(*Lec 14*) Placement & Partitioning: Part III

What you know

- That there are 3 big placement styles: iterative, recursive, direct
- Placement via iterative improvement using simulated annealing
- Recursive-style placement via min-cut with F&M partitioning

What you don't know

- The last style: direct placement
- One issue is mathematical model: quadratic wirelength minimization
- Second issue is legalization strategy: we do PROUD-style legalization

Copyright Notice

Where Are We?

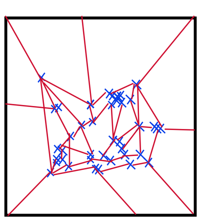
- Physical design--placement via *direct* methods

	M	T	W	Th	F	
Aug	27	28	29	30	31	1
Sep	3	4	5	6	7	2
	10	11	12	13	14	3
	17	18	19	20	21	4
	24	25	26	27	28	5
Oct	1	2	3	4	5	6
	8	9	10	11	12	7
	15	16	17	18	19	8
	22	23	24	25	26	9
	29	30	31	1	2	10
Nov	5	6	7	8	9	11
	12	13	14	15	16	12
Thnxgive	19	20	21	22	23	13
	26	27	28	29	30	14
Dec	3	4	5	6	7	15
	10	11	12	13	14	16

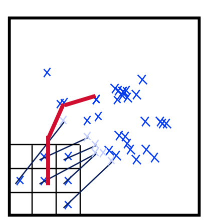
Introduction
Advanced Boolean algebra
JAVA Review
Formal verification
2-Level logic synthesis
Multi-level logic synthesis
Technology mapping
Placement
Routing
Static timing analysis
Electrical timing analysis
Geometric data structs & apps

Strategy: Direct Placement

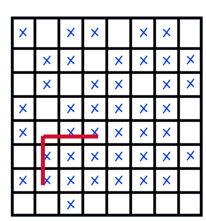
- All these use a technique called "Quadratic Placement"
  - **Model all gates as movable points, all wires as 2-point "springs"**
  - **Minimize total squared Euclidean length:** $\sum_i$ **EuclideanLength^2(net i)**
  - **Surprisingly, can do initial parts of this directly, numerically, exactly**



Initial Solution



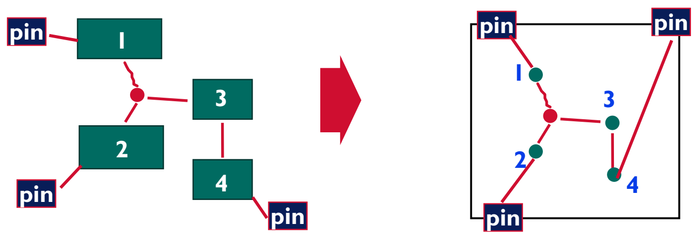
Legalization Strategy



Final Placement

Model Assumptions

- **Geometric simplifications**
  - **Gates: model as dimensionless points**
  - **Grid slots: none, ie, no placement grid, no "1 gate in 1 slot" constraints**
  - **Pins: must be fixed somewhere around boundary of the chip**
  - **Wires: we only allow 2-point connections; we minimize Σ length^2**



quadratic wirelength:

Model Assumptions: Multipoint Wires

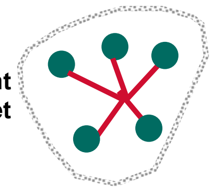
- **In real netlists, can have a wire connect to > 2 objects**
  - **If it connects to just 2 objects -- "points" -- called a "2 point net"**
  - **If it connects to > 2 objects -- called a "multipoint net"**

- **Idea**
  - **Decompose each multipoint net into a set of 2 point nets**
  - ***Necessary* to be able use the quadratic wirelength model: square of the length of the wire only really makes sense for 2 point net**
  - **How to decompose...?**

Decomposing Multipoint Nets

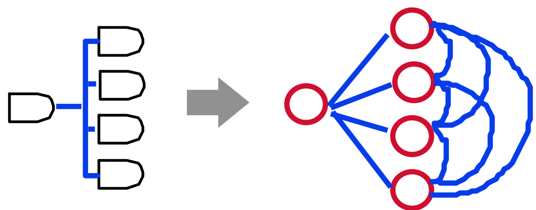
- Multi-point net
  - Suppose we have a 5 point net here

A 5-point net



Problem: quadratic wirelength is *what*?

