King Fahd University of Petroleum & Minerals
College of Computer Sciences & Engineering
Department of Computer Engineering

VLSI Placement

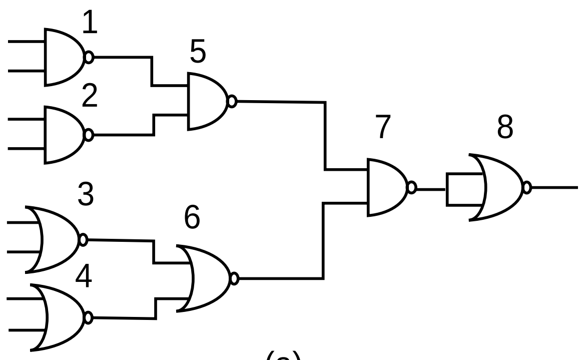
Sadiq M. Sait & Habib Youssef

December 1995

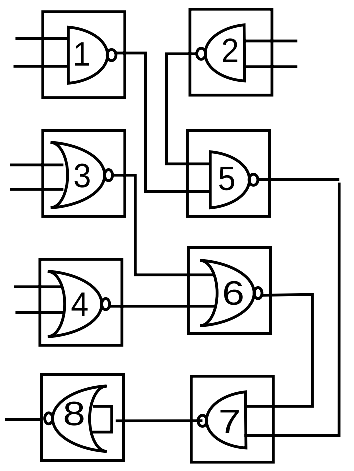
Placement

- *Placement* is the process of arranging the circuit components on a layout surface.

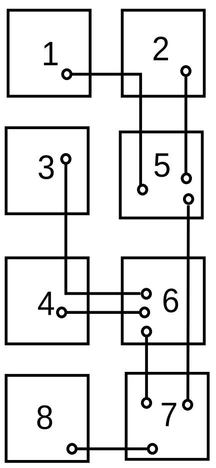
- Example: (a) A tree circuit. (b) A 2-D placement of gates. (c) A 2-D symbolic placement. (d) A 2-D placement requiring 12 units (estimated) of wiring. (e) A 1-D placement requiring 10 units (estimated) of wiring.



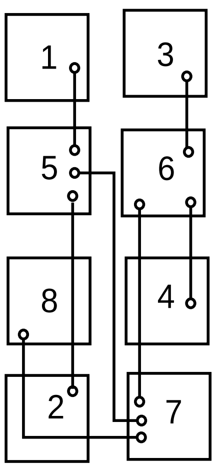
(a)



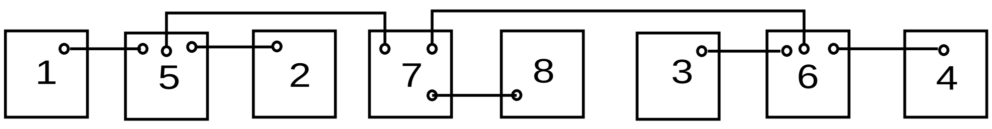
(b)



(c)



(d)

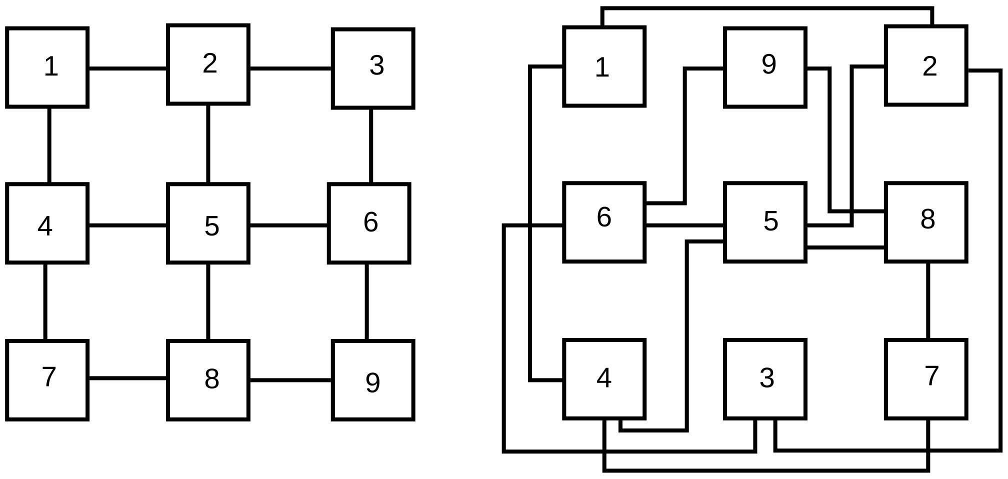


(e)

- The total wirelength ω is a widely used measure of the quality of the placement (easy to compute).

- Consider the symbolic placement of Figure (a) below.

- (a) Optimal placement with ω=12. (b) Alternate solution with ω=22.



(a) (b)

- The area of a layout consists of two parts — the functional area, and the wiring area.

Definition & Complexity

- Placement is NP-complete.
- Even the simplest case (namely 1-D placement), is hard to solve; there are $\frac{n!}{2}$ arrangements of n cells.
- For $n = 50$, (a small design), $\frac{n!}{2} = 1.5 \times 10^{64}$.
- **Problem Statement**

  Given:

  - A collection of cells or modules with ports on the boundaries.
  - The dimensions of these cells.
  - A collection of nets.

  Goal:

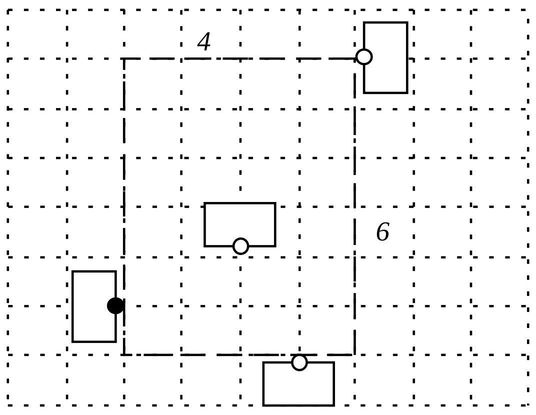
  Find suitable physical locations for each cell on the entire layout.

  Sometimes a subset of the modules are preassigned to locations.

Cost Functions and Constraints

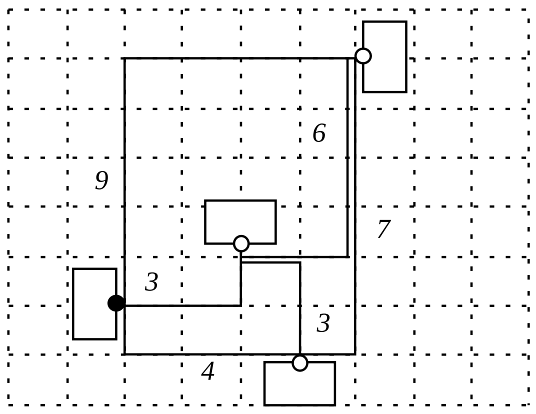
- Routability.
- Wirelength.
- Area.
- Performance (timing, power, etc.)
- Most widely used cost function is wirelength.
- Performing actual routing to compare various placement solutions is impractical; therefore, *estimates* are used.
- Various methods of estimation are:
  - Semi-perimeter Method
  - Complete Graph
  - Minimum Chain
  - Source to Sink Connection
  - Steiner Tree Approximation
  - Minimum Spanning Tree

Application of Different Estimation Methods



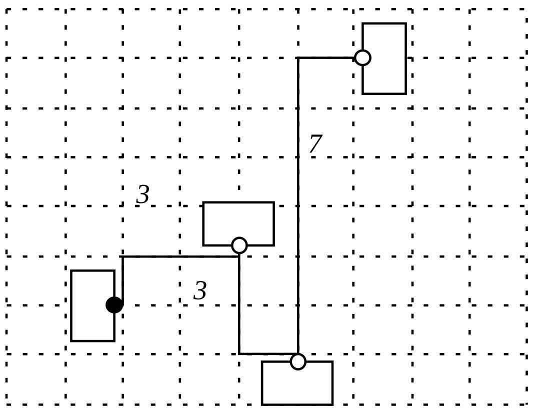
Semi-perimeter length= 10

(a)



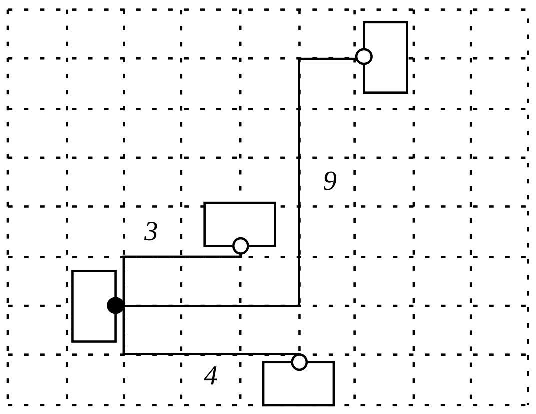
Complete graph length * 2/n= 16

(b)



