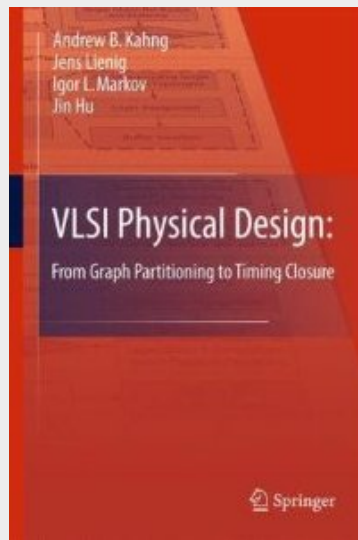


VLSI Physical Design: From Graph Partitioning to Timing Closure

Chapter 4 – Global and Detailed Placement



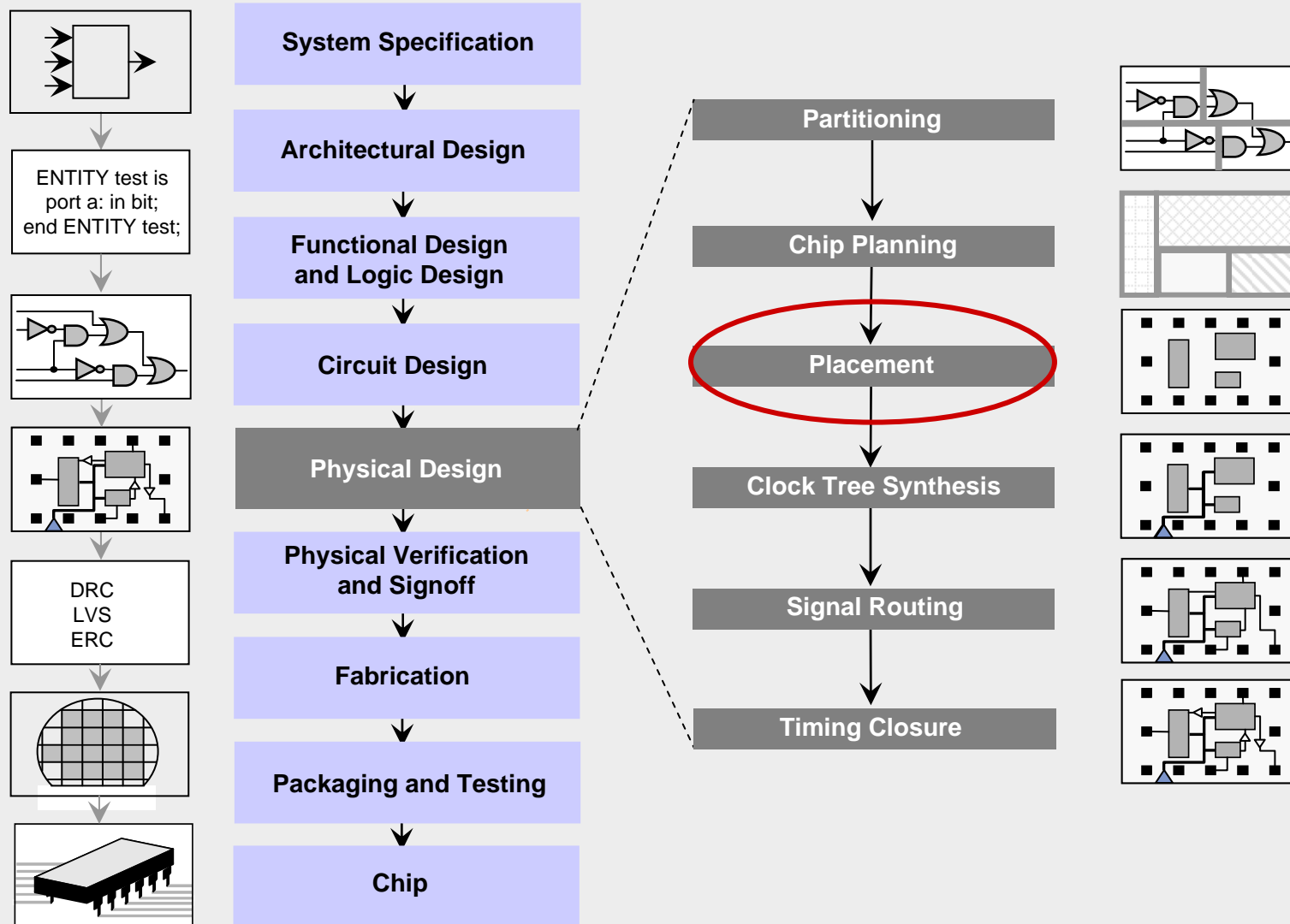
Original Authors:

Andrew B. Kahng, Jens Lienig, Igor L. Markov, Jin Hu

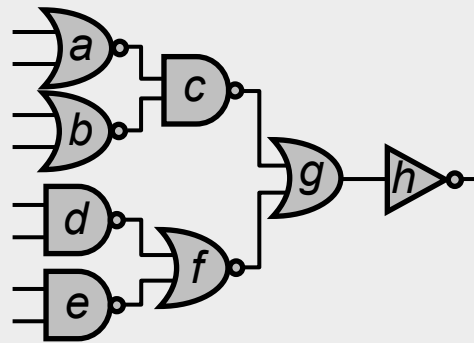
Chapter 4 – Global and Detailed Placement

- 4.1 Introduction
- 4.2 Optimization Objectives
- 4.3 Global Placement
 - 4.3.1 Min-Cut Placement
 - 4.3.2 Analytic Placement
 - 4.3.3 Simulated Annealing
 - 4.3.4 Modern Placement Algorithms
- 4.4 Legalization and Detailed Placement

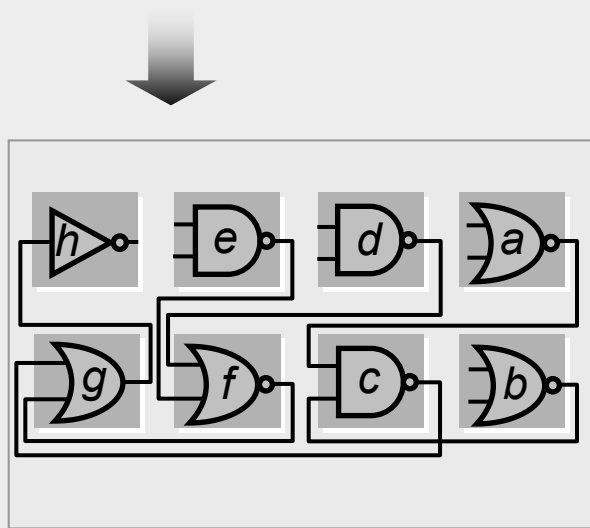
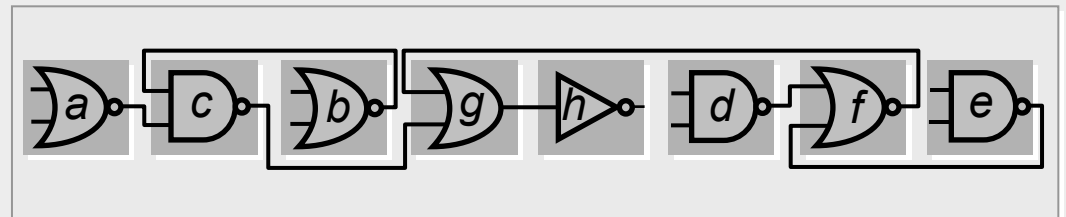
4.1 Introduction



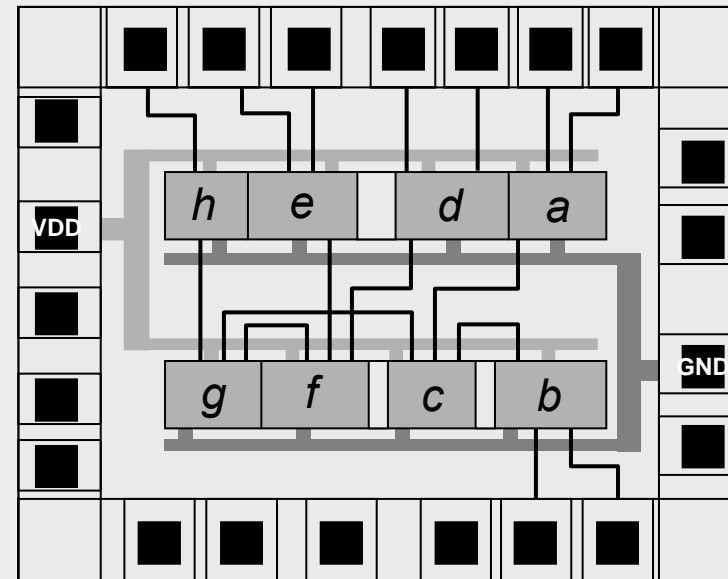
4.1 Introduction



Linear Placement



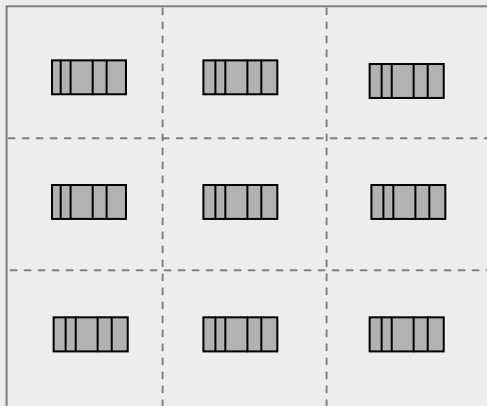
2D Placement



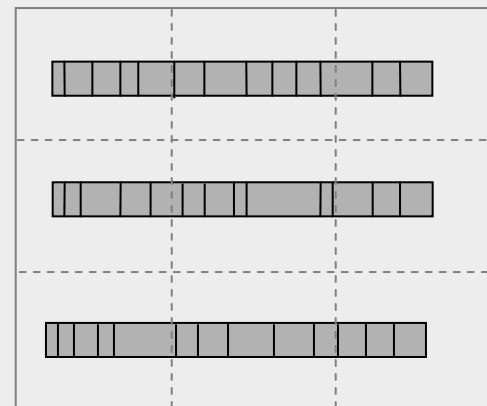
Placement and Routing with Standard Cells