- Solution: assume "fully connected" nets



All pt-to-pt connections must be included

k-pt net becomes [k •(k-1)] / 2 2-pt nets for us

#2-pt nets here ==

Weighting the Wires

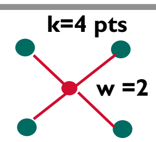
- Each wire can have a "weight"
  - Specifies its importance in the minimization problem...
  - ...or that there actually are multiple wires between 2 objects
  - In this formulation you can't tell the difference

- But what about a weighted multipoint wire?

Weighting the Wires

Question

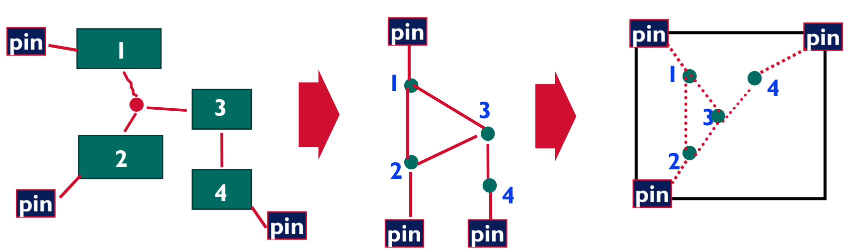
- When we decompose, what happens to the weights?
- Solution: for k-point net, multiply each 2 pt connection by
- Example: 4 point net, look at typical partition of it objects



Overall Model

Ideas

- Objects are dimensionless points: (xi, yi) placed arbitrarily; pins fixed
- Nets are all 2 point connections (maybe weighted) among these points
- Wirelength is measured as sum of quadratic net lengths



About the Model

- Why quadratic wirelength?
  - One reason: **can get an analytical solution to min[Σ quadratic wirelen]**
  - We can write equations, solve numerically for an exact, best minimum
- Tradeoffs
  - Quadratic wirelen NOT a particularly good model of the length of real wires after routing -- but we can get an analytical min length
  - Objects as dimensionless points NOT a particularly good model of a real placement -- must fix problems caused by ignoring shapes, and slots
  - But--there are "fixes" that deal with these problems, and it turns out you can do HUGE things--millions of gates--with these methods

Direct Formulation

- All quadratic wirelength min. problems look like this eqn:
- How to solve?
  - Transform this into a *standard* optimization problem
  - Requires some linear algebra, some calculus

Basic Matrix Stuff

- Linear equations
  - By now (I hope!) you should know that N linear equations in N unknows can be written compactly as a single matrix equation

$$\begin{aligned} a11\,x1 + a12\,x2 + a13\,x3 &= k1 \\ a21\,x1 + a22\,x2 + a23\,x3 &= k2 \\ a31\,x1 + a32\,x2 + a33\,x3 &= k3 \end{aligned} \quad \Rightarrow \quad \begin{pmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{pmatrix} \begin{pmatrix} x1 \\ x2 \\ x3 \end{pmatrix} = \begin{pmatrix} k1 \\ k2 \\ k3 \end{pmatrix}$$

- But how do we get to quadratic wirelength?

Quadratic Forms

- Turns out that x^T A x is the right form for quadratic wirelens
  - x is a column vector, x^T is a row vector, A a square matrix

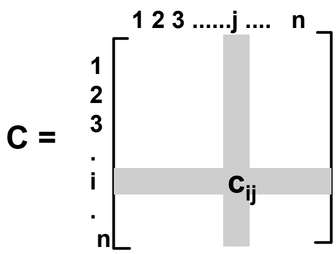
$x = \begin{bmatrix} x1 \\ x2 \end{bmatrix}$ $\quad x^T =$ $\quad A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ $\quad A^T =$

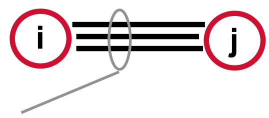
$x^T\ A\ x =$

  - x^T A x can represent a sum of (constant)•xi•xj for *all* possible pairs or i, j
  - But what exactly is the right way to set up this problem?