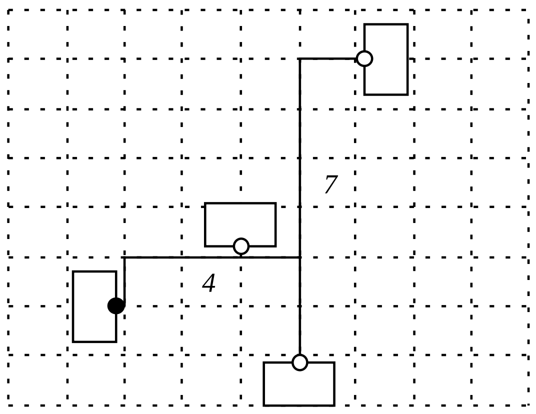
Chain length= 13

(c)



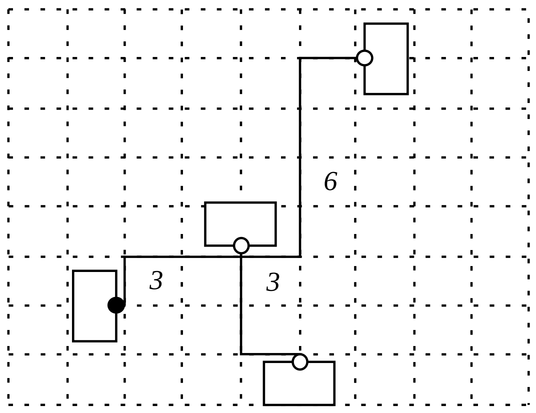
Source to sink length= 16

(d)



Steiner tree length= 11

(e)



Spanning tree length= 12

(f)

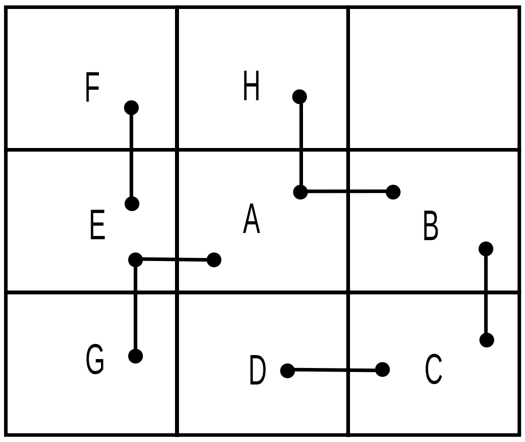
Minimize Total Wirelength

- The total weighted wirelength expressed as:

$$L(P) = \sum_{n \in N} w_n \cdot d_n$$

where,

d_n=estimated length of net n;
w_n=weight of net n.

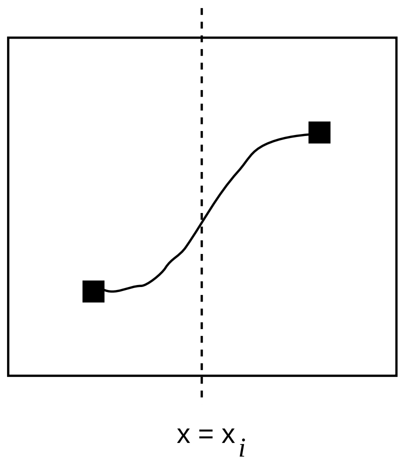


Nets	Weights
$N_1 = (A_1, B_1, H)$	$w_1 = 2$
$N_2 = (B_2, C_1)$	$w_2 = 4$
$N_3 = (C_2, D)$	$w_3 = 3$
$N_4 = (E_1, F)$	$w_4 = 1$
$N_5 = (A_2, E_2, G)$	$w_5 = 3$

$$L(P) = 2 \cdot 2 + 4 \cdot 1 + 3 \cdot 1 + 1 \cdot 1 + 3 \cdot 2 = 18.$$

Minimize Maximum Cut

- Consider the Figure below.



- Let $\Phi_P(x_i)$ and $\Phi_P(y_i)$ denote the number of nets for placement P cut by lines x_i, and y_i.

- For a given placement P, let $X(P)$ indicate the maximum value of $\Phi_P(x_i)$ over all i, that is,

$$X(P) = \max_i [\Phi_P(x_i)]$$
$$Y(P) = \max_j [\Phi_P(y_j)]$$

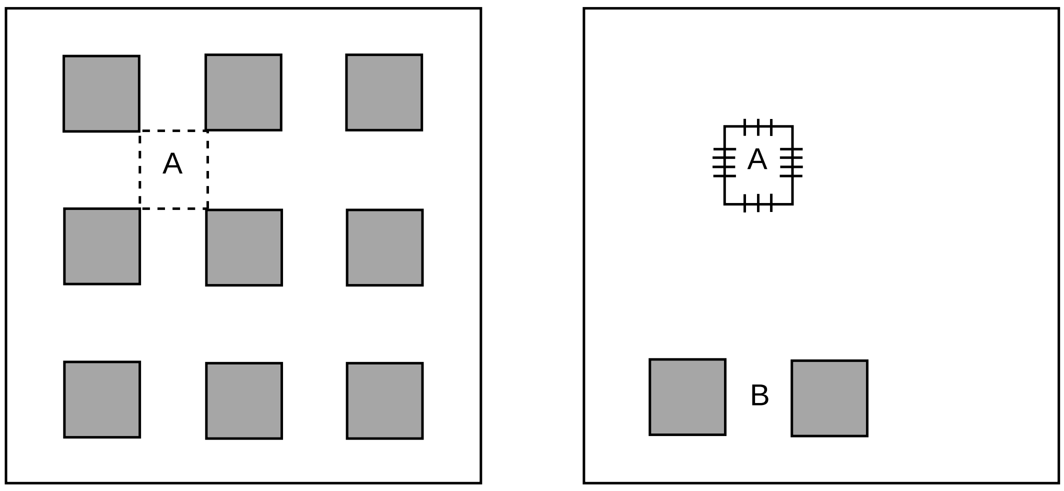
- $\Phi_P(x_i)$ and $\Phi_P(y_j)$ are also related to $L(P)$.

$$L(P) = \sum_i \Phi_P(x_i) + \sum_j \Phi_P(y_j)$$

- Reducing $X(P)$ and $Y(P)$ increases routability.

Minimize Maximum Density

- An alternate measure for routability is the *density* $D(P)$ defined as follows.



(a) (b)

  - Let $\eta_P(e_i)$ indicate the number of nets that must pass through each edge e_i; and
  - Let $\psi_P(e_i)$ indicate the capacity of the edge e_i,

  Then we define the density of edge e_i as

$$d_P(e_i) = \frac{\eta_P(e_i)}{\psi_P(e_i)}$$

- $d_P(e_i)$ must be ≤ 1 for routability. The routability measure of the placement is given by

$$D(P) = \max_i [d_P(e_i)]$$

```
Algorithm  $Con\_Lin\_Plmt(n, C, P)$
Begin .
    (* $n$ is the number of cells.*)
    (* $C[1 \cdots n, 1 \cdots n]$ is the conn matrix.*)
    (* $P[1 \cdots n]$ is the placement vector.*)
    (* $P[i]$ is the slot in which $m_i$ is placed.*)
        For $i = 1$ to $n$
            $P[i] = -\infty$;
            (*$P[i] = -\infty$ means slot $i$ is empty.*)
        EndFor
        $S \leftarrow$ Seed$(n, C)$;
        (*Determine the Seed cell.*)
        $P[S] \leftarrow \frac{n}{2}$;
        (*Place the Seed cell in the center.*)
        Mark $S$ as placed;
        For $i = 1$ to $n - 1$ Do
            $sc \leftarrow Select\_Cell(n, P, C)$;
            $ss \leftarrow Select\_Slot(n, sc, P, C)$;
            $P[sc] \leftarrow ss$;
            Mark $sc$ as placed;
        End;
End.
```