4.1 Introduction

Global
Placement

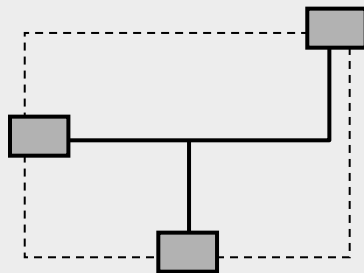


Detailed
Placement

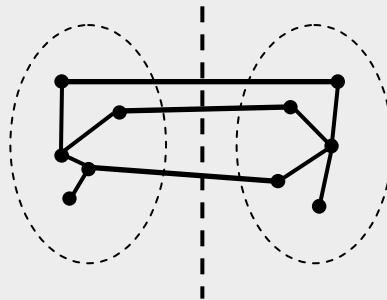


4.2 Optimization Objectives

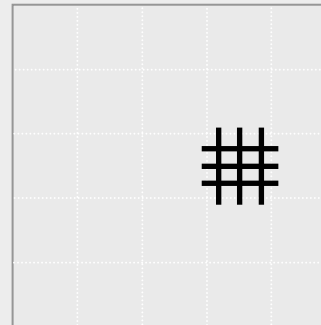
Total Wirelength



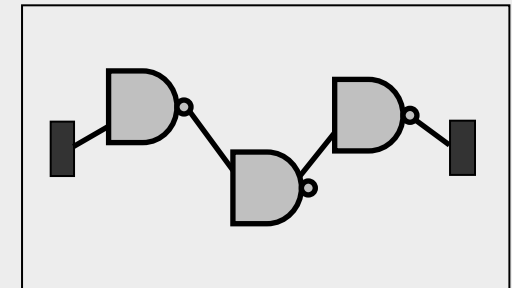
Number of Cut Nets



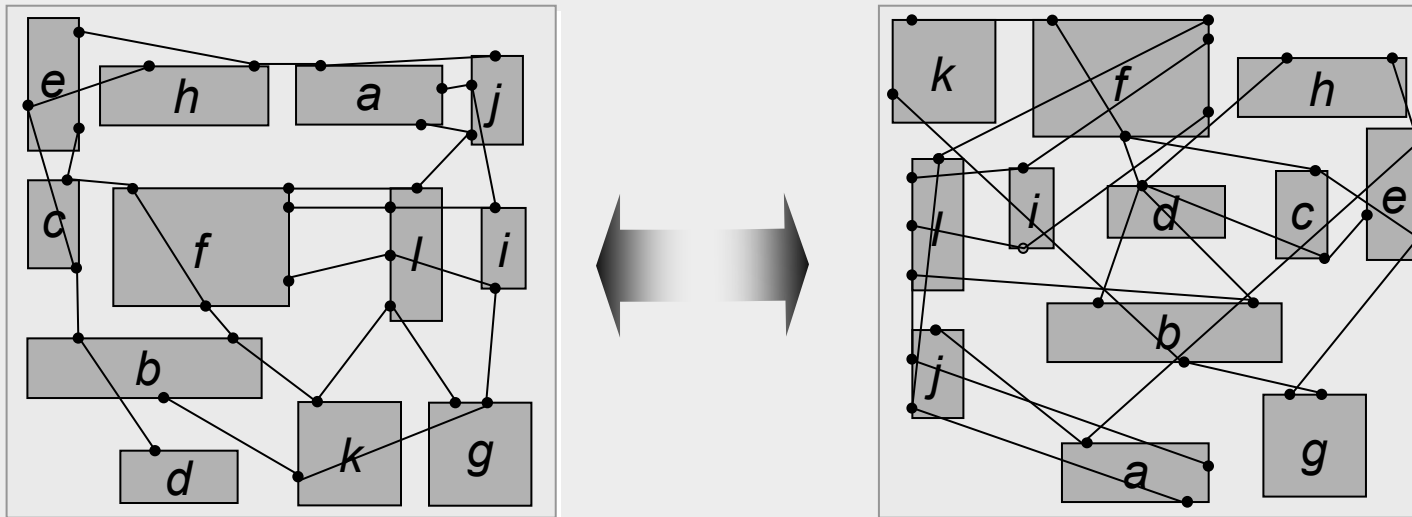
Wire Congestion



Signal Delay



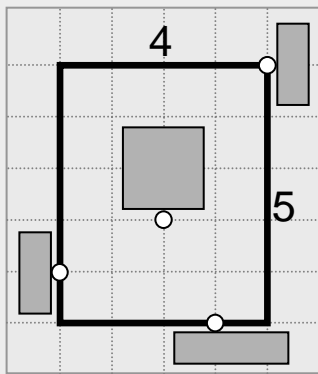
4.2 Optimization Objectives – Total Wirelength



4.2 Optimization Objectives – Total Wirelength

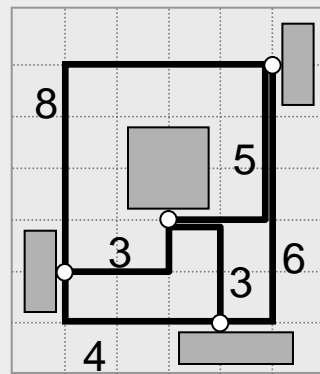
Wirelength estimation for a given placement

Half-perimeter
wirelength
(HPWL)



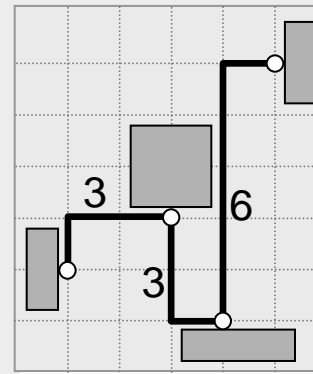
HPWL = 9

Complete
graph
(clique)



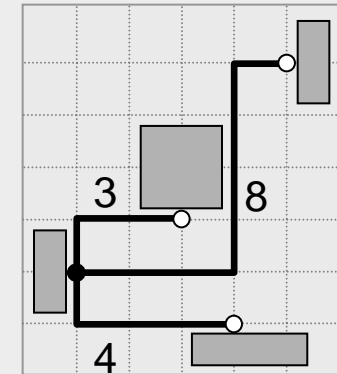
Clique Length =
 $(2/p)\sum_{e \in \text{clique}} d_M(e) = 14.5$

Monotone
chain



Chain Length = 12

Star model

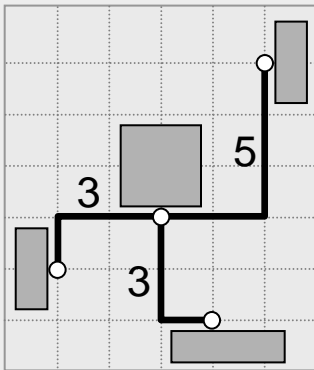


Star Length = 15

4.2 Optimization Objectives – Total Wirelength

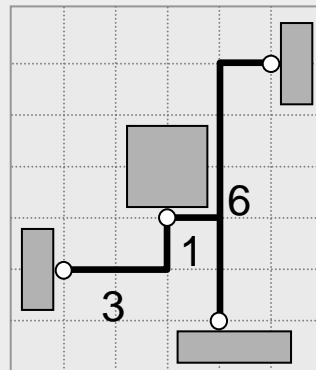
Wirelength estimation for a given placement (cont'd.)

Rectilinear
minimum
spanning
tree (RMST)



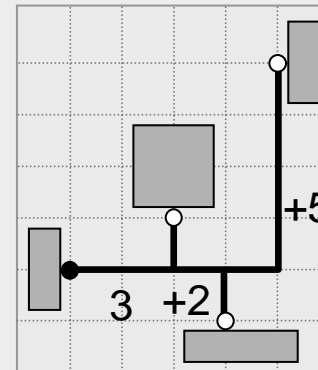
RMST Length = 11

Rectilinear
Steiner
minimum
tree (RSMT)



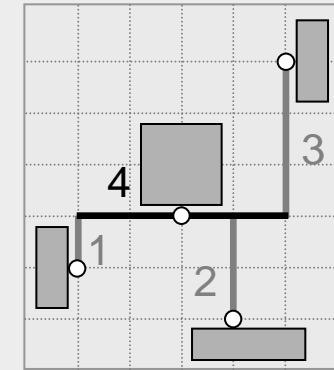
RSMT Length = 10

Rectilinear
Steiner
arborescence
model (RSA)



RSA Length = 10

Single-trunk
Steiner
tree (STST)



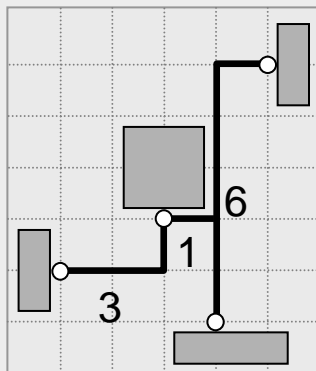
STST Length = 10

4.2 Optimization Objectives – Total Wirelength

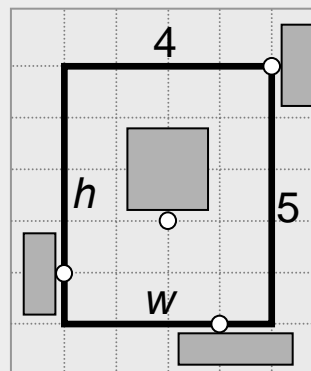
Wirelength estimation for a given placement (cont'd.)

Preferred method: Half-perimeter wirelength (HPWL)

- Fast (order of magnitude faster than RSMT)
- Equal to length of RSMT for 2- and 3-pin nets
- Margin of error for real circuits approx. 8% [Chu, ICCAD 04]



RSMT Length = 10



HPWL = 9

$$L_{\text{HPWL}} = w + h$$

4.2 Optimization Objectives – Total Wirelength

Total wirelength with net weights (weighted wirelength)

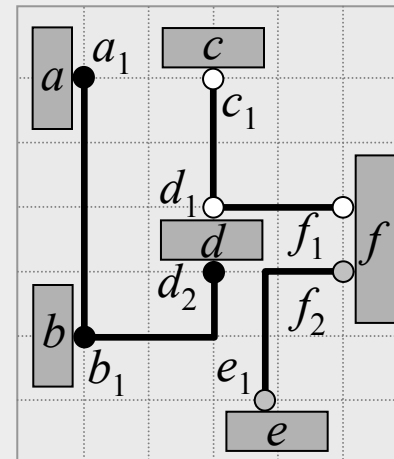
- For a placement P , an estimate of total weighted wirelength is

$$L(P) = \sum_{net \in P} w(net) \cdot L(net)$$

where $w(net)$ is the weight of net , and $L(net)$ is the estimated wirelength of net .