Start with Point-to-Point Connectivity Info

- **Placeable objects**
  - Set of n connected points {1, 2, ..., n}
- **Nets**
  - 2 point connections only, as discussed before
  - A weighted connectivity matrix represents these connections





c_{ij} = 3 here

Can think of this as either:

i connects to j with 1 net of weight 3
i connects to j with 3 nets of weight 1

We can't tell the difference using C matrix

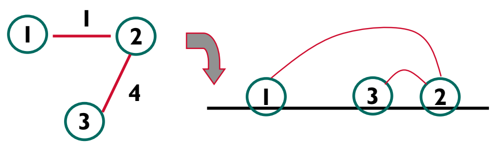
Matrix Formulation

- **Start with a simpler problem, 1-dimensional placement**
  - We want to place the objects {1, 2, ..., n} *on a line*
  - Means we want to solve for x1, x2, ..., xn to minimize weighted quadratic wirelength
- **Question: what is right A for $x^T A x$?**

Quadratic wirelen	Matrix form

Matrix Formulation

- When in doubt, try a little example: 3 objects placed on a line



C matrix is:

Quadratic wirelen is:

Matrix Formulation

- Turns out this is the right matrix A for the job:

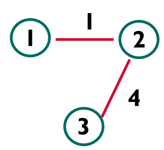
Quadratic wirelen = $1*x_1^2 + 5*x_2^2 + 4*x_3^2 - 2*x_1*x_2 - 8*x_2*x_3 - 0*x_1*x_3$

$$= x^T A x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 5 & -4 \\ 0 & -4 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Try it:

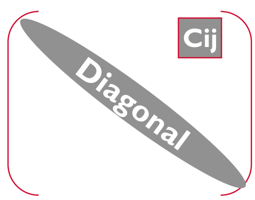
Matrix Formulation

- Look closely: compare C and A; can you see *pattern*?



$$C = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 4 \\ 0 & 4 & 0 \end{bmatrix} \qquad A = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 5 & -4 \\ 0 & -4 & 4 \end{bmatrix}$$

A =



Matrix Formulation Summary

- It all works

netlist ➡ C $\begin{bmatrix} c_{ij} \end{bmatrix}$ ➡ A $\begin{bmatrix} a_{ij} \end{bmatrix}$

a_{ii} diagonal = $\Sigma_j\, c_{ij}$

a_{ij} off diagonal = $-c_{ij}$

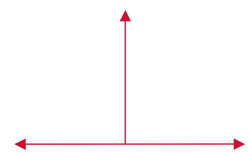
$$1/2\ \Sigma_i\ \Sigma_j\ [c_{ij} \bullet (xi - xj)^2\ ] \Rightarrow x^T A\, x$$

- New problem
  - I don't just want to write this wirelength down…
  - …I want to *solve for the vector of x locations that minimizes it*
  - How?

Minimizing $x^T A x$

- This minimization is just a higher dimensional version of something you already should know…
- 1-variable version

Just one variable, x



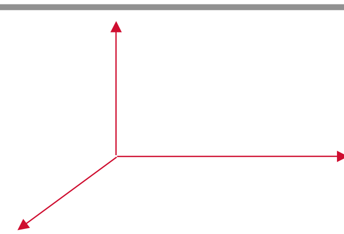
To minimize…

Minimizing $x^T A x$

- Higher dimensional version: 2-variable case

2 variables, x_1 x_2

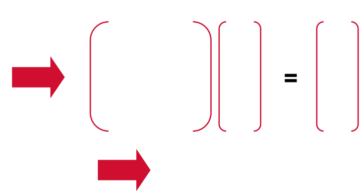
$$f = x_1^2 - x_1 x_2 + x_2^2$$



To minimize…

$$\frac{\partial f}{\partial x} = 2x_1 - x_2 = 0$$

$$\frac{\partial f}{\partial x} = -x_1 + 2x_2 = 0$$



Oops: Problem

- The *only* direct solution here is xi=0 for all i
  - This is the solution to the **unconstrained** form of the problem
  - We have to add some additional **constraints** to avoid this trivial soln