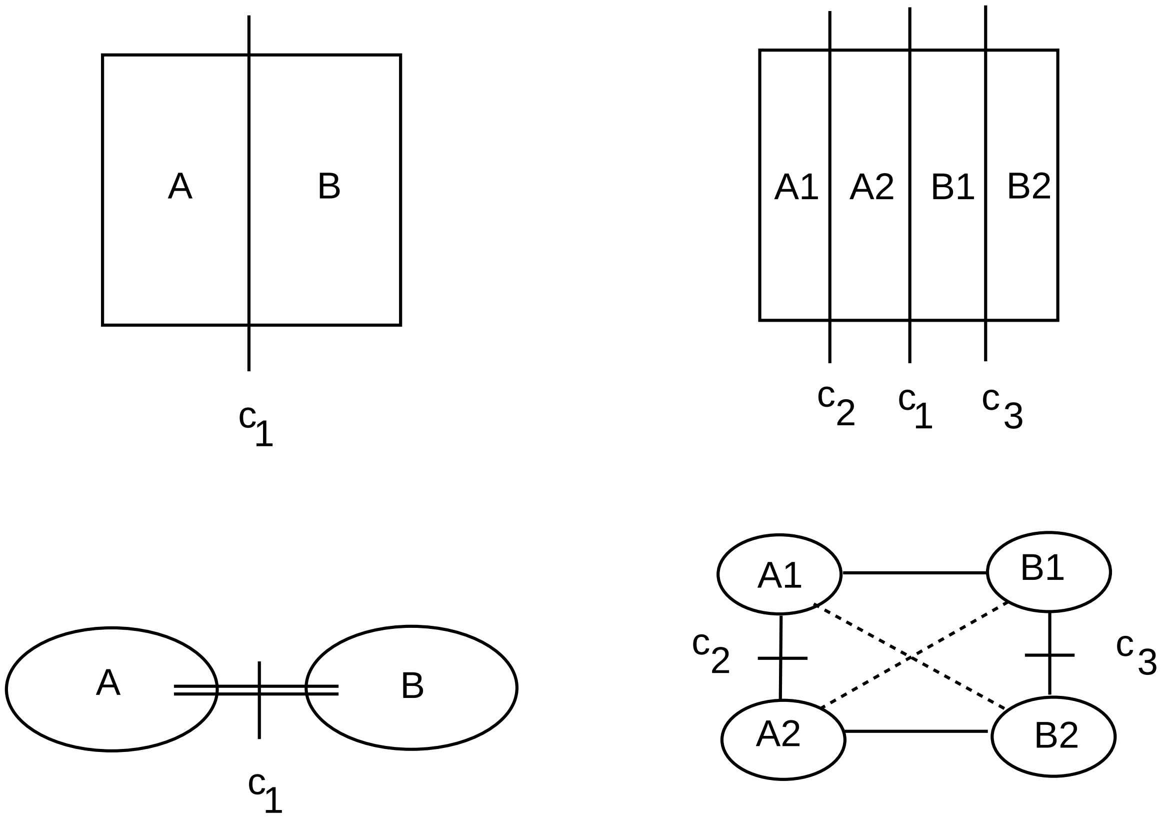
Popular Approaches to Placement

- *Partition-based method* which is based on the min-cut heuristic.
- *Simulated Annealing* based placement.
- *Mathematical Programming* approach; (this has been covered in floorplanning).
- *Force-directed* heuristic which is a numerical technique.
- Other approaches: e.g., Genetic placement.

Partition-Based Methods

- A partitioning algorithm
  - groups together closely connected modules
  - grouping reduces interconnection length and wiring congestion
  - is be applied repeatedly to generate a placement

- **Illustration**



- The procedure described above does not minimize $X(P)$, but minimizes $\Phi_P(c_2)$ subject to the constraint that $\Phi_P(c_1)$ is minimum.

- We write this function as $\Phi_P(c_2)|\Phi_P(c_1)$.

- The procedure also minimizes $\Phi_P(c_3)|\Phi_P(c_1)$.

- A *sequential objective function* denoted by $F(P)$ simplifies the problem.

  $$F(P) = \min[\Phi_P(c_r)] \,|\, \min[\Phi_P(c_{r-1})] \,|\, \ldots \\ \ldots \,|\, \min[\Phi_P(c_1)]$$

  where $c_1, c_2, \ldots, c_r$, is an ordered sequence of vertical or horizontal cutlines.

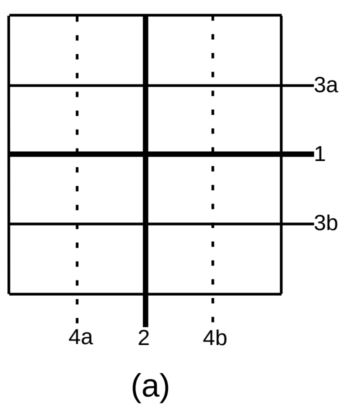
The Min-Cut Placement Algorithm

- Assumes the availability of an ordered sequence of cutlines.
- These cutlines divide the layout into slots. Two key requirements of the algorithm are:

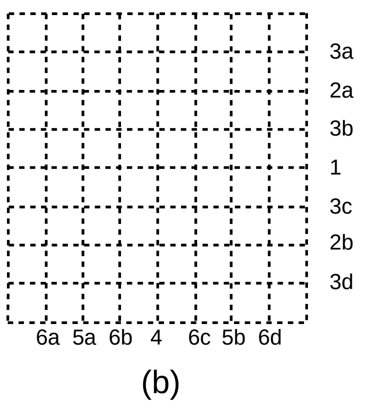
(1) an efficient procedure to partition the circuit, and
(2) the selection of cutlines.

- *Greedy* procedure, therefore solution obtained is not globally optimal.

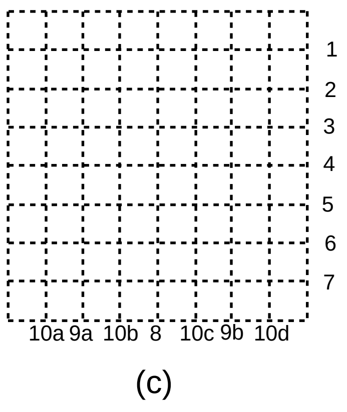
- **Illustration of sequences of cutlines.**



(a)



(b)

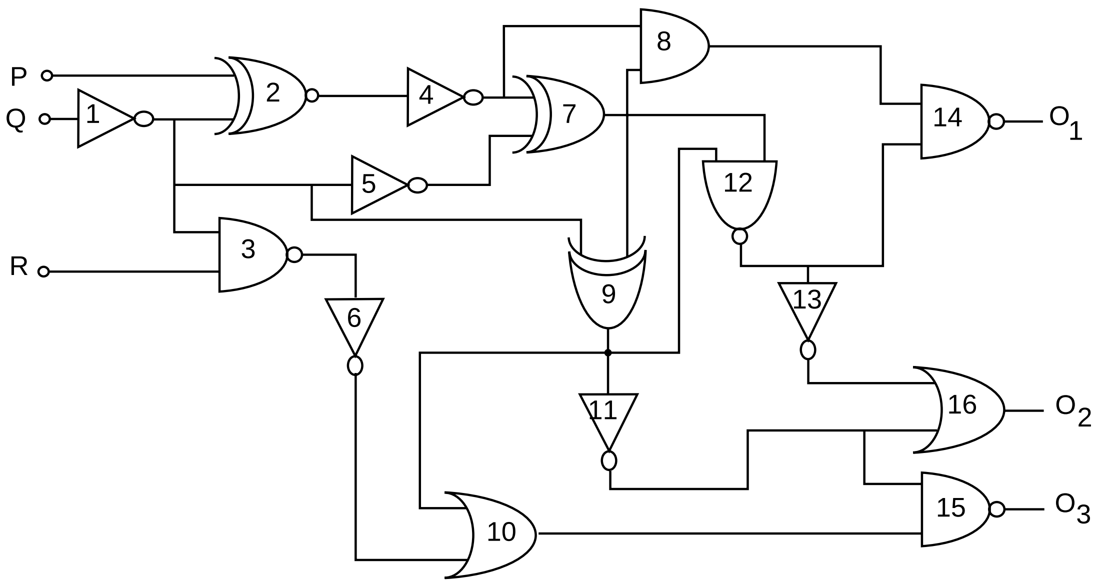


(c)

- Three schemes:
  1. Quadrature Placement Procedure
  2. Bisection Placement Procedure
  3. Slice/Bisection

```
Algorithm  $Min-cut(\aleph, n, C)$
    (* $\aleph$ is the layout surface.
    $n$ is the number of cells to be placed.
    $n_0$ is the number of cells in a slot.
    $C$ is the connectivity matrix *).
        Begin
            If $(n \leq n_0)$ Then place-cells $(\aleph, n, C)$
            Else Begin
                $(\aleph_1, \aleph_2) \leftarrow cut\text{-}surface(\aleph)$;
                $(n_1, c_1), (n_2, c_2) \leftarrow partition(n, C)$;
                Call Min-cut $(\aleph_1, n_1, c_1)$;
                Call Min-cut $(\aleph_2, n_2, c_2)$;
            EndIf;
        End.
```