- Example:

Nets	Weights
$N_1 = (a_1, b_1, d_2)$	$w(N_1) = 2$
$N_2 = (c_1, d_1, f_1)$	$w(N_2) = 4$
$N_3 = (e_1, f_2)$	$w(N_3) = 1$



$$L(P) = \sum_{net \in P} w(net) \cdot L(net) = 2 \cdot 7 + 4 \cdot 4 + 1 \cdot 3 = 33$$

4.2 Optimization Objectives – Number of Cut Nets

Cut sizes of a placement

- To improve total wirelength of a placement P , separately calculate the number of crossings of global vertical and horizontal cutlines, and minimize

$$L(P) = \sum_{v \in V_P} \Psi_P(v) + \sum_{h \in H_P} \Psi_P(h)$$

where $\Psi_P(\text{cut})$ be the set of nets cut by a cutline cut

4.2 Optimization Objectives – Number of Cut Nets

Cut sizes of a placement

- Example:

Nets

$$N_1 = (a_1, b_1, d_2)$$

$$N_2 = (c_1, d_1, f_1)$$

$$N_3 = (e_1, f_2)$$

- Cut values for each global cutline

$$\psi_P(v_1) = 1 \quad \psi_P(v_2) = 2$$

$$\psi_P(h_1) = 3 \quad \psi_P(h_2) = 2$$

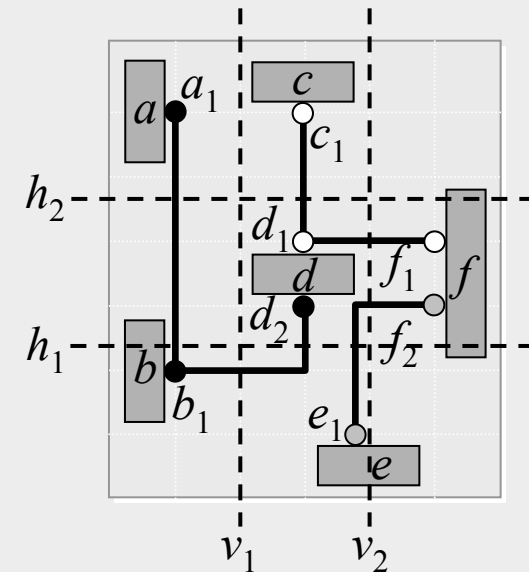
- Total number of crossings in P

$$\psi_P(v_1) + \psi_P(v_2) + \psi_P(h_1) + \psi_P(h_2) = 1 + 2 + 3 + 2 = 8$$

- Cut sizes

$$X(P) = \max(\psi_P(v_1), \psi_P(v_2)) = \max(1, 2) = 2$$

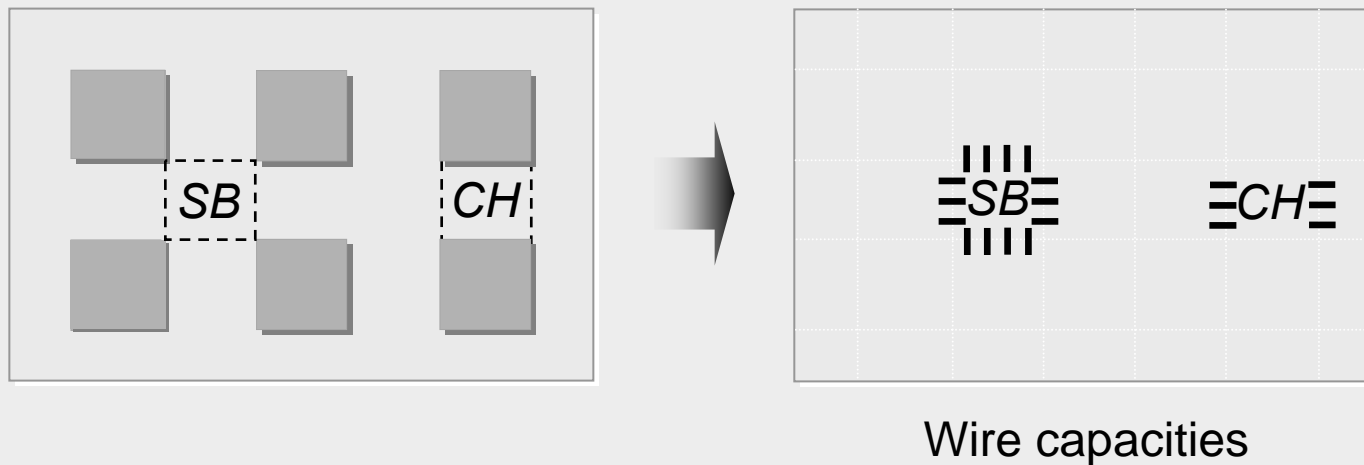
$$Y(P) = \max(\psi_P(h_1), \psi_P(h_2)) = \max(3, 2) = 3$$



4.2 Optimization Objectives – Wire Congestion

Routing congestion of a placement

- Ratio of demand for routing tracks to the supply of available routing tracks
- Estimated by the number of nets that pass through the boundaries of individual routing regions



4.2 Optimization Objectives – Wire Congestion

Routing congestion of a placement

- Formally, the local wire density $\phi_P(e)$ of an edge e between two neighboring grid cells is

$$\phi_P(e) = \frac{\eta_P(e)}{\sigma_P(e)}$$

where $\eta_P(e)$ is the estimated number of nets that cross e and $\sigma_P(e)$ is the maximum number of nets that can cross e

- If $\phi_P(e) > 1$, then too many nets are estimated to cross e , making P more likely to be unroutable.
- The wire density of P is $\Phi(P) = \max_{e \in E}(\phi_P(e))$

where E is the set of all edges

- If $\Phi(P) \leq 1$, then the design is estimated to be fully routable, otherwise routing will need to detour some nets through less-congested edges

4.2 Optimization Objectives – Wire Congestion

Wire Density of a placement

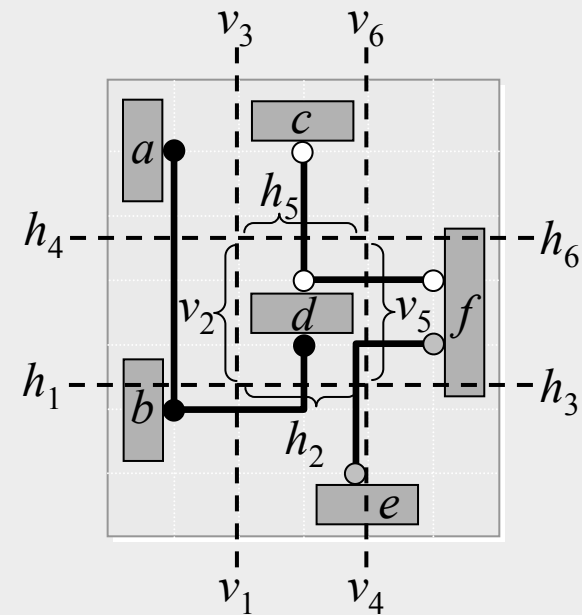
$$\begin{array}{ll} \eta_P(h_1) = 1 & \eta_P(v_1) = 1 \\ \eta_P(h_2) = 2 & \eta_P(v_2) = 0 \\ \eta_P(h_3) = 0 & \eta_P(v_3) = 0 \\ \eta_P(h_4) = 1 & \eta_P(v_4) = 0 \\ \eta_P(h_5) = 1 & \eta_P(v_5) = 2 \\ \eta_P(h_6) = 0 & \eta_P(v_6) = 0 \end{array}$$

Maximum: $\eta_P(e) = 2$

$$\Phi(P) = \frac{\eta_P(e)}{\sigma_P(e)} = \frac{2}{3}$$



Routable



4.2 Optimization Objectives – Signal Delay

Circuit timing of a placement

- Static timing analysis using actual arrival time (AAT) and required arrival time (RAT)
 - $AAT(v)$ represents the latest transition time at a given node v measured from the beginning of the clock cycle
 - $RAT(v)$ represents the time by which the latest transition at v must complete in order for the circuit to operate correctly within a given clock cycle.
- For correct operation of the chip with respect to setup (maximum path delay) constraints, it is required that $AAT(v) \leq RAT(v)$.

4.1 Introduction

4.2 Optimization Objectives

→ 4.3 Global Placement
4.3.1 Min-Cut Placement
4.3.2 Analytic Placement
4.3.3 Simulated Annealing
4.3.4 Modern Placement Algorithms

4.4 Legalization and Detailed Placement

Global Placement

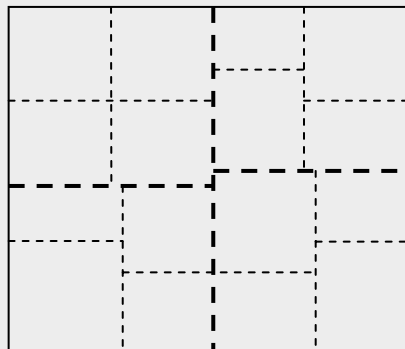
- **Partitioning-based algorithms:**
 - The netlist and the layout are divided into smaller sub-netlists and sub-regions, respectively
 - Process is repeated until each sub-netlist and sub-region is small enough to be handled optimally
 - Detailed placement often performed by optimal solvers, facilitating a natural transition from global placement to detailed placement
 - Example: min-cut placement
- **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
- **Stochastic algorithms:**
 - Randomized moves that allow hill-climbing are used to optimize the cost function
 - Example: simulated annealing