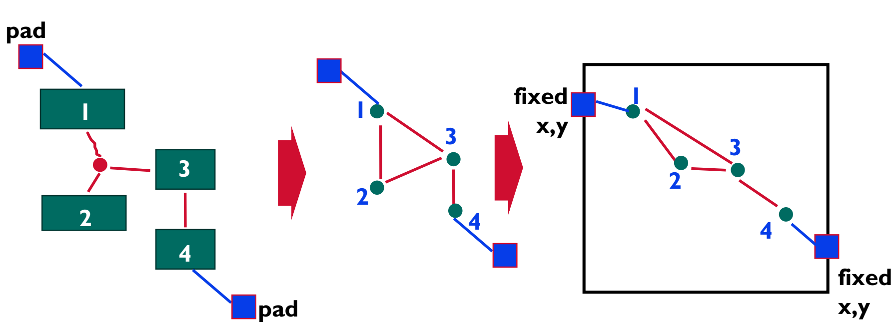
$\min x^T A x$ →

For placement

Add so-called "pad" constraint to force the solution to "spread out", pads represent *fixed* objects connected to wires; they can't move

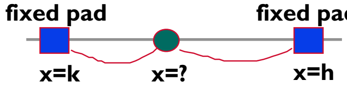
Pad Constraints: Basics

- Assume some objects are fixed, can't move, and that there are wires from these to the movable objects
  - Like pads are fixed around the periphery of a chip surface



Pad Constraints

- Back to the 1-var case to see how to optimize



Quadratic wirelen:

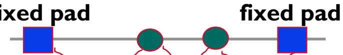
$$1*(x-k)^2 + 1*(x-h)^2$$
$$= x^2 - 2kx + k^2 + x^2 - 2hx + h^2$$
$$=$$

- Functional form
  - $1/2 \bullet a\,x^2 + b \bullet x + \text{constant}$

To minimize

Pad Constraints

- 2-var example
  - 2 variables (objects); arbitrary number of pads and nets



Quadratic wirelen:

$$1(x_1-k)^2 + 1(x_2-h)_2 + 1(x_1-x_2)^2$$

$$= [2x_1^2 + 2x_2^2 - 2x_1x_2] + [-2kx_1 - 2hx_2] + [k^2 + h_2]$$

- Functional form

$$\begin{bmatrix} x1 & x2 \end{bmatrix} \bullet \begin{bmatrix} A\ (2x2) \end{bmatrix} \bullet \begin{bmatrix} x1 \\ x2 \end{bmatrix} + 2 \bullet \begin{bmatrix} b1 & b2 \end{bmatrix} \bullet \begin{bmatrix} x1 \\ x2 \end{bmatrix} + \text{constant}$$

Written: $x^T A\, x + 2b^T x + \text{constant}$

Pad Constraints

- Concrete 2-variable example

$$\min \quad f = x_1^2 - x_1 x_2 + x_2^2 + b_1 x_1 + b_2 x_2 + \text{constant}$$

$$\frac{\partial f}{\partial x_1} = 0 =$$

$$\Rightarrow \begin{bmatrix} & \\ & \end{bmatrix} \begin{bmatrix} x1 \\ x2 \end{bmatrix} = \begin{bmatrix} \\ \\ \end{bmatrix}$$

$$\frac{\partial f}{\partial x_2} = 0 =$$

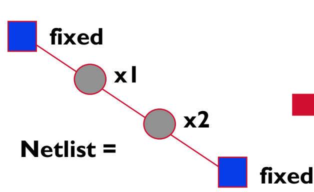
Pad Constraints

- Reformulate all this with matrices

$$\min \quad x1^2 - x1\ x2 + x2^2 + b1\ x1 + b2\ x2 + \text{constant}$$

Starting problem

$$\begin{aligned} 2x_1 \ -x_2 &= -b_1 \\ x_1 \ -2x_2 &= -b_2 \end{aligned}$$



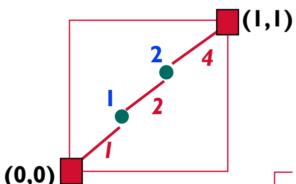
$$\Rightarrow C = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Solution

$$\begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -b_1 \\ -b_2 \end{bmatrix}$$

Be Careful...