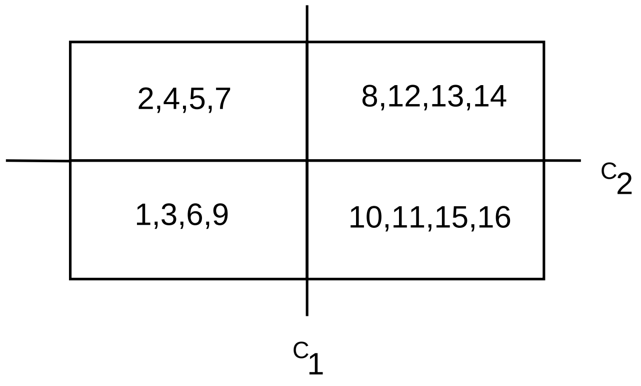
Example



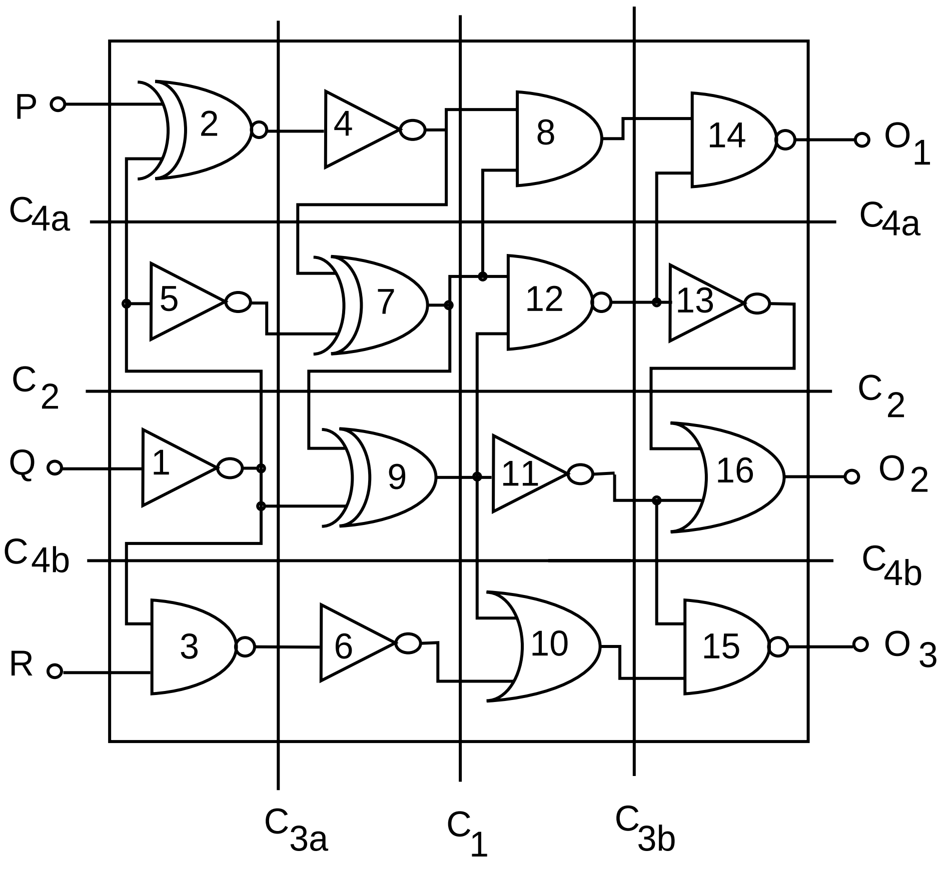
- Partitioning using the KL algorithm yields two sets of gates, namely L and R, where L=\{1,2,3,4,5,6,7,9\} and R=\{8,10,11,12,13,14,15,16\}. The cost of this cut is found to be 4.

- Elements of subsets after second partition are:

LT=\{2,4,5,7\};	(* Top Left *)
LB=\{1,3,6,9\};	(* Bottom Left *)
RT=\{8,12,13,14\};	(* Top Right *)
RB=\{10,11,15,16\}.	(* Bottom Right *)

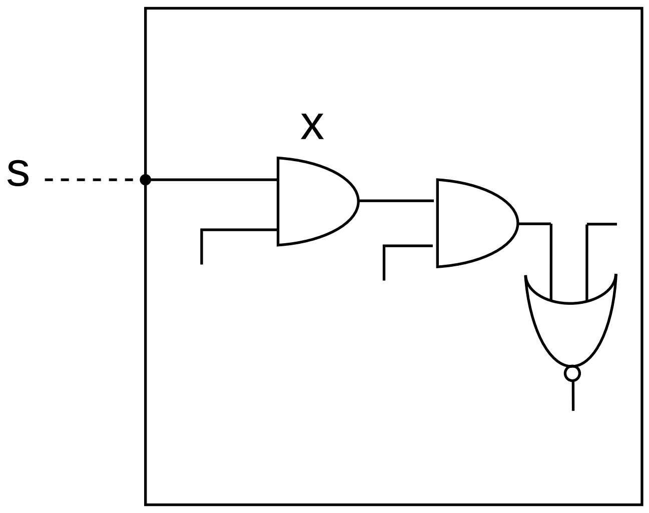


- The procedure is repeated again with two cut-lines running vertically/horizontally (c_{3a} and c_{3b})/(c_{4a} and c_{4b}).
- Final division of layout into slots and the assignment of gates.



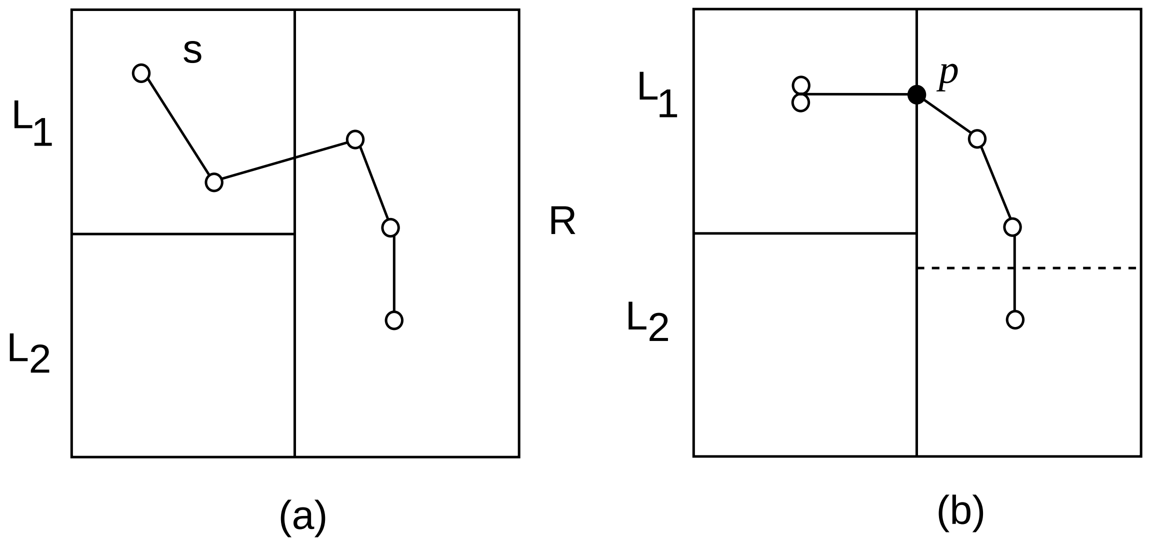
Limitation of the Min-cut Heuristic

- Location of external pins not taken into account.
- The inclusion of these signals in partitioning-based placement is *terminal propagation*.
- Cell x of a group connected to an external signal s.

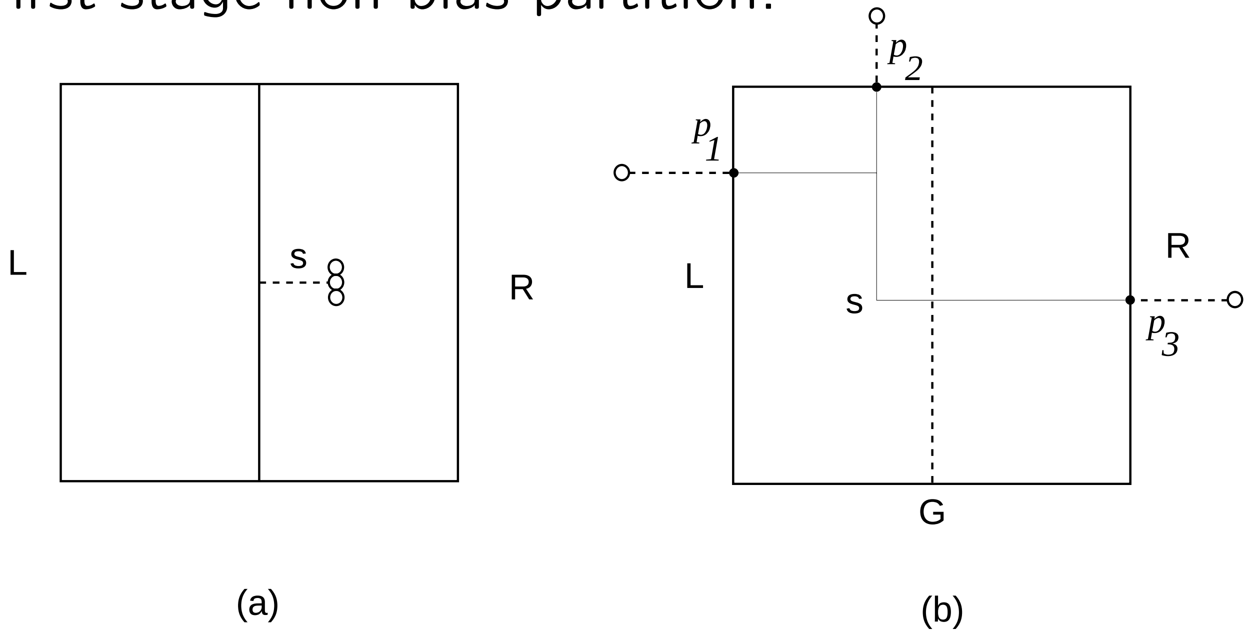


- Clearly cell x has to be nearest to the point where signal s enters.
- At the outermost level, signal positions are typically fixed by pad positions.
- What happens at an inner level of partitioning?

- (a) Partitioning of R following partitioning of L. (b) Propagating s to the axis of partitioning.