Global Placement

Partitioning-based



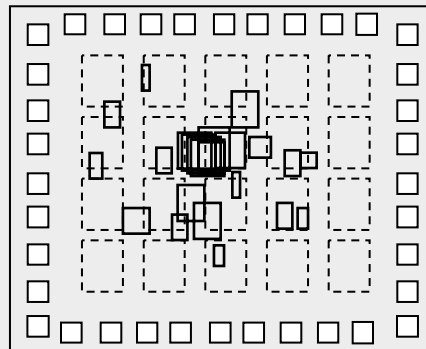
Min-cut placement



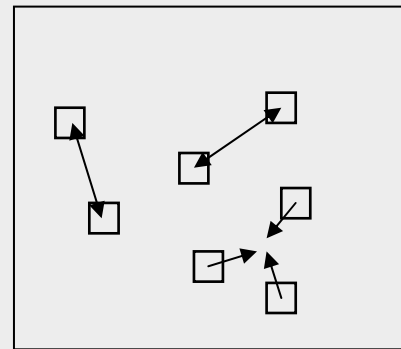
Analytic



Quadratic placement



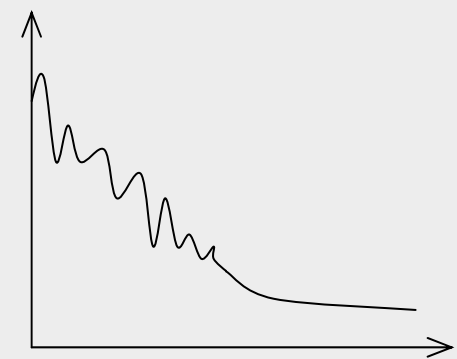
Force-directed placement



Stochastic



Simulated annealing

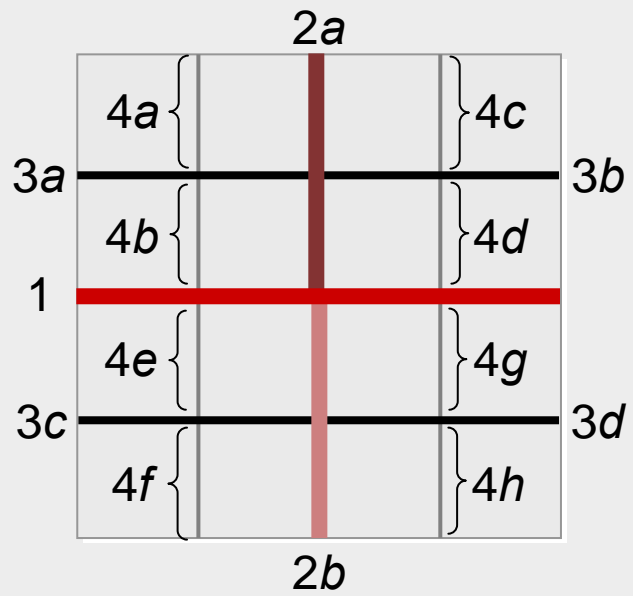


4.3.1 Min-Cut Placement

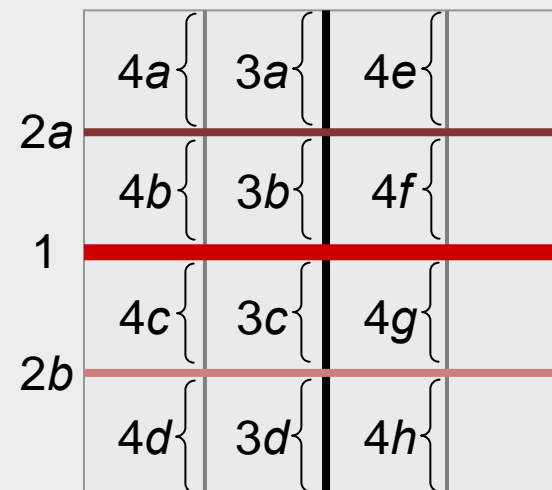
- Uses partitioning algorithms to divide (1) the netlist and (2) the layout region into smaller sub-netlists and sub-regions
- Conceptually, each sub-region is assigned a portion of the original netlist
- Each cut heuristically minimizes the number of cut nets using, for example,
 - Kernighan-Lin (KL)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

4.3.1 Min-Cut Placement

Alternating cutline directions



Repeating cutline directions



4.3.1 Min-Cut Placement

Input: netlist *Netlist*, layout area *LA*, minimum number of cells per region *cells_min*

Output: placement *P*

$P = \emptyset$

regions = ASSIGN(*Netlist*,*LA*)

// assign netlist to layout area

while (*regions* $\neq \emptyset$)

// while regions still not placed

region = FIRST_ELEMENT(*regions*)

// first element in *regions*

 REMOVE(*regions*, *region*)

// remove first element of *regions*

if (*region* contains more than *cell_min* cells)

 (*sr1*,*sr2*) = BISECT(*region*)

// divide *region* into two subregions

// *sr1* and *sr2*, obtaining the sub-

// netlists and sub-areas

 ADD_TO_END(*regions*,*sr1*)

// add *sr1* to the end of *regions*

 ADD_TO_END(*regions*,*sr2*)

// add *sr2* to the end of *regions*

else

 PLACE(*region*)

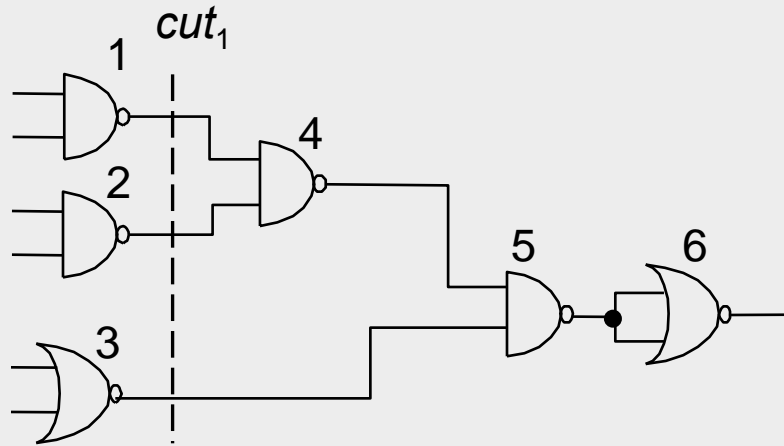
// place *region*

 ADD(*P*,*region*)

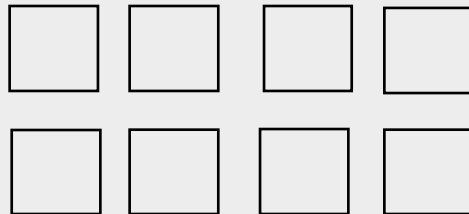
// add *region* to *P*

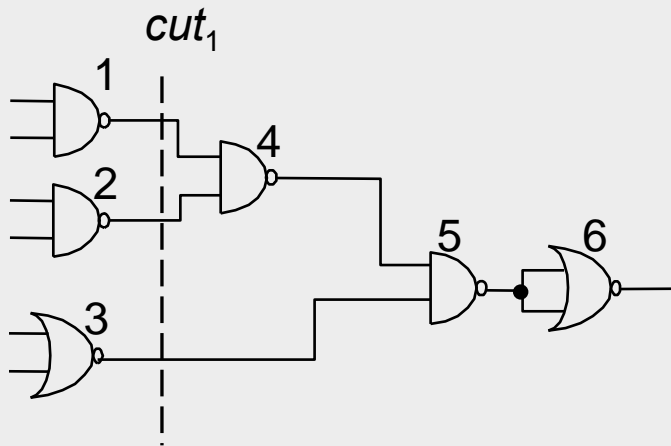
4.3.1 Min-Cut Placement – Example

Given:

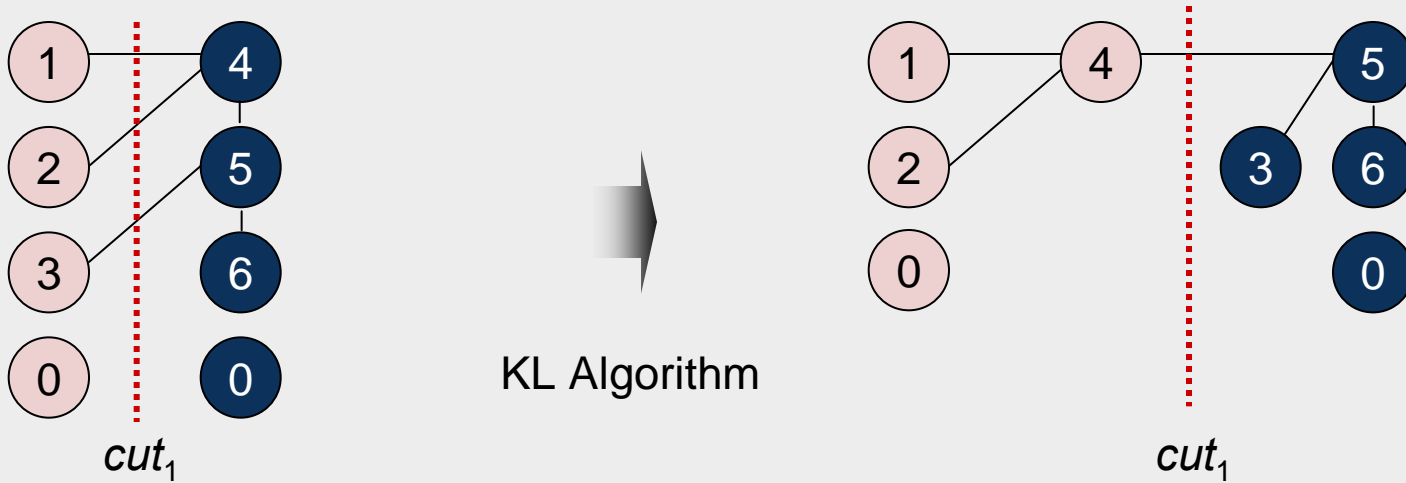


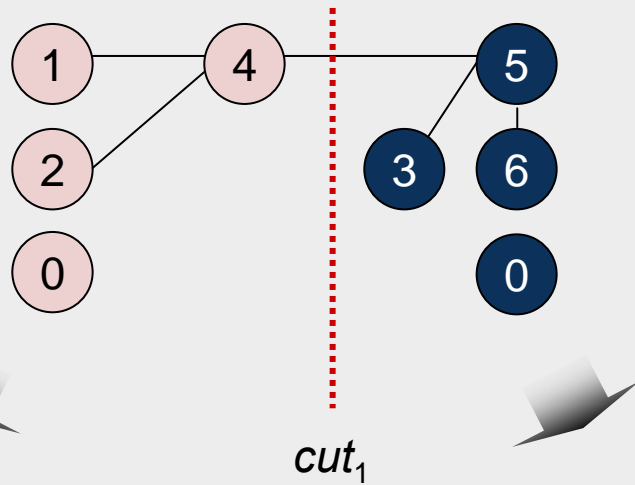
Task: 4 x 2 placement with minimum wirelength using alternative cutline directions and the KL algorithm