- Gotta be *careful* about the 2s and 1/2s floating around here
  - Often see this formulated as $1/2 \; x^T A x + b^T x + \text{const}$
  - It's the same thing, just divide by the "2" in front of b, get a new const
  - Here is another simple example



quadratic wirelen for just x part is:

$$\underbrace{3\,x1^2 + 6\,x2^2 \;\; -4\,x1x2}\;\; \underbrace{-\,8\,x2}\; \underbrace{+\;1}$$

$$\begin{bmatrix} x1 & x2 \end{bmatrix} \boxed{\begin{bmatrix} 3 & -2 \\ -2 & 6 \end{bmatrix}} \begin{bmatrix} x1 \\ x2 \end{bmatrix} + 2 \boxed{\begin{bmatrix} 0 & -4 \end{bmatrix}} \begin{bmatrix} x1 \\ x2 \end{bmatrix} + 1$$

$$= x^T A\, x + 2b^T x + \text{const}$$

solution still: $Ax = -b = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$

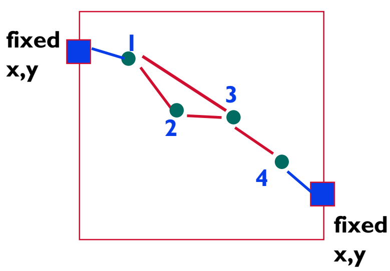
Conditions for Solution

- 1-var case
  - min $1/2\; a\, x^2 + b\, x + \text{const}$ has a solution $x = -b/a$ as long as a = positive
- General case
  - min $1/2 \bullet x^T A\, x + b^T x + \text{const}$ has a solution if A is positive definite
- Definition: Positive definite
  - Matrix A is positive definite if, for any vector x, $x^TAx > 0$ always

Useful result

Placing in (x,y) Plane

- How to handle the *2 dimensional* wirelen minimization task?
  - **Formulate the x variables and the y variables as 2 *separate* minimization problems; minimize them *separately***
  - **Why? There are *never* any x • y terms in the quadratic wirelength formula; OK to separate out the problem like this**



Formulation:

Min x quadratic wirelength

wirelen => Ax = -b

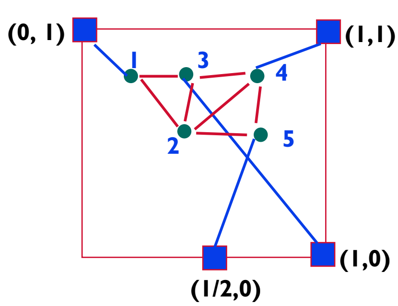
Min y quadratic wirelength

wirelen => A y = -b'

$$\Rightarrow \begin{pmatrix} x1 \\ x2 \\ \dots \\ xn \end{pmatrix} \quad \begin{pmatrix} y1 \\ y2 \\ \dots \\ yn \end{pmatrix}$$

Example

- 4 pads, a new 5 object netlist

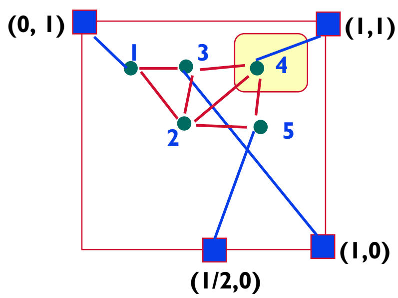


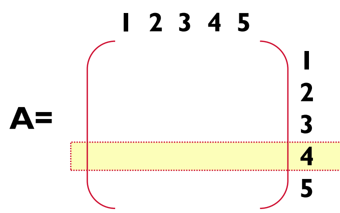
$$C = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad A = \begin{pmatrix} 3 & -1 & -1 & 0 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & 0 \\ 0 & -1 & -1 & 4 & -1 \\ 0 & -1 & 0 & -1 & 3 \end{pmatrix}$$

for x: A x = -b = $\begin{pmatrix} \\ \\ \\ \end{pmatrix}$ **y: A y =-b'=** $\begin{pmatrix} \\ \\ \\ \end{pmatrix}$

