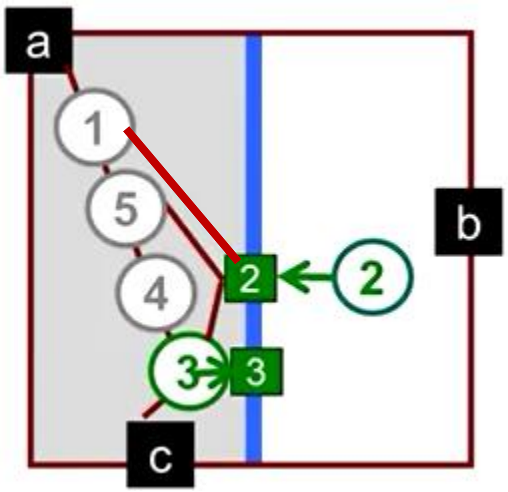
3. First partition

Sort on X:

Order is 1,5,4,3,2

Pick: 1,5,4 on left

Partitioning Example



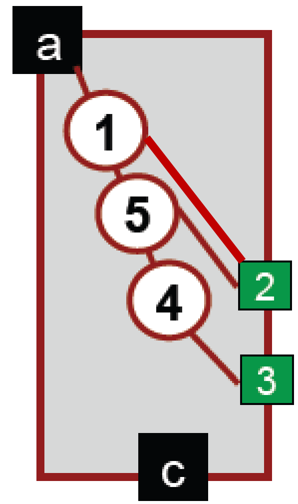
4. Propagate gates/pads

Right-side gates: 2,3

Right-side pads: b

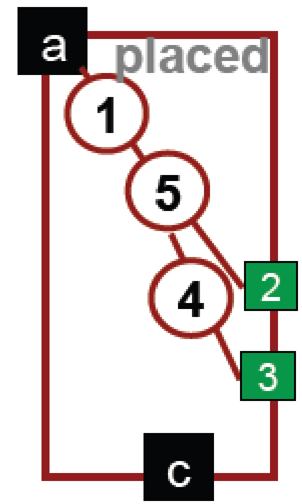
Push to vertical cut, using y coordinates.

Note: do not propagate pad b, since **no wires** on left connect to it



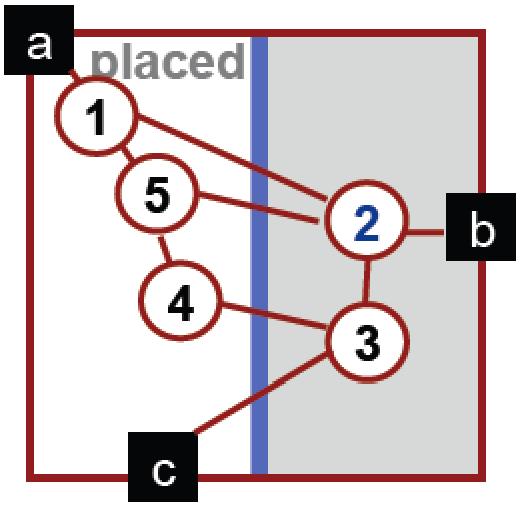
5. 2nd QP input

This is set up for a new smaller placement



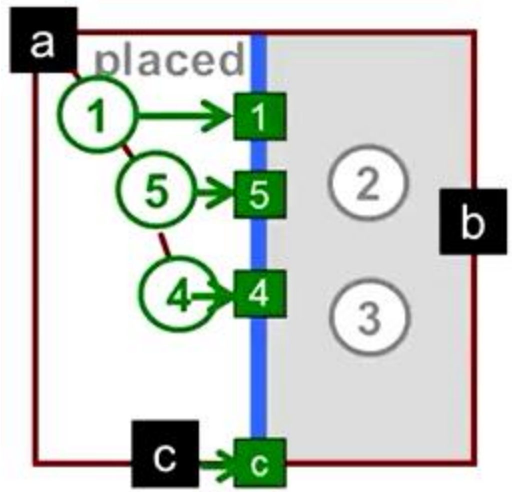
6. 2nd QP result

Partitioning Example



7. Left side placed

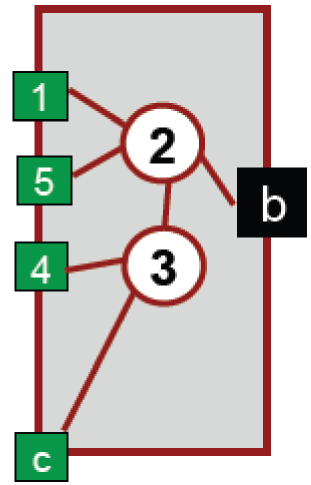
Now, re-place right-side gates.



8. Propagate gates/pads

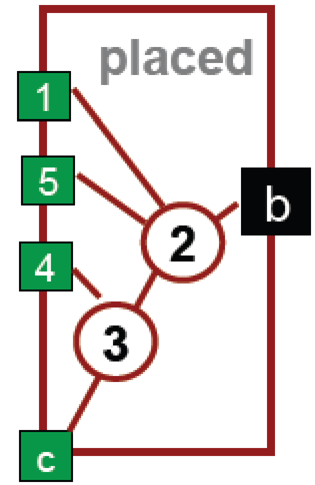
This is set up for next, new smaller placement