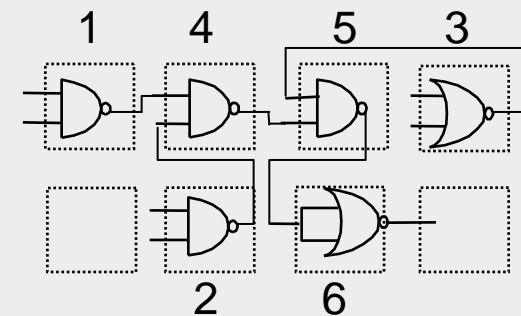
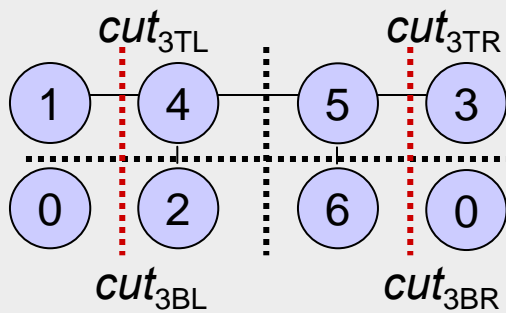
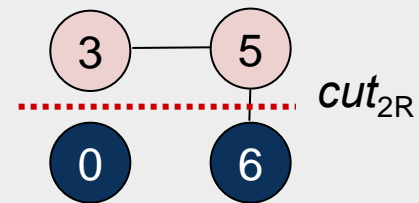
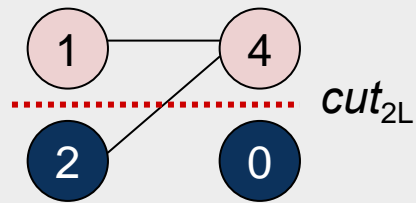
Vertical cut cut_1 : $L=\{1,2,3\}$, $R=\{4,5,6\}$



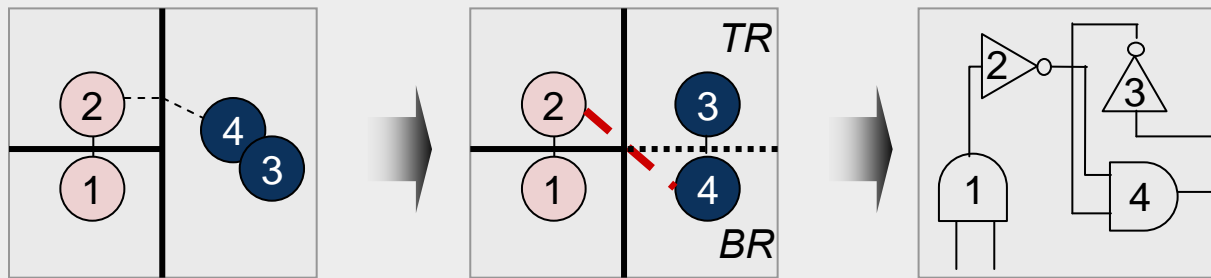


Horizontal cut cut_{2L} : $T=\{1,4\}$, $B=\{2,0\}$

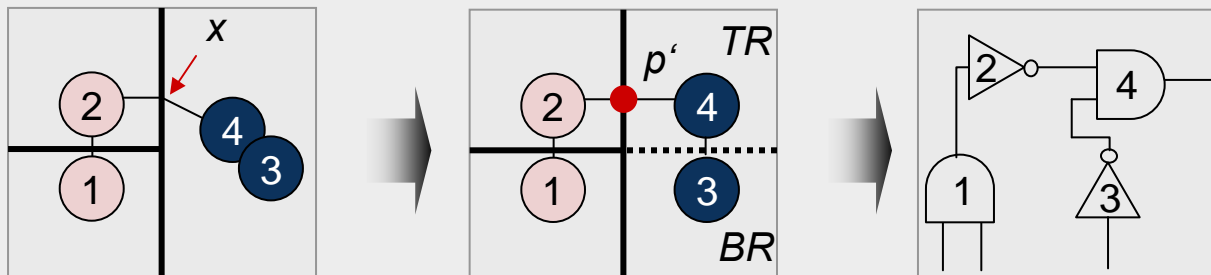
Horizontal cut cut_{2R} : $T=\{3,5\}$, $B=\{6,0\}$



4.3.1 Min-Cut Placement – Terminal Propagation



- Terminal Propagation
 - External connections are represented by artificial connection points on the cutline
 - Dummy nodes in hypergraphs



4.3.1 Min-Cut Placement

- 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

4.3.2 Analytic Placement – Quadratic Placement

- Objective function is quadratic; sum of (weighted) **squared Euclidean distance** represents placement objective function

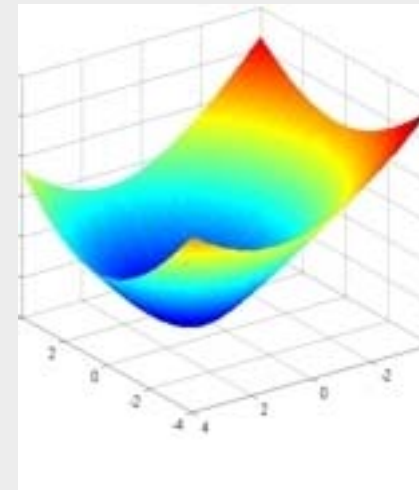
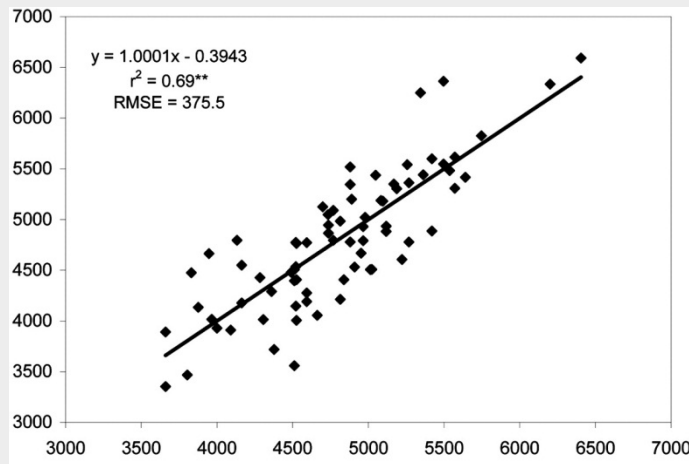
$$L(P) = \frac{1}{2} \sum_{i,j=1}^n c_{ij} \left((x_i - x_j)^2 + (y_i - y_j)^2 \right)$$

where n is the total number of cells, and $c(i,j)$ is the connection cost between cells i and j .

- Only two-point-connections
- Minimize objective function by equating its derivative to zero which reduces to solving a system of linear equations

4.3.2 Analytic Placement – Quadratic Placement

- Similar to Least-Mean-Square Method (root mean square)
- Build error function with analytic form: $E(a,b) = \sum (a \cdot x_i + b - y_i)^2$



4.3.2 Analytic Placement – Quadratic Placement

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

where n is the total number of cells, and $c(i,j)$ is the connection cost between cells i and j .

- Each dimension can be considered independently:

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

- 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

4.3.2 Analytic Placement – Quadratic Placement


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


where n is the total number of cells, and $c(i,j)$ is the connection cost between cells i and j .

- Each dimension can be considered independently:

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

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


$$\frac{\partial L_x(P)}{\partial X} = AX - b_x = 0$$


$$\frac{\partial L_y(P)}{\partial Y} = AY - b_y = 0$$

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 .

4.3.2 Analytic Placement – Quadratic Placement


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


where n is the total number of cells, and $c(i,j)$ is the connection cost between cells i and j .

- Each dimension can be considered independently:

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

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

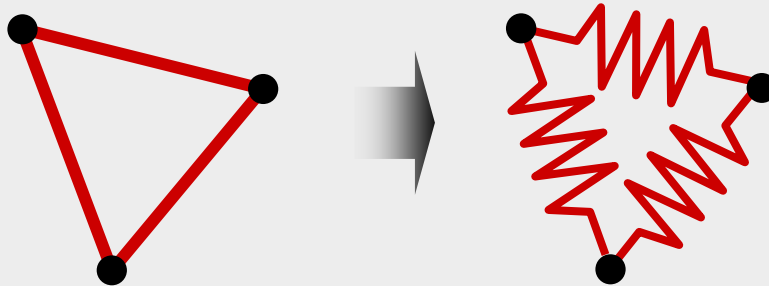

$$\frac{\partial L_x(P)}{\partial X} = AX - b_x = 0$$


$$\frac{\partial L_y(P)}{\partial Y} = AY - b_y = 0$$

- System of linear equations for which iterative numerical methods can be used to find a solution

4.3.2 Analytic Placement – Quadratic Placement

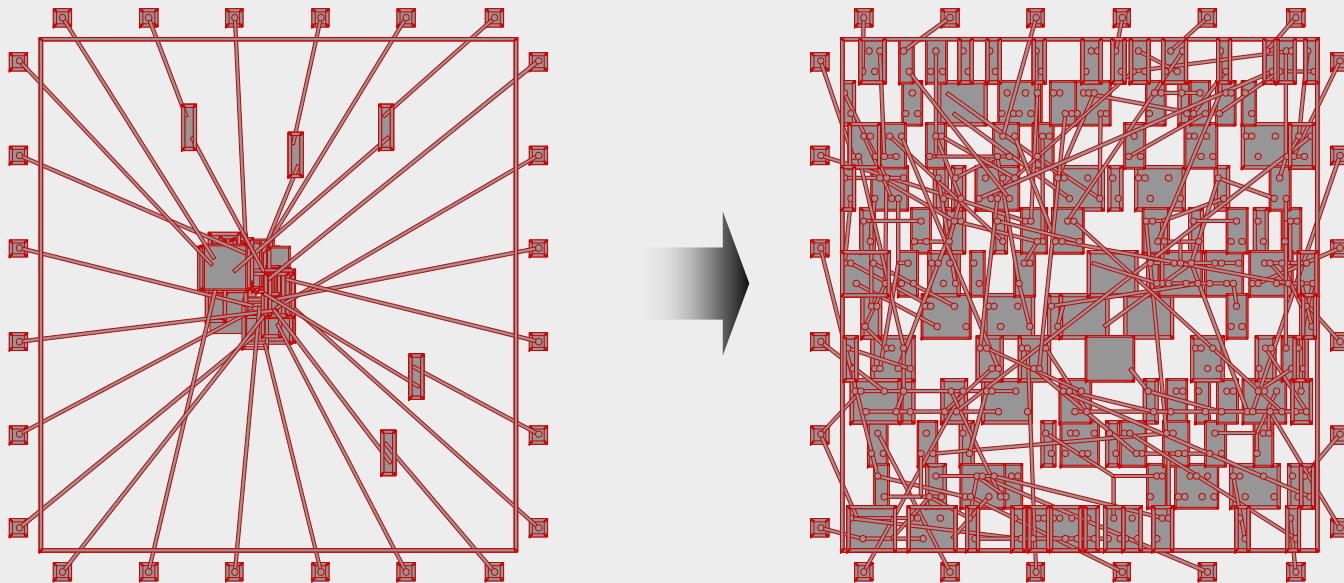
- Mechanical analogy: mass-spring system



- Squared Euclidean distance is proportional to the energy of a spring between these points
 - Quadratic objective function represents total energy of the spring system; for each movable object, the x (y) partial derivative represents the total force acting on that object
 - Setting the forces of the nets to zero, an equilibrium state is mathematically modeled that is characterized by zero forces acting on each movable object
 - At the end, all springs are in a force equilibrium with a minimal total spring energy; this equilibrium represents the minimal sum of squared wirelength
- Result: many cell overlaps

4.3.2 Analytic Placement – Quadratic Placement

- Second stage of quadratic placers: cells are spread out to remove overlaps
- Methods:
 - Adding fake nets that pull cells away from dense regions toward anchors
 - Geometric sorting and scaling
 - Repulsion forces, etc.