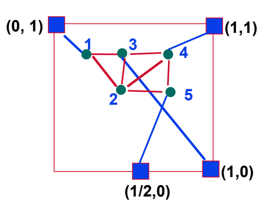
Some Subtleties Here

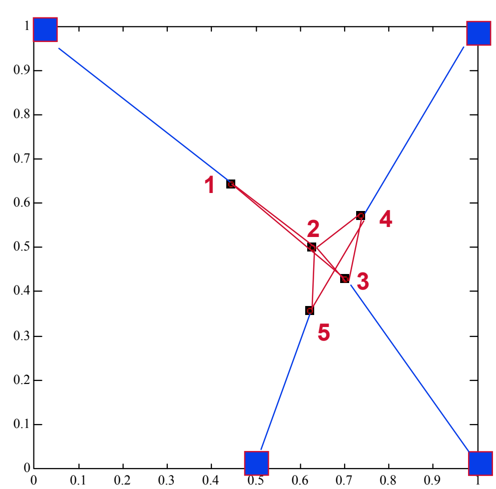
- Note: *not* precisely the same A as before
  - Start with the same C connectivity matrix among placeable objects
  - A is still (special diagonal) - $[c_{ij}]$
  - But you now have to account for the extra connections to the fixed pad objects for these elements on the diagonal; you sum weights on these wires as well as wires to movable objects





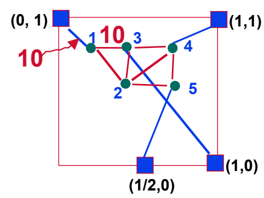
Placement Result (MATLAB)





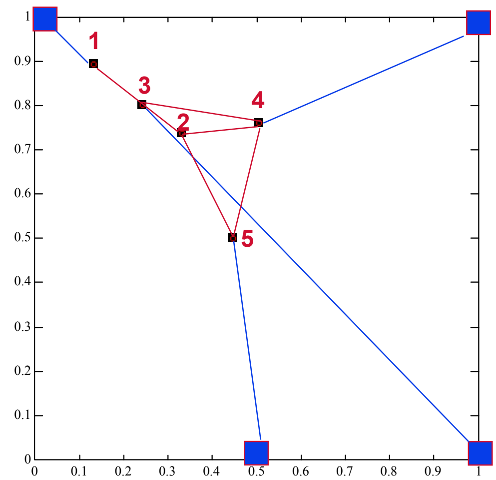
Another Placement Result

- *Change* weights: remember to change A and b vectors!



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

$$bx = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0.5 \end{pmatrix} \qquad by = \begin{pmatrix} 10 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$



Summary So Far…

- Direct placement
  - Dimensionless points, 2-point weighted wires
  - Minimize sum of squares of wire lengths
  - Has a direct-form representation of aggregate wirelength with functional form

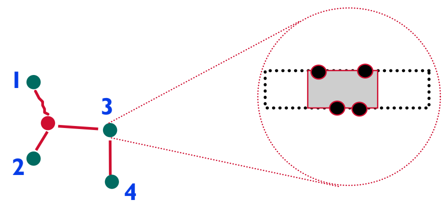
    $1/2 \cdot x^T A x + b^T x + \text{const}$ or equivalently

    $x^T A x + 2 b^T x + \text{const}$
  - …this is minimized at Ax = -b
  - Do x and y placements separately

- Open issues
  - These objects are really not dimensionless points, and we don't yet have a legal placement when this is finished
  - There are ways around these problems

Dealing with Shape

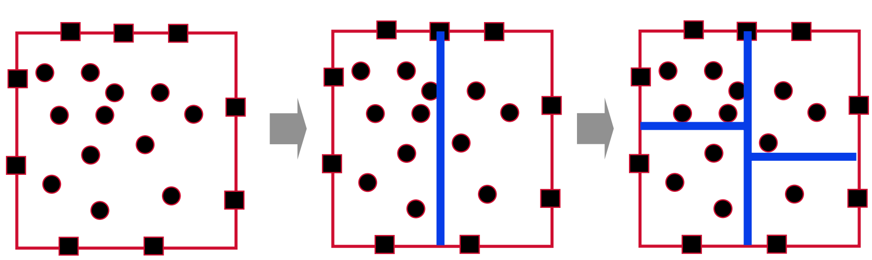
- The "points" really have shape, and need to be in rows