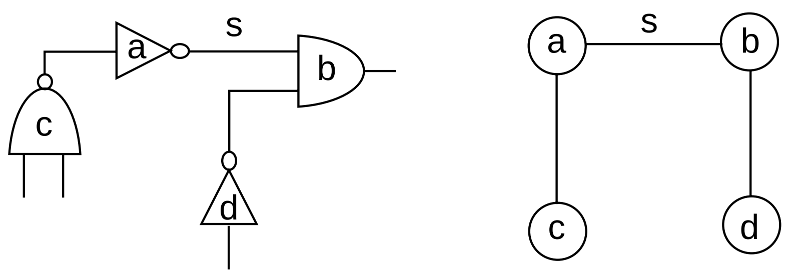
- First stage non-bias partition.



- To do terminal propagation, the partitioning has to be done breadth first.

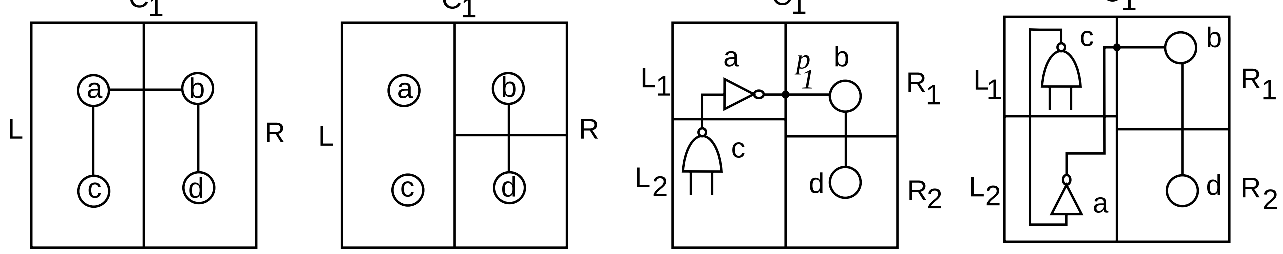
Example

- The gates of the circuit shown in Figure below are to be assigned to slots of a 2×2 array.
- (a) Circuit for Example. (b) Corresponding graph.



(a) (b)

- Solution. (a) Dividing the circuit into L and R. (b) Unbiased partition of R. (c) Biased partition of L producing P. (d) L partitioned *without* terminal propagation.



(a) (b) (c) (d)

Simulated Annealing for Placement

- We now adapt Simulated annealing for placement. The requirement are:

  1. a suitable $perturb$ function to generate a new placement configuration (cell assignment to slots), and
  2. a suitable *accept* function.

- A simple neighbor function is the pairwise interchange.

- Other schemes to generate neighboring states include
  - displacing a randomly selected cell to a random location,
  - the rotation and mirroring of cells, etc.

- In simulated annealing, the swap is accepted if
  - $\Delta h < 0$ ($\Delta h = (Cost(NewS) - Cost(S))$), or
  - if the acceptance function ($random < e^{-\Delta h/T}$) is true.

Example

- Given a netlist with 9 cells and 13 nets. Use SA annealing for placement.
  - Minimize the total Manhattan routing length.
  - Use the $semiperimeter$ method to estimate the wirelength.
  - Use sequential *pairwise exchange* as the *perturb* function.
  - Use the following annealing schedule:

    Initial temperature: T_0=10;
    Constants: M=20; α=0.9; β=1.0.

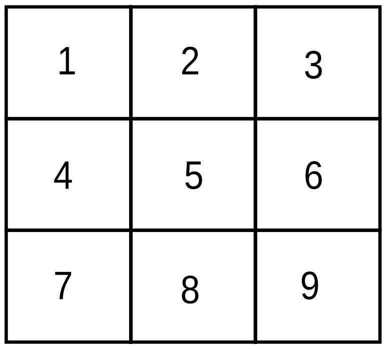
Nets

$N_1 = \{C_4, C_5, C_6\}$	$N_2 = \{C_4, C_3\}$	$N_3 = \{C_2, C_4\}$
$N_4 = \{C_3, C_7, C_8\}$	$N_5 = \{C_2, C_3, C_6\}$	$N_6 = \{C_4, C_7, C_9\}$
$N_7 = \{C_2, C_8\}$	$N_8 = \{C_1, C_7\}$	$N_9 = \{C_3, C_5, C_9\}$
$N_{10} = \{C_6, C_8\}$	$N_{11} = \{C_2, C_6, C_7\}$	$N_{12} = \{C_4, C_7, C_9\}$
$N_{13} = \{C_3, C_9\}$		

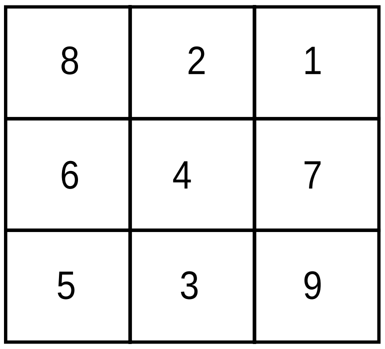
Termination condition: Halt if no cost improvement is observed at two consecutive temperatures.

Solution

- (a) Initial configuration for Example. (b) P obtained by simulated annealing, wirelength using semi-perimeter estimate=24.



(a)



(b)

- The output of the program is given in Table on the next page.

- The entries shown are those where the new configuration was accepted.

Output of Simulated Annealing run

cnt	$\alpha \cdot T$	(a,b)	$random$	$C(S)$	$C(NewS)$	$e^{-\Delta h/T}$
1		(1,2)	0.05538	34	36	0.8187
2		(1,3)	0.37642	36	36	1.0000
3		(1,4)		36	35	
4		(1,5)	0.11982	35	38	0.7408
5		(1,6)		38	36	
6		(1,7)		36	32	
7		(1,8)	0.62853	32	32	1.0000
8		(1,9)		32	31	
10	10	(2,3)	0.75230	31	32	0.9048
11		(2,4)	0.36827	32	32	1.0000
12		(2,5)		32	30	
13		(2,6)	0.86363	30	30	1.0000
14		(2,7)	0.76185	30	31	0.9048
15		(2,8)	0.33013	31	32	0.9048
16		(2,9)	0.65729	32	32	1.0000
17		(3,1)	0.47104	32	33	0.9048
18		(3,2)		33	32	
19		(3,4)	0.42597	32	32	1.0000
20		(3,5)	0.86318	32	33	0.9048
21		(3,6)		33	27	
22		(3,7)		27	26	
24		(3,9)	0.20559	26	28	0.8007
25		(4,1)	0.58481	28	32	0.6411
26		(4,2)	0.30558	32	36	0.6411
27		(4,3)		36	33	
28		(4,5)	0.31229	33	33	1.0000
29		(4,6)	0.00794	33	35	0.8007
30	9	(4,7)		35	34	
31		(4,8)		34	33	
32		(4,9)		33	31	
33		(5,1)		31	30	
34		(5,2)	0.28514	30	32	0.8007
35		(5,3)	0.35865	32	34	0.8007
36		(5,4)	0.87694	34	35	0.8948
37		(5,6)		35	34	
38		(5,7)		34	33	
39		(5,8)	0.03769	33	35	0.8007
40		(5,9)		35	34	

- The same program is executed again by *suppressing* the condition that probabilistically accepts bad moves.

- Output generated by deterministic pairwise interchange algorithm.

iterations	$(swap)$	$Cost(S)$	$Cost(NewS)$
7	(1,8)	34	33
15	(2,8)	33	32
20	(3,5)	32	30
21	(3,6)	30	28
49	(7,1)	28	27
60	(8,4)	27	26

- This transforms the simulated annealing algorithm to the deterministic pairwise exchange algorithm.

- The results of this execution are shown in Table above and the corresponding placement obtained is shown in Figure below.

- The algorithm converges to a *local optimum* after 60 iterations.

TimberWolf Algorithm

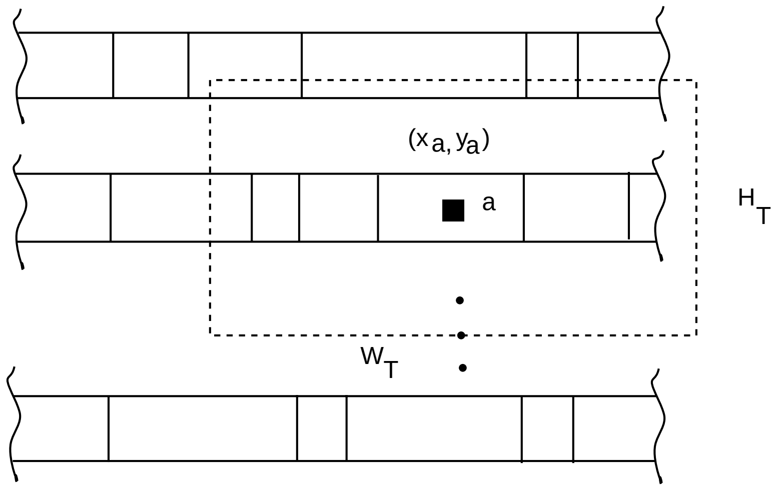
- Placement for standard-cell design with Macro blocks (up to 11 are allowed).
- Pads and macro blocks retain their initial positions is optimized.
- Placement and routing is performed in three distinct stages.
  - Ist stage, cells are placed to minimize the wirelength.
  - 2nd stage, feed-through cells are inserted, wirelength is minimized, and preliminary global routing is done.
  - 3rd stage, local changes are made in the placement to reduce the number of wiring tracks.
- We are concerned only with the Ist stage. Simulated annealing is used.