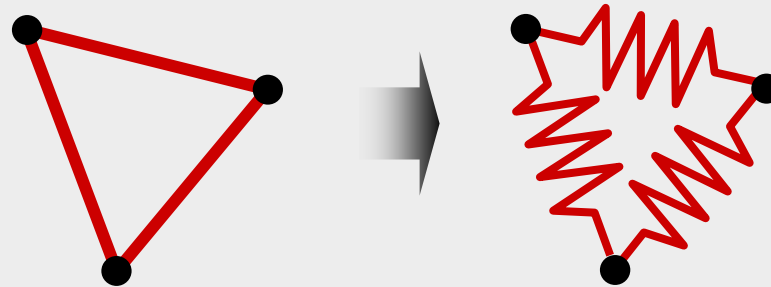


4.3.2 Analytic Placement – Quadratic Placement

- Advantages:
 - Captures the placement problem concisely in mathematical terms
 - Leverages efficient algorithms from numerical analysis and available software
 - Can be applied to large circuits without netlist clustering (flat)
 - Stability: small changes in the input do not lead to large changes in the output
- Disadvantages:
 - Connections to fixed objects are necessary: I/O pads, pins of fixed macros, etc.

4.3.2 Analytic Placement – Force-directed Placement

- Cells and wires are modeled using the mechanical analogy of a mass-spring system, i.e., masses connected to Hooke's-Law springs



- Attraction force between cells is directly proportional to their distance
- Cells will eventually settle in a **force equilibrium** → minimized wirelength

4.3.2 Analytic Placement – Force-directed Placement

- Given two connected cells a and b , the attraction force \vec{F}_{ab} exerted on a by b is

$$\vec{F}_{ab} = c(a,b) \cdot (\vec{b} - \vec{a})$$

where

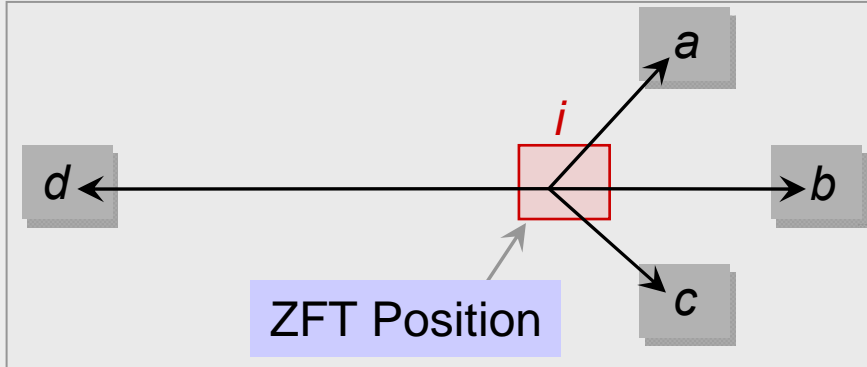
- $c(a,b)$ is the connection weight (priority) between cells a and b , and
 - $(\vec{b} - \vec{a})$ is the vector difference of the positions of a and b in the Euclidean plane
- The sum of forces exerted on a cell i connected to other cells $1 \dots j$ is

$$\vec{F}_i = \sum_{c(i,j) \neq 0} \vec{F}_{ij}$$

- Zero-force target (ZFT):** position that minimizes this sum of forces

4.3.2 Analytic Placement – Force-directed Placement

Zero-Force-Target (ZFT) position of cell i



$$\min \vec{F}_i = c(i,a) \cdot (\vec{a} - \vec{i}) + c(i,b) \cdot (\vec{b} - \vec{i}) + c(i,c) \cdot (\vec{c} - \vec{i}) + c(i,d) \cdot (\vec{d} - \vec{i})$$

4.3.2 Analytic Placement – Force-directed Placement

Basic force-directed placement

- Iteratively moves all cells to their respective ZFT positions
- x- and y-direction forces are set to zero:

$$\sum_{c(i,j) \neq 0} c(i,j) \cdot (x_j^0 - x_i^0) = 0 \quad \sum_{c(i,j) \neq 0} c(i,j) \cdot (y_j^0 - y_i^0) = 0$$

- Rearranging the variables to solve for x_i^0 and y_i^0 yields

$$x_i^0 = \frac{\sum_{c(i,j) \neq 0} c(i,j) \cdot x_j^0}{\sum_{c(i,j) \neq 0} c(i,j)}$$

$$y_i^0 = \frac{\sum_{c(i,j) \neq 0} c(i,j) \cdot y_j^0}{\sum_{c(i,j) \neq 0} c(i,j)}$$

Computation of
ZFT position of cell i
connected with
cells 1 ... j

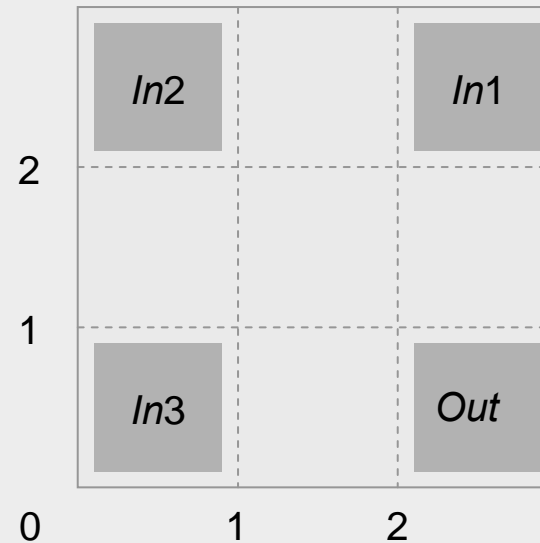
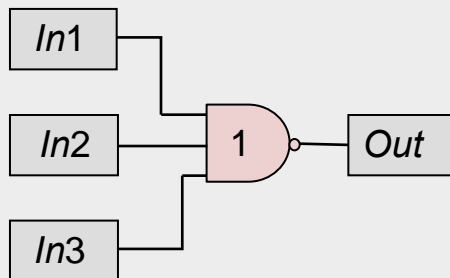
4.3.2 Analytic Placement – Force-directed Placement

Example: ZFT position

Given:

- Circuit with NAND gate 1 and four I/O pads on a 3 x 3 grid
- Pad positions: $In1$ (2,2), $In2$ (0,2), $In3$ (0,0), Out (2,0)
- Weighted connections: $c(a,In1) = 8$, $c(a,In2) = 10$, $c(a,In3) = 2$, $c(a,Out) = 2$

Task: find the ZFT position of cell a



4.3.2 Analytic Placement – Force-directed Placement

Example: ZFT position

Given:

- Circuit with NAND gate 1 and four I/O pads on a 3 x 3 grid
- Pad positions: $In1$ (2,2), $In2$ (0,2), $In3$ (0,0), Out (2,0)

Solution:

$$x_a^0 = \frac{\sum_{c(i,j) \neq 0} c(a,j) \cdot x_j^0}{\sum_{c(i,j) \neq 0} c(a,j)} = \frac{c(a, In1) \cdot x_{In1} + c(a, In2) \cdot x_{In2} + c(a, In3) \cdot x_{In3} + c(a, Out) \cdot x_{Out}}{c(a, In1) + c(a, In2) + c(a, In3) + c(a, Out)} = \frac{8 \cdot 2 + 10 \cdot 0 + 2 \cdot 0 + 2 \cdot 2}{8 + 10 + 2 + 2} = \frac{20}{22} \approx 0.9$$

$$y_a^0 = \frac{\sum_{c(i,j) \neq 0} c(a,j) \cdot y_j^0}{\sum_{c(i,j) \neq 0} c(a,j)} = \frac{c(a, In1) \cdot y_{In1} + c(a, In2) \cdot y_{In2} + c(a, In3) \cdot y_{In3} + c(a, Out) \cdot y_{Out}}{c(a, In1) + c(a, In2) + c(a, In3) + c(a, Out)} = \frac{8 \cdot 2 + 10 \cdot 2 + 2 \cdot 0 + 2 \cdot 0}{8 + 10 + 2 + 2} = \frac{36}{22} \approx 1.6$$

ZFT position of cell a is (1,2)

4.3.2 Analytic Placement – Force-directed Placement

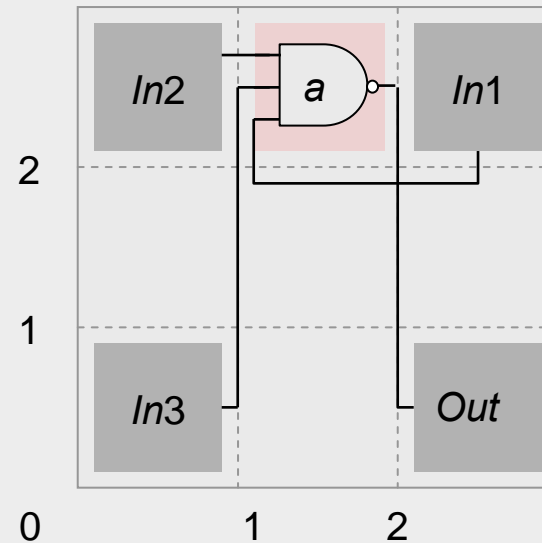
Example: ZFT position

Given:

- Circuit with NAND gate 1 and four I/O pads on a 3 x 3 grid
- Pad positions: $In1$ (2,2), $In2$ (0,2), $In3$ (0,0), Out (2,0)

Solution:

ZFT position of cell a is (1,2)



4.3.2 Analytic Placement – Force-directed Placement

Input: set of all cells V

Output: placement P

```
 $P = \text{PLACE}(V)$  // arbitrary initial placement
 $loc = \text{LOCATIONS}(P)$  // set coordinates for each cell in  $P$ 
foreach (cell  $c \in V$ )
     $status[c] = \text{UNMOVED}$ 
while ( $\text{ALL\_MOVED}(V) \parallel \text{!STOP}()$ ) // continue until all cells have been
    // moved or some stopping
    // criterion is reached
     $c = \text{MAX\_DEGREE}(V, status)$  // unmoved cell that has largest
    // number of connections
     $ZFT\_pos = \text{ZFT\_POSITION}(c)$  // ZFT position of  $c$ 
    if ( $loc[ZFT\_pos] == \emptyset$ ) // if position is unoccupied,
    // move  $c$  to its ZFT position
         $loc[ZFT\_pos] = c$ 
    else
         $\text{RELOCATE}(c, loc)$  // use methods discussed next
     $status[c] = \text{MOVED}$  // mark  $c$  as moved
```

4.3.2 Analytic Placement – Force-directed Placement

Finding a valid location for a cell with an occupied ZFT position
(p : incoming cell, q : cell in p 's ZFT position)

- If possible, move p to a cell position close to q .
- Chain move: cell p is moved to cells q 's location.
 - Cell q , in turn, is shifted to the next position. If a cell r is occupying this space, cell r is shifted to the next position.
 - This continues until all affected cells are placed.
- Compute the cost difference if p and q were to be swapped.
If the total cost reduces, i.e., the weighted connection length $L(P)$ is smaller, then swap p and q .