Note: locations of 1, 5, 4 from latest placement



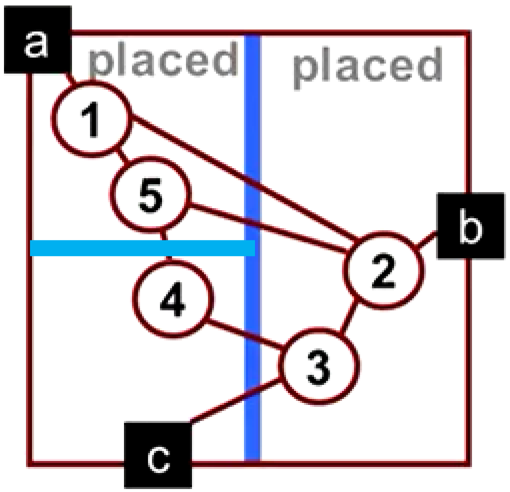
9. 3rd QP input

This is set up for a new smaller placement

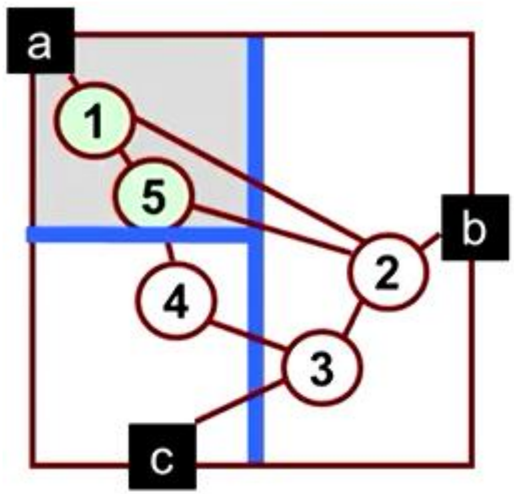


10. 3rd QP result

Partitioning Example



Repeat: Horizontal partition on left

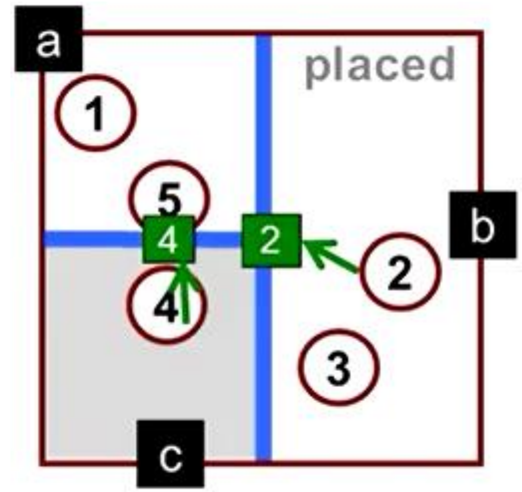


Focus on top

Sort gates on Y:

Order is 1,5, 4.

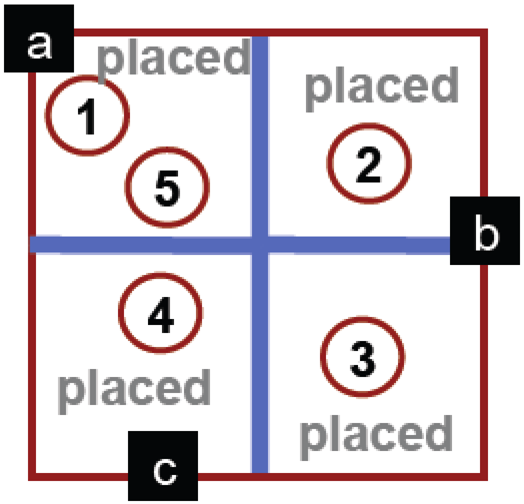
Assign 1,5 to region.



Propagate gates/pads

<u>**Note**</u>: Gate 4 propagates up to **bottom** of new region, while gate 2 propagates to **corner** of new region (nearest point)

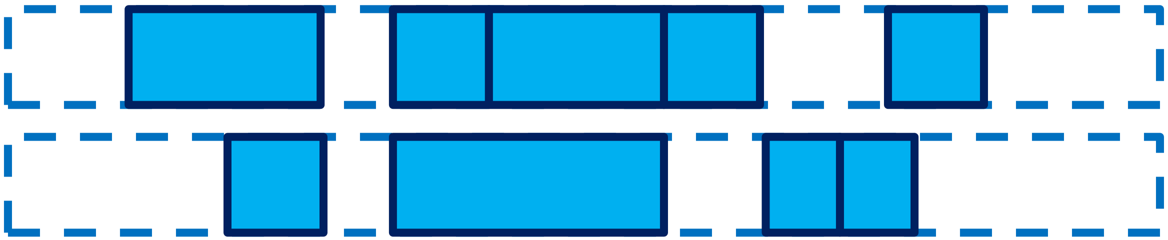
Keep Repeating this Recursion



Continue…

- **Keep recursively partitioning…**
  - Usually, you continue until you have a "**small**" number of gates in each region.
  - Small is typically 10~100.
  - Get a good, "global" placement, but not a "**<u>final</u>**" placement.

Final Placement Step: Legalization



- Still need to force gates in **precise rows** for final result.
  - QP methods **cannot** force all gates into standard cell rows
- We also need to remove overlaps.
- Solution step is called: **Legalization**
  - Many different algorithms, but we won't discuss.

Placement Summary

- Early placers based on **iterative improvement**.
  - **Simulated annealing** is a very good, famous example.
  - Annealing technique is used widely in VLSI CAD – but not for placers. Too inefficient.
- Modern placers are all **analytical**.
  - Many different mathematical formulations, but all similar.
  - Numerically optimize a mathematically friendly model of wirelength.
  - **Quadratic placement** is a famous, important, practical example.