- Problems
  - Legalization: the objects almost certainly overlap after quadratic place.
  - How do we fix this…?
  - Several strategies; we will look informally at one

Strategy: PROUD

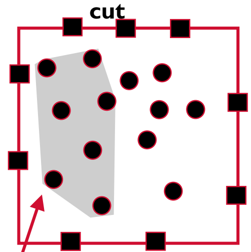
- Who
  - Ren Song Tsay, Ernest Kuh, Chi Ping Hsu, "PROUD: A Sea-Of-gates Placement Algorithm," IEEE Design & Test of Computers, Dec 1988.
- What
  - *Recursive* legalization by partitioning & refining
  - Use quadratic placement as starting point for a recursive strategy



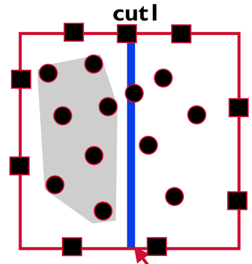
PROUD: Mechanics

Mechanics

Balancing cut



Physical cut1



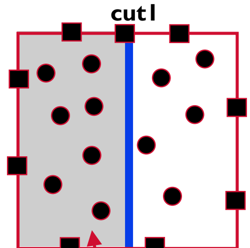
Perform the first quadratic placement, inside this region. Via sorting gates on X, decide which gates need to be on the left side (want ~1/2 on left)

Place a *physical* cutline at the X center of the region; We will now reformulate a new placement problem just to re-do the gates on the left.

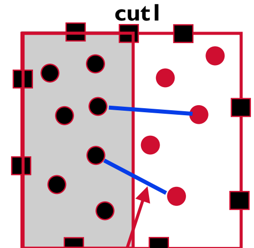
PROUD: Mechanics

Mechanics

Physical cut1



Physical cut1

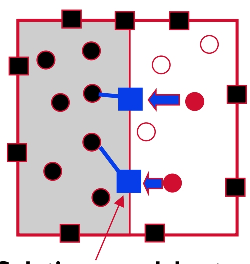


Focus on the gates inside the shaded region on the left side of the cut.

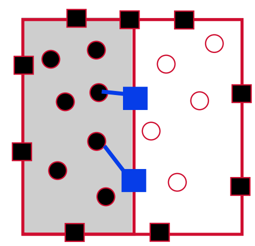
Big question:
How do we model the fact that wires connect to gates on the right? We can't just ignore these when we re-place gates on left in their own smaller (shaded) region!

PROUD: Mechanics

- **Idea**
  - We model physical effect of wires that go "outside" our left-side region via wires to *psuedo-pins* which represent "approximately" where these wires need to go
  - Now, we can solve the left-side *alone*, *again*, to get the next cut



Solution: model gates on the right as new, "fake" pins on the left.



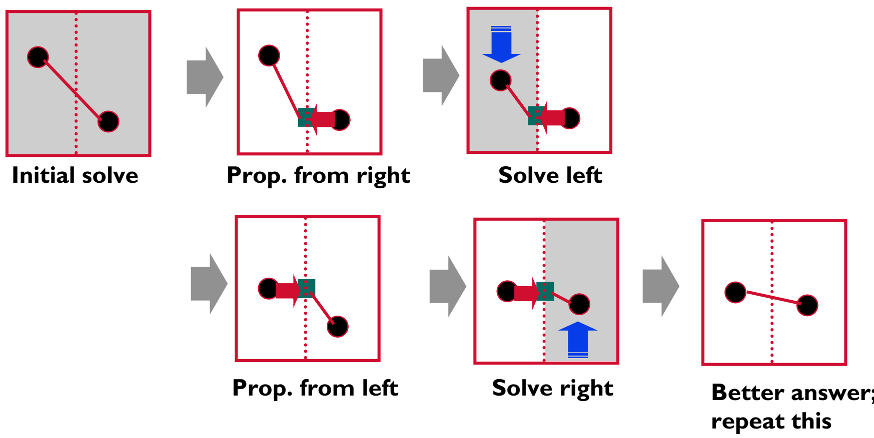
Process is called "pseudo pin propagation"

PROUD: Processing the Subregions

- **Idea**
  - Pick one of the regions R1 (eg, the left one) of cut hierarchy
  - Propagate pseudo-pins to R1's cut boundary
  - Solve (quadratic re-place) region R1
  - Now, pick NEXT region, R2
  - Propagate pseudo-pins to R2's cut boundary
    - Note, some of these may be due to the most recent gate placement motions of solving R1
  - Solve (quadratic re-place) region R2
  - Pick NEXT region, R3, etc
- **Iteration**
  - Tsay says he goes around this *whole* loop 3-5 times at each level of the hierarchy
  - ...i.e., we "propagate & replace" each region 3-5 times, which allows effects of global movements to be "felt" by everybody

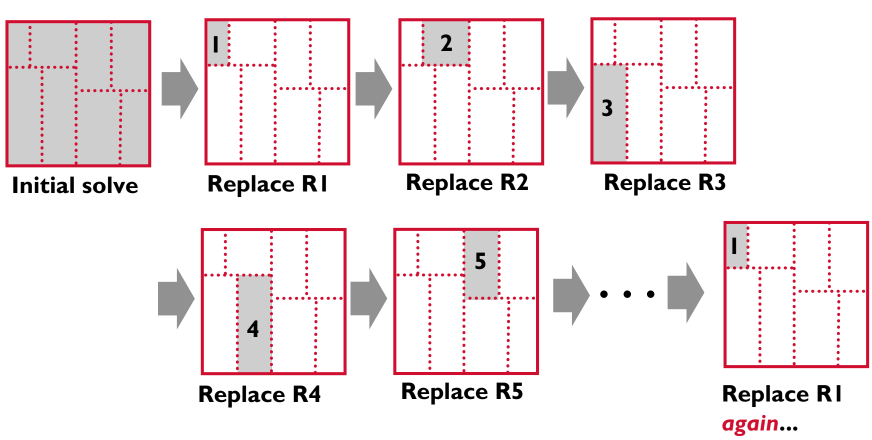
PROUD: Iteration

- Why do we repeat this operation?
  - Ping-pong back and forth thru subregions?
  - Gives objects in region a chance to "influence" other regions



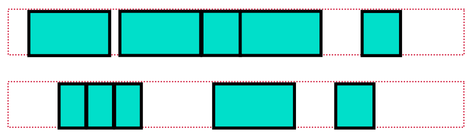
PROUD: Iteration

- Easier to see when there are many regions at the level of the cut hierarchy



PROUD: Finalizing Placement

- Does this create a legal placement buy itself? No
  - It does a pretty good job of global placement, and guaranteeing that you do not put more modules in any region than the area allows
  - But, it cannot really force individual gates into cell rows



- Solution
  - Don't partition all the way down to individual objects
  - Go down to regions with many (10s) of objects, snap onto row grid, and then do iterative improvement based on swaps of modules
  - People do annealing down here, among other things…

PROUD: Summary

- Quadratic place
  - To get the initial placement
  - Again, on each region of the cut hierarchy, to help legalize the region, to move objects to good place after they are forced to go in a region
- Recursive cutting
  - To force ~ right number of placeable objects in each region
  - Uses quadratic placement and psuedo-pins to do each region
- Final legalization
  - Run above till each region has few tens of cells
  - Then do iterative improvement

Summary

- **Iterative improvement placement by annealing**
  - "The" approach in the 1980s; runs out of gas at a few 100,000 gates
- **Recursive mincut placers**
  - Based on clever, iterative improvement partitioning
  - Coming back into style today; very good for very large ASICs
- **Quadratic direct placement**
  - Point-based, 2-point-wires; can minimize quadratic wirelen exactly, fast
  - But, placement not really legal (overlaps); lots of work *here*.
- **Today**
  - Mix of quadratic and mincut techniques to do "gross" placement; iterative improvement "local refinement" to get legal final placement
  - This is really how people really do millions of gates today…