4.3.2 Analytic Placement – Force-directed Placement (Example)

Given:

Nets

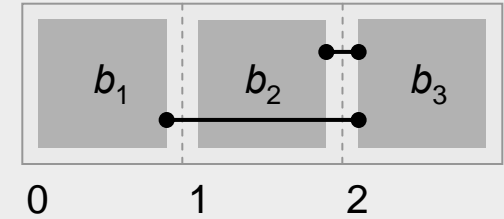
$N_1 = (b_1, b_3)$

$N_2 = (b_2, b_3)$

Weight

$c(N_1) = 2$

$c(N_2) = 1$



4.3.2 Analytic Placement – Force-directed Placement (Example)

Given:

Nets

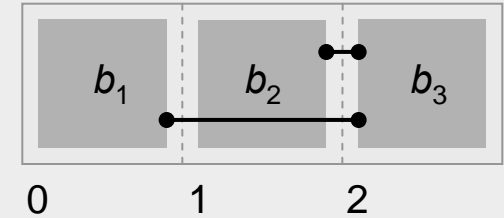
$$N_1 = (b_1, b_3)$$

$$N_2 = (b_2, b_3)$$

Weight

$$c(N_1) = 2$$

$$c(N_2) = 1$$



Incoming cell p

ZFT position of cell p

Cell q

$L(P)$ before move

$L(P)$ / placement after move

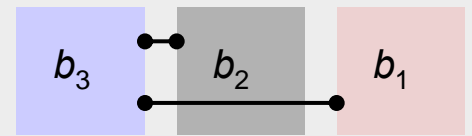
b_3

$$x_{b_3}^0 = \frac{\sum_{c(b_3,j) \neq 0} c(b_3,j) \cdot x_j^0}{\sum_{c(b_3,j) \neq 0} c(b_3,j)} = \frac{2 \cdot 0 + 1 \cdot 1}{2 + 1} \approx 0$$

b_1

$$L(P) = 5$$

$$L(P) = 5$$

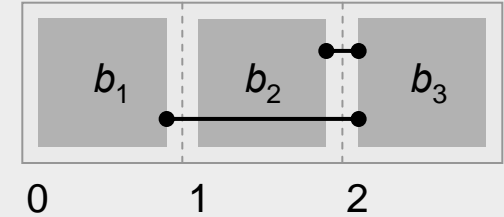


\Rightarrow No swapping of b_3 and b_1

4.3.2 Analytic Placement – Force-directed Placement (Example)

Given:

Nets	Weight
$N_1 = (b_1, b_3)$	$c(N_1) = 2$
$N_2 = (b_2, b_3)$	$c(N_2) = 1$

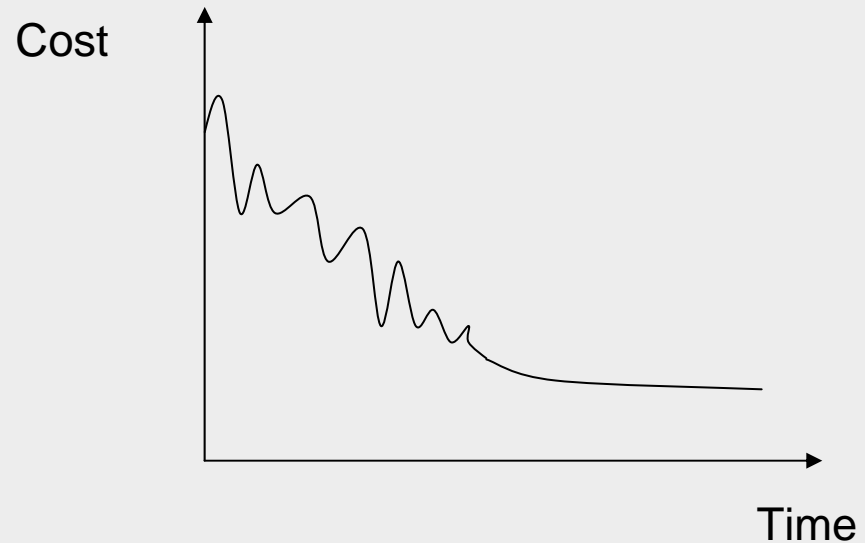
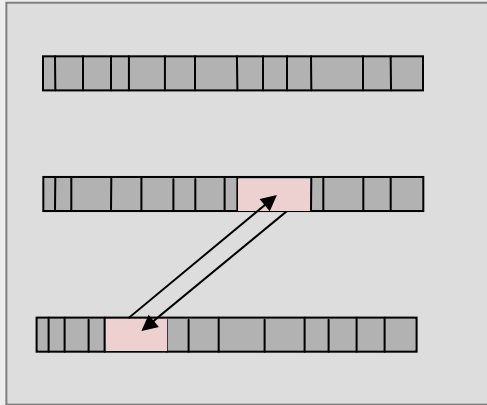


Incoming cell p	ZFT position of cell p	Cell q	$L(P)$ before move	$L(P)$ / placement after move
b_3	$x_{b_3}^0 = \frac{\sum_{c(b_3,j) \neq 0} c(b_3,j) \cdot x_j^0}{\sum_{c(b_3,j) \neq 0} c(b_3,j)} = \frac{2 \cdot 0 + 1 \cdot 1}{2 + 1} \approx 0$	b_1	$L(P) = 5$	$L(P) = 5$ <p>→ No swapping of b_3 and b_1</p>
b_2	$x_{b_2}^0 = \frac{\sum_{c(b_2,j) \neq 0} c(b_2,j) \cdot x_j^0}{\sum_{c(b_2,j) \neq 0} c(b_2,j)} = \frac{1 \cdot 2}{1} = 2$	b_3	$L(P) = 5$	$L(P) = 3$ <p>→ Swapping of b_2 and b_3</p>

4.3.2 Analytic Placement – Force-directed Placement

- Advantages:
 - Conceptually simple, easy to implement
 - Primarily intended for global placement, but can also be adapted to detailed placement
- Disadvantages:
 - Does not scale to large placement instances
 - Is not very effective in spreading cells in densest regions
 - Poor trade-off between solution quality and runtime
- In practice, FDP is extended by specialized techniques for cell spreading
 - This facilitates scalability and makes FDP competitive

4.3.3 Simulated Annealing



- Analogous to the physical **annealing process**
 - Melt metal and then slowly cool it
 - Result: energy-minimal crystal structure
- Modification of an initial configuration (placement) by moving/exchanging of randomly selected cells
 - Accept the new placement if it improves the objective function
 - If no improvement: Move/exchange is accepted with temperature-dependent (i.e., decreasing) probability

4.3.3 Simulated Annealing – Algorithm

Input: set of all cells V

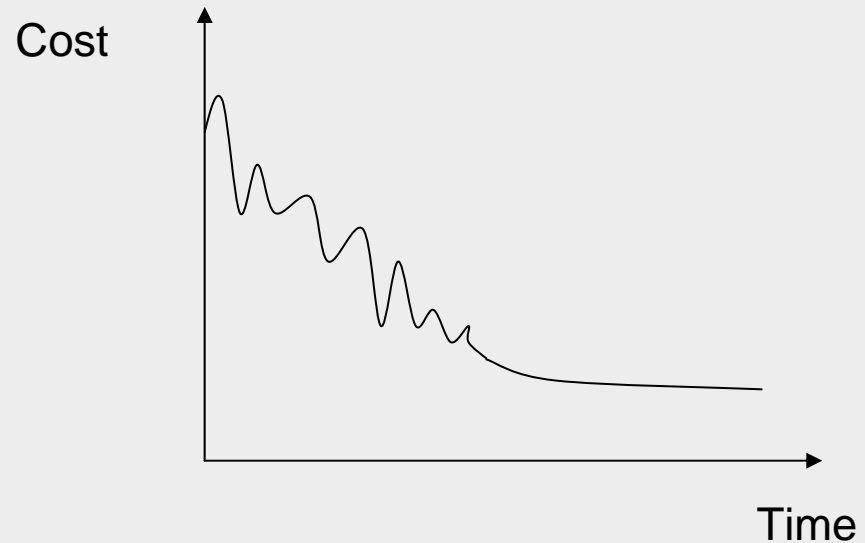
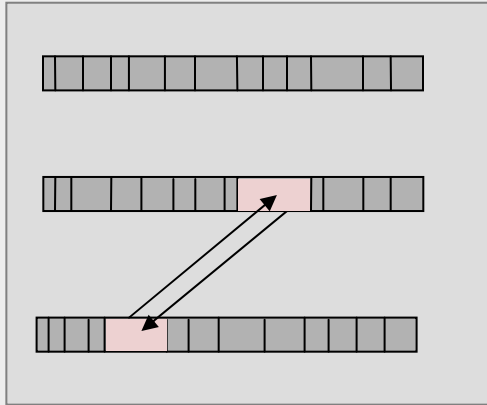
Output: placement P

```
 $T = T_0$  // set initial temperature
 $P = \text{PLACE}(V)$  // arbitrary initial placement
while ( $T > T_{min}$ ) // not yet in equilibrium at  $T$ 
  while (!STOP())
     $new\_P = \text{PERTURB}(P)$ 
     $\Delta \text{cost} = \text{COST}(new\_P) - \text{COST}(P)$ 
    if ( $\Delta \text{cost} < 0$ ) // cost improvement
       $P = new\_P$  // accept new placement
    else // no cost improvement
       $r = \text{RANDOM}(0,1)$  // random number [0,1)
      if ( $r < e^{-\Delta \text{cost}/T}$ ) // probabilistically accept
         $P = new\_P$ 
   $T = \alpha \cdot T$  // reduce  $T$ ,  $0 < \alpha < 1$ 
```