Perturb Functions

(1) Move a single cell to a new location, say to a different row.
(2) Swap two cells.
(3) Mirror a cell about the x-axis.

- TimberWolf3.2 uses cell mirroring less frequently (10%) when compared to cell displacement and pairwise cell swapping.

- Perturbations are limited to a region within a window of height H_T and width W_T.



- The dimensions of the window are decreasing functions of the temperature T.

- If current temperature is T_1 and next temperature is T_2, the window width and height are decreased as follows:

$$W(T_2) = W(T_1)\frac{\log(T_2)}{\log(T_1)}$$

$$H(T_2) = H(T_1)\frac{\log(T_2)}{\log(T_1)}$$

Cost Function

- The cost function used by the TimberWolf3.2 algorithm is the sum of three components

$$\gamma = \gamma_1 + \gamma_2 + \gamma_3$$

  γ_1 is a measure of the total estimated wire-length. For any net i, if the horizontal and vertical spans are given by X_i and Y_i, then the estimated length of the net i is $(X_i + Y_i)$.

- This must be multiplied by the weight w_i of the net.

- Further sophistication may be achieved by associating two weights with a net — a horizontal component w_i^H and a vertical component w_i^V. Thus,

$$\gamma_1 = \sum_{i \in Nets} [w_i^H \cdot X_i + w_i^V \cdot Y_i]$$

  where the summation is taken over all nets i.

- The weight of a net is useful in indicating how *critical* the net is.

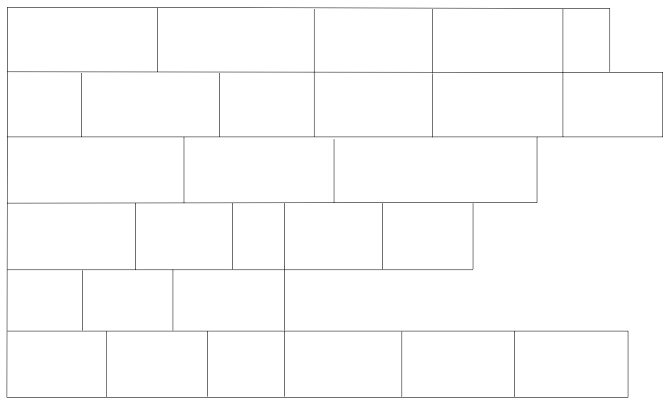
- Let O_{ij} indicate the area of overlap between two cells i and j.

- The second component of the cost function, γ_2, is interpreted as the penalty of overlaps.

$$\gamma_2 = w_2 \sum_{i \neq j} [O_{ij}]^2$$

- In the above equation w_2 is the weight for penalty. The reason for squaring the overlap is to provide much larger penalties for larger overlaps.

- Due to cell displacements and pairwise exchanges of cells, the length of a row may become larger or smaller (see Figure below).



- γ_3 represents a penalty for the length of a row R exceeding (or falling short of) the expected length $\overline{L_R}$.

$$\gamma_3 = w_3 \sum_{rows} \mid L_R - \overline{L_R} \mid$$

where w_3 is the weight of unevenness.

Annealing Schedule

1. The cooling schedule is represented by

   $$T_{i+1} = \alpha(T_i) \times T_i$$

   where $\alpha(T)$ is the cooling rate parameter which is determined experimentally.

2. The annealing process is started at a very high initial temperature say 4×10^6.

3. Initially, the temperature is reduced rapidly [$\alpha(T) \approx 0.8$], in the medium range $\alpha(T) \approx 0.95$, and in the low temperature range, again $\alpha(T) \approx 0.8$.

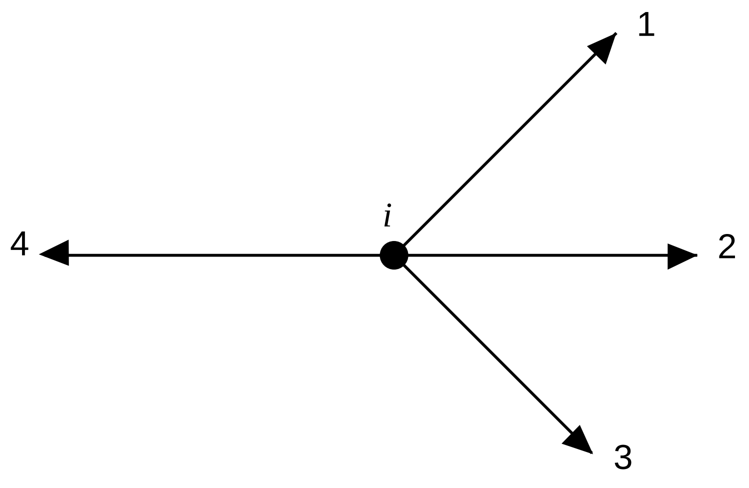
- From experiments, for a 200-cell circuit, 100 moves per cell are recommended, which calls for the evaluation of 2.34×10^6 configurations in about 125 temperature steps.

- For a 3000-cell circuit, 700 moves per cell are recommended, which translates to a total of 247.5×10^6 attempts.

Force-Directed Placement

- The idea behind the method is that cells connected by a net exert forces on one another.
- Magnitude of the force F is proportional to the distance between them.
- Analogous to Hooke's law in mechanics, (force exerted on each other by two masses connected by a spring).
- Force with which the masses pull each other is $k \times d$; (k = spring constant, d = distance between them).
- Total force F_i experienced by cell i connected to several cells j at distances d_{ij} is given by

$$F_i = \sum_j w_{ij} \cdot d_{ij}$$



- Referring to Figure above, the force F_i on cell i connected to 4 other cells is given by

$$F_i = w_{i1} \cdot d_{i1} + w_{i2} \cdot d_{i2} + w_{i3} \cdot d_{i3} + w_{i4} \cdot d_{i4}$$

- If the cell i is free to move, it would do so until the resultant force on it is zero. (*zero-force target location*).

- When all the cells move to their zero-force target locations, the total wirelength ω is minimized.

- The method consists of computing the forces on any given cell, and then moving it to its zero-force target location.

- This location $({x_i}^\circ, {y_i}^\circ)$ can be determined by equating the x- and y- components of the forces on the cell to zero, i.e.,

$$\sum_j w_{ij} \cdot (x_j^\circ - x_i^\circ) = 0; \sum_j w_{ij} \cdot (y_j^\circ - y_i^\circ) = 0$$

- Solving for ${x_i}^\circ$ and ${y_i}^\circ$,

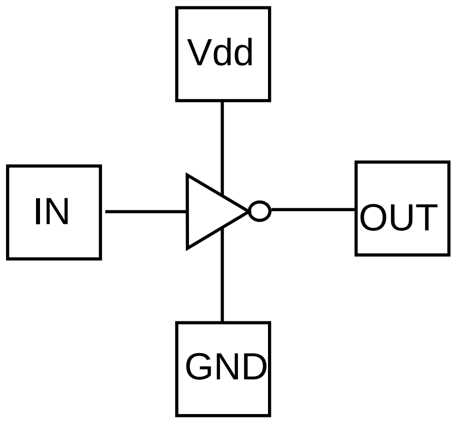
$$\{x_i\}^\circ = \frac{\sum_j w_{ij} \cdot x_j}{\sum_j w_{ij}}$$

$$\{y_i\}^\circ = \frac{\sum_j w_{ij} \cdot y_j}{\sum_j w_{ij}}$$

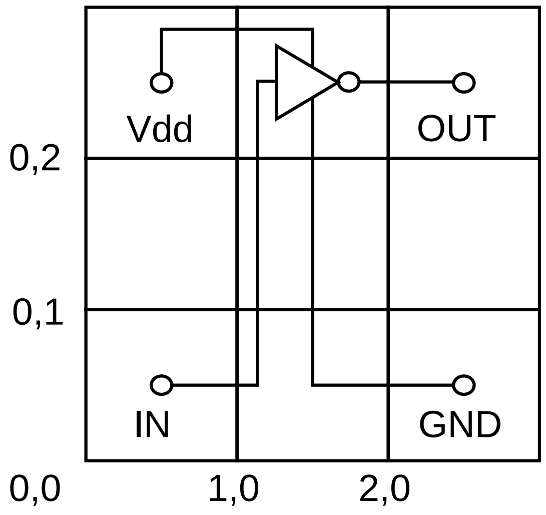
Example

- A circuit with one gate and four I/O pads is given in Figure (a) below. The four pads are to be placed on the four corners of a 3 by 3 grid. If the weights of the wires connected to the gate of the circuit are w_{vdd}=8; w_{out}=10; w_{in}=3; and w_{gnd}=3; find the zero-force target location of the gate inside the grid.

  (a) Circuit for Example. (b) Placement obtained.



(a)



(b)

Solution

The zero-force location for the gate is given by:

$$\{x_i\}^\circ = \frac{\sum_j w_{ij} \cdot x_j}{\sum_j w_{ij}}$$

$$= \frac{w_{vdd} \cdot x_{vdd} + w_{out} \cdot x_{out} + w_{in} \cdot x_{in} + w_{gnd} \cdot x_{gnd}}{w_{vdd} + w_{out} + w_{in} + w_{gnd}}$$

$$= \frac{8 \times 0 + 10 \times 2 + 3 \times 0 + 3 \times 2}{8 + 10 + 3 + 3} = \frac{26}{24} = 1.083$$

$$\{y_i\}^\circ = \frac{\sum_j w_{ij} \cdot y_j}{\sum_j w_{ij}}$$

$$= \frac{w_{vdd} \cdot y_{vdd} + w_{out} \cdot y_{out} + w_{in} \cdot y_{in} + w_{gnd} \cdot y_{gnd}}{w_{vdd} + w_{out} + w_{in} + w_{gnd}}$$

$$= \frac{8 \times 2 + 10 \times 2 + 3 \times 0 + 3 \times 0}{8 + 10 + 3 + 3} = \frac{36}{24} = 1.50$$

The zero-force location for the gate can be approximated to be at grid location (1,2). The final placement of pads and the gate is shown in Figure (b) above.

Force-directed Placement ...

- The approach can be generalized into a constructive placement procedure as follows.
- Starting with some initial placement, a cell at a time is selected, and its zero-force location computed.
- The decisions to be made include:
  - the order in which cells are selected, and
  - where the selected cell is to be put in case the zero-force location is occupied.
- The cell to be moved may be selected randomly or by using a heuristic technique.
- It seems logical to select the cell i for which F_i is maximum in the present configuration.
- If the zero-force location is occupied by another cell q, then options available include:

(1) Move p to a free location close to q.
(2) Evaluate the change in cost if p is swapped with q.
(3) Ripple move.
(4) Chain move.
(5) Find all pairs (p, q) where zero force location of p is q and vice versa. Swap the cells p and q.

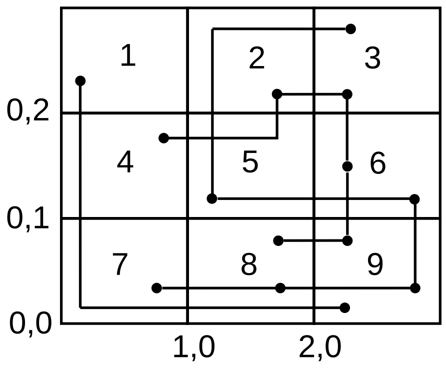
```
Algorithm $\{ForcedirectedPlacement\}$
    Compute total connectivity of each cell;
    Order the cells in decreasing order of their connectivities
    and store them in a list L.
    While $(iteration\_count < iteration\_limit)$
        Seed = next module from L
        Declare the position of the cell $vacant$;
        While $end\_ripple$ = false
            Compute target point of the cell to nearest integer;
            Case target point is
                VACANT:
                    Move seed to target point and lock;
                    $end\_ripple \leftarrow true$;
                    $abort\_count \leftarrow 0$;
                LOCKED:
                    Move selected cell to nearest vacant location;
                    $end\_ripple \leftarrow true$;
                    $abort\_count \leftarrow abort\_count + 1$;
                    If $abort\_count > abort\_limit$ Then
                        Unlock all cell locations;
                        $iteration\_count \leftarrow iteration\_count + 1$;
                    EndIf;
                SAME AS PRESENT LOCATION:
                    $end\_ripple \leftarrow true$;
                    $abort\_count \leftarrow 0$;
                OCCUPIED: (*and not locked*)
                    Select cell at target point for next move;
                    Move seed cell to target point and lock
                    the target point;
                    $end\_ripple \leftarrow false$;
                    $abort\_count \leftarrow 0$;
            EndCase;
        EndWhile;
    EndWhile;
End.
```

Example

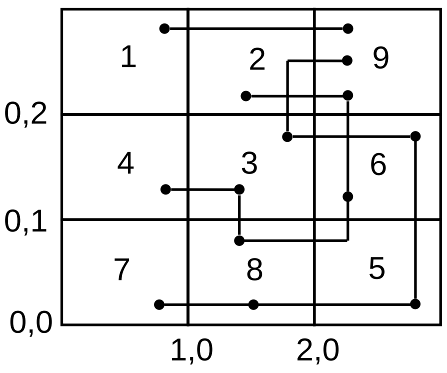
- Consider a gate-array of size 3 rows and 3 columns. A circuit with 9 cells and 3 signal nets is to be placed on the gate-array using the force-directed algorithm. The initial placement is shown in Figure (a) below. The modules are numbered $C_1, \cdots, C_9$ and the nets N_1, N_2, N_3 are shown below. Show the final placement and calculate the improvement in total wirelength achieved by the algorithm.

Nets
$N_1 = (C_3, C_5, C_6, C_7, C_8, C_9)$
$N_2 = (C_2, C_3, C_4, C_6, C_8, C_9)$
$N_3 = (C_1, C_9)$

Placement of Example. (a) Initial placement, wirelength estimate using chain connection=16. (b) Final placement, wirelength estimate using chain connection=14.



(a)



(b)

Solution

The connectivity matrix for the given netlist and the total connectivity of cells is shown in Table below.

Cells	1	2	3	4	5	6	7	8	9	$\sum$
1	0	0	0	0	0	0	0	0	1	1
2	0	0	1	1	0	1	0	1	1	5
3	0	1	0	1	1	2	1	2	2	10
4	0	1	1	0	0	1	0	1	1	5
5	0	0	1	0	0	1	1	1	1	5
6	0	1	2	1	1	0	1	2	2	10
7	0	0	1	0	1	1	0	1	1	5
8	0	1	2	1	1	2	1	0	2	10
9	1	1	2	1	1	2	1	2	0	11

We will use the algorithm of Figure discussed earlier to solve the problem. We select $abort\_count = 3$ and $iteration\_count = 2$.

First 2 iterations of force-directed placement of Example.

$iter$	Selected Cell	Target	Case	Placed	Result
1	9 (Seed)	(1,1)	Occupied	(1,1)	1 2 3 4 9 6 7 8 -
	5	(1,1)	Locked $abort\_count = 1$	(2,0)	1 2 3 4 9 6 7 8 5
	3 (Seed)	(1,1)	Locked $abort\_count = 2$	(2,2)	1 2 3 4 9 6 7 8 5
	6 (Seed)	(1,1)	Locked $abort\_count = 3$	(2,1)	1 2 3 4 9 6 7 8 5
2	9 (Seed)	(1,1)	Same	(1,1)	1 2 3 4 9 6 7 8 5
	3 (Seed)	(1,1)	Occupied	(1,1)	1 2 - 4 3 6 7 8 5
	9	(1,1)	Locked $abort\_count = 1$	(2,2)	1 2 9 4 3 6 7 8 5
	6 (Seed)	(1,1)	Locked $abort\_count = 2$	(2,1)	1 2 9 4 3 6 7 8 5
	8 (Seed)	(1,1)	Locked $abort\_count = 3$	(1,0)	1 2 9 4 3 6 7 8 5

Other Approaches and Recent Work

- Artificial Neural Networks.
- Genetic Algorithm.
- Stochastic/Simulated Evolution.
- Performance Driven Placement (timing, power, etc.,).
- Placement for new design methodologies and/or technologies.

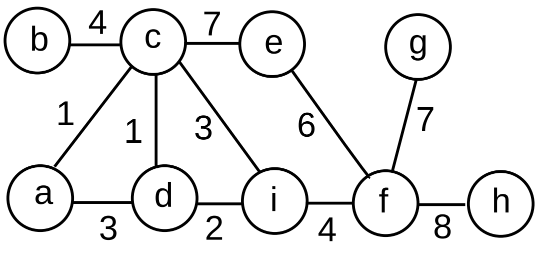
Genetic Placement

- It is a search technique which emulates the natural process of evolution as a means of progressing toward the optimum.
- It has been applied in solving various optimization problems including VLSI cell placement.
- **Terminology**
  - *Population*
  - *Genes*
  - *Chromosome*
  - *Schema*
  - *Generation*
  - *Fitness*
  - *Parents* and *Offsprings*
  - *Genetic Operators*, **Crossover**, **Mutation**, **Inversion**.

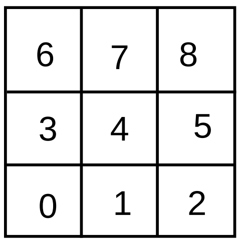
Example

- Consider the graph of Figure (a) below. The 9 vertices represent modules and the numbers on the edges represent their weighted interconnection. Give a possible solution and express it as a string of symbols. Generate a population of 4 chromosomes and compute their fitness using the reciprocal of weighted Manhattan distance as a measure of fitness.

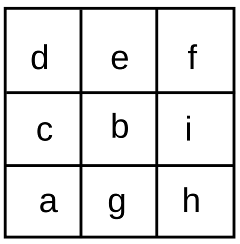
(a) Graph of a circuit to be placed. (b) Position definition. (c) One possible placement.



(a)



(b)



(c)

Solution

- The nine modules can be placed in the nine slots as shown in Figure (b).
- One possible solution is shown in Figure (c).
- Let us use a string to represent the solution as follows.

- Let the left most index of the string of the solution correspond to position ‘0’ of Figure (b) and the right most position to location 8.

- Then the solution of Figure (c) can be then represented by the string $\boxed{\text{aghcbidef}}$ $(\frac{1}{85})$.

- The number in parenthesis represents the fitness value which is the reciprocal of the weighted wirelength based on the Manhattan measure.

- If the lower left corner of the grid in Figure (b) is treated as the origin, then it is easy to compute the Cartesian locations of any module.

- For example the index of module i is 5.

- Its Cartesian coordinates are given by $x = (5 \bmod 3) = 2$, and $y = \lceil \frac{5}{3} \rceil = 1$.

- Any string (of length 9) containing characters $[a,\ b,\ c,\ d,\ e,\ f,\ g,\ h,\ i]$ represents a possible solution.

- There are 9! solutions equal to the number of permutations of length 9.

Genetic Operators

- *Crossover* is the main genetic operator.
- It operates on two parents and generates an offspring.
- It is an inheritance mechanism.
- The operation consists of choosing a random cut point and generating the offspring by combining the segment of one parent to the left of the cut point with the segment of the other parent to the right of the cut.
- From our previous example consider the two parents $\boxed{\text{bidef}|\text{aghc}}$ $(\frac{1}{86})$, and $\boxed{\text{bdefi}|\text{gcha}}$ $(\frac{1}{110})$.
- If the cut point is randomly chosen after position 4, then the offspring produced is $\boxed{\text{bidefgcha}}$.
- Simple crossover sometimes fails.
- Modifications to the above crossover operations to avoid repetition of symbols are
  a. Order crossover,
  b. Partially Mapped Crossover (PMX), and
  c. Cycle crossover.

Partially Mapped Crossover

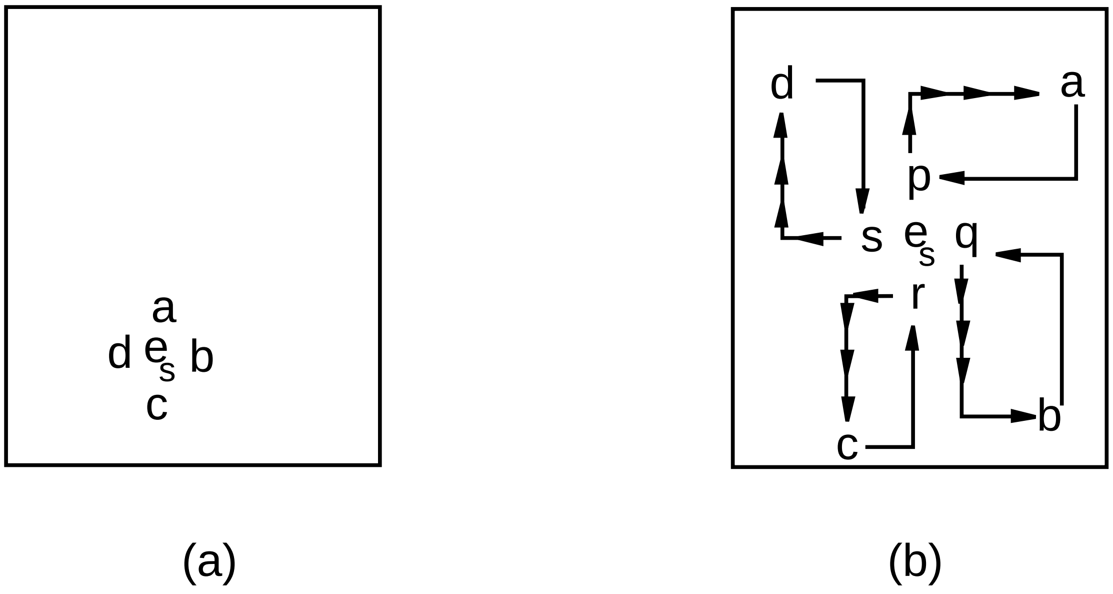
- Here we will explain the operation of the PMX technique.
- The PMX crossover is implemented as follows:
  - Select two parents (say 1 and 2) and choose a random cut point.
  - As before the entire right substring of parent 2 is copied to the offspring.
  - Next, the left substring of parent 1 is scanned from the left, gene by gene, to the point of the cut.
  - If a gene does not exist in the offspring then it is copied to the offspring.
  - However if it already exists in the offspring, then its position in parent 2 is determined and the gene from parent 1 in the determined position is copied.
- As an example consider the 2 parents $\boxed{\text{bidef}|\text{gcha}}$ ($\frac{1}{86}$), and $\boxed{\text{aghcb}|\text{idef}}$ ($\frac{1}{85}$). Let the crossover position be after 4.
- Then the offspring due to PMX is $\boxed{\text{bgcha}|\text{idef}}$.

Crossovers used in *Genie*

Genie: a genetic placement system for placing modules on a rectangular grid.

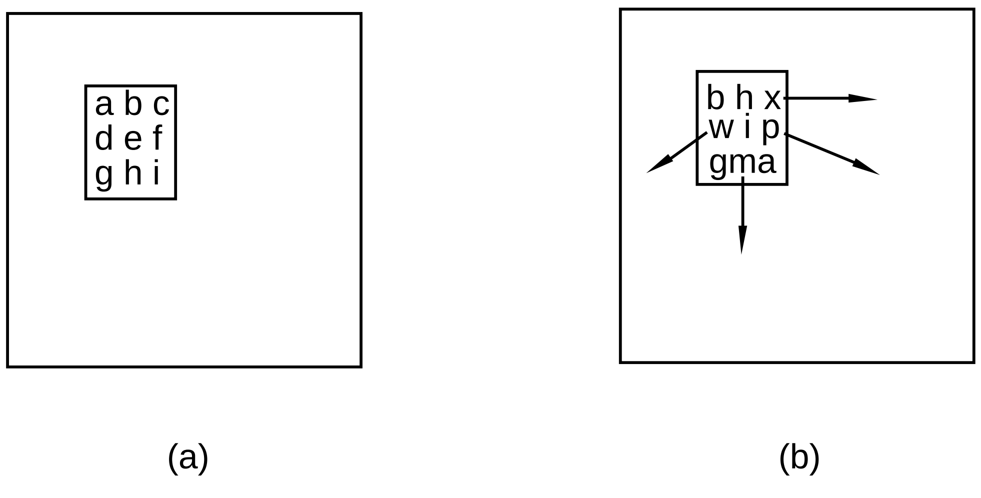
- The first crossover operator selects a random module e_s and brings its four neighbors in parent 1 to the location of the corresponding neighboring slots in parent 2.
- **Illustration**

  (a) A random module and its neighbors. (b) The neighbors in (a) of parent 1 replace neighboring modules in parent 2.



- The second crossover operator selects a square consisting of $k \times k$ modules from parent 1 and copies it to parent 2.

- **Illustration** (a) A square is selected in parent 1. (b) Modules of square in parent 1 are copied to parent 2 and duplicate modules are moved out.



```
Algorithm ($Genetic\_Algorithm$)
  (* $N_p$= Population Size *)
  (* $N_g$= Number of Generations *)
  (* $N_o$= Number of Offsprings *)
  (* $P_i$= Inversion Probability *)
  (* $P_\mu$= Mutation Probability *)
  Begin
    (* Randomly generate the Initial Population *)
    Construct_Population($N_p$);
    For $j = 1$ to $N_p$
      Evaluate $Fitness$(Population[$N_p$])
    EndFor;
    For $i = 1$ to $N_g$
      For $j = 1$ to $N_o$
        (* Choose parents with probability *)
        (* proportional to fitness *)
        $(x, y) \leftarrow Choose\_parents$;
        (* Perform crossover to generate offsprings *)
        offspring[$j$] $\leftarrow Generate\_offspring(x, y)$;
        For $k = 1$ to $N_p$
          With probability $P_\mu$ Apply
          $Mutation$(Population[$k$])
        EndFor;
        For $k = 1$ to $N_p$
          With probability $P_i$ Apply
          $Inversion$(Population[$k$])
        EndFor;
        Evaluate $Fitness$(offspring[$j$])
      EndFor;
      Population $\leftarrow Select$(Population, offspring, $N_p$)
    EndFor;
    Return highest scoring configuration in Population
  End.
```

Conclusion

- We discussed a major VLSI design automation subproblem, namely placement.
- Wirelength is one of the most commonly optimized objective function.
- Different techniques used to estimate the wirelength of a given placement were presented.
- Other cost functions, (minimization of maximum cut, and of maximum density) were studied.
- Three algorithms were discussed:
  - Min-cut partitioning based placement.
  - Simulated annealing algorithm.
  - Force-directed placement algorithm.
- Terminal propagation which also considers external pins during placement was studied.
- SA is currently the most popular technique in terms of placement quality, (takes an excessive amount of time).
- We also discussed the TimberWolf3.2 package which uses SA for module placement.
- Force-directed algorithms operate on the physical analogy of masses connected by springs.
- We also discussed some recent attempts.

```
Procedure ($Genetic\_Algorithm$)
    M= Population size.                      (*# Of possible solutions a
    $N_g$= Number of generations.            (*# Of iterations.*)
    $N_o$= Number of offsprings