4.3.3 Simulated Annealing

- Advantages:
 - Can find global optimum (given sufficient time)
 - Well-suited for detailed placement
- Disadvantages:
 - Very slow
 - To achieve high-quality implementation, laborious parameter tuning is necessary
 - Randomized, chaotic algorithms - small changes in the input lead to large changes in the output
- Practical applications of SA:
 - Very small placement instances with complicated constraints
 - Detailed placement, where SA can be applied in small windows (not common anymore)
 - FPGA layout, where complicated constraints are becoming a norm

4.3.3 Simulated Annealing



- Analogous to the physical **annealing process**
 - Melt metal and then slowly cool it
 - Result: energy-minimal crystal structure
- Modification of an initial configuration (placement) by moving/exchanging of randomly selected cells
 - Accept the new placement if it improves the objective function
 - If no improvement: Move/exchange is accepted with temperature-dependent (i.e., decreasing) probability

4.3.4 Modern Placement Algorithms

- Predominantly analytic algorithms
- Solve two challenges: interconnect minimization and cell overlap removal (spreading)
- Two families:



Quadratic placers



Non-convex
optimization placers

4.3.4 Modern Placement Algorithms



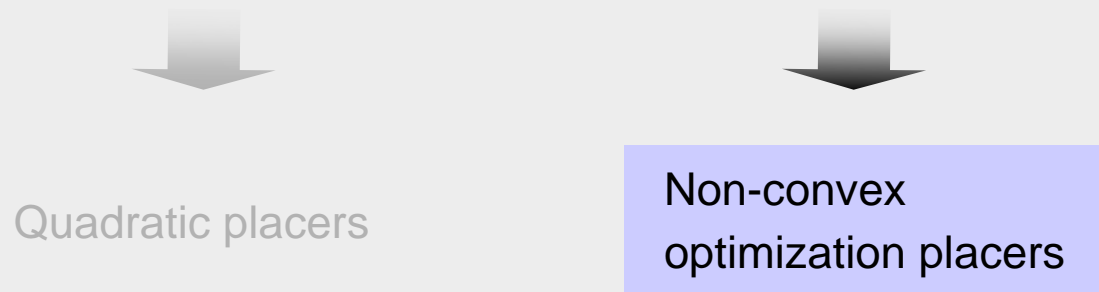
Quadratic placers



Non-convex
optimization placers

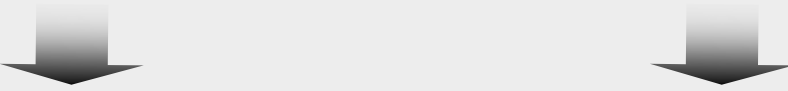
- Solve large, sparse systems of linear equations (formulated using force-directed placement) by the Conjugate Gradient algorithm
- Perform cell spreading by adding fake nets that pull cells away from dense regions toward carefully placed anchors

4.3.4 Modern Placement Algorithms



- Model interconnect by sophisticated differentiable functions, e.g., log-sum-exp is the popular choice
- Model cell overlap and fixed obstacles by additional (non-convex) functional terms
- Optimize interconnect by the non-linear Conjugate Gradient algorithm
- Sophisticated, slow algorithms
- All leading placers in this category use netlist clustering to improve computational scalability (this further complicates the implementation)

4.3.4 Modern Placement Algorithms



Quadratic
Placement

Non-convex
optimization placers

Pros and cons:

- Quadratic placers are simpler and faster, easier to parallelize
- Non-convex optimizers tend to produce better solutions
- As of 2011, quadratic placers are catching up in solution quality while running 5-6 times faster ^[1]

4.4 Legalization and Detailed Placement

4.1 Introduction

4.2 Optimization Objectives


4.3 Global Placement

4.3.1 Min-Cut Placement

4.3.2 Analytic Placement

4.3.3 Simulated Annealing

4.3.4 Modern Placement Algorithms

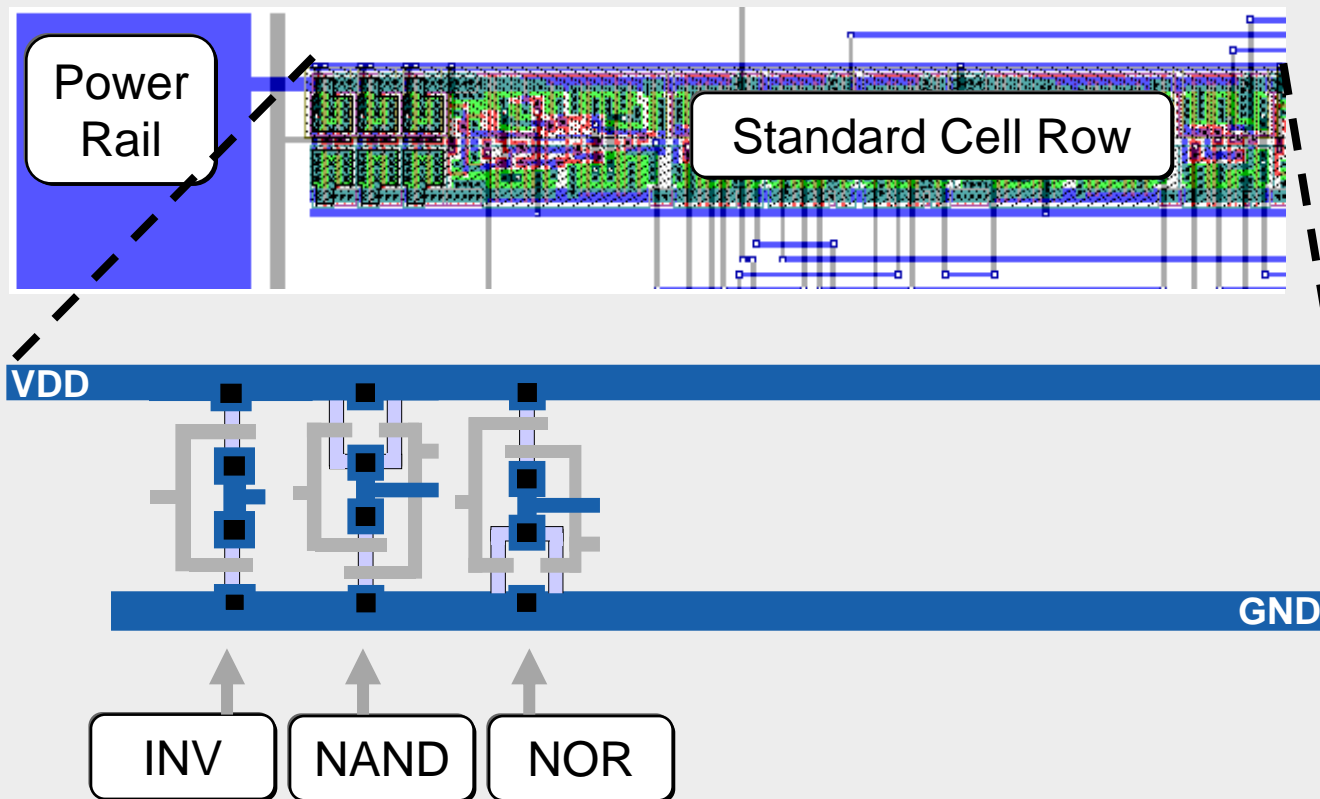
 4.4 Legalization and Detailed Placement

4.4 Legalization and Detailed Placement

- Global placement must be legalized
 - Cell locations typically do not align with power rails
 - Small cell overlaps due to incremental changes, such as cell resizing or buffer insertion
- **Legalization** seeks to find legal, non-overlapping placements for all placeable modules
- Legalization can be improved by **detailed placement** techniques, such as
 - Swapping neighboring cells to reduce wirelength
 - Sliding cells to unused space
- Software implementations of legalization and detailed placement are often bundled

4.4 Legalization and Detailed Placement

Legal positions of standard cells between VDD and GND rails



Summary of Chapter 4 – Problem Formulation and Objectives

- Row-based standard-cell placement
 - Cell heights are typically fixed, to fit in rows (but some cells may have double and quadruple heights)
 - Legal cell sites facilitate the alignment of routing tracks, connection to power and ground rails
- Wirelength as a key metric of interconnect
 - Bounding box half-perimeter (HPWL)
 - Cliques and stars
 - RMSTs and RSMTs
- Objectives: wirelength, routing congestion, circuit delay
 - Algorithm development is usually driven by wirelength
 - The basic framework is implemented, evaluated and made competitive on standard benchmarks
 - Additional objectives are added to an operational framework

Summary of Chapter 4 – Global Placement

- Combinatorial optimization techniques: min-cut and simulated annealing
 - Can perform both global and detailed placement
 - Reasonably good at small to medium scales
 - SA is very slow, but can handle a greater variety of constraints
 - Randomized and chaotic algorithms – small changes at the input can lead to large changes at the output
- Analytic techniques: force-directed placement and non-convex optimization
 - Primarily used for global placement
 - Unrivaled for large netlists in speed and solution quality
 - Capture the placement problem by mathematical optimization
 - Use efficient numerical analysis algorithms
 - Ensure stability: small changes at the input can cause only small changes at the output
 - Example: a modern, competitive analytic global placer takes 20mins for global placement of a netlist with 2.1M cells (single thread, 3.2GHz Intel CPU) ^[1]

[1] M.-C. Kim, D. Lee, I. L. Markov: SimPL: An effective placement algorithm. ICCAD 2010: 649-656

Summary of Chapter 4 – Legalization and Detailed Placement

- Legalization ensures that design rules & constraints are satisfied
 - All cells are in rows
 - Cells align with routing tracks
 - Cells connect to power & ground rails
 - Additional constraints are often considered, e.g., maximum cell density
- Detailed placement reduces interconnect, while preserving legality
 - Swapping neighboring cells, rotating groups of three
 - Optimal branch-and-bound on small groups of cells
 - Sliding cells along their rows
 - Other local changes
- Extensions to optimize routed wirelength, routing congestion and circuit timing
- Relatively straightforward algorithms, but high-quality, fast implementation is important
- Most relevant after analytic global placement, but are also used after min-cut placement
- Rule of thumb: 50% runtime is spent in global placement, 50% in detailed placement [1]

[1] M.-C. Kim, D. Lee, I. L. Markov: SimPL: An effective placement algorithm. ICCAD 2010: 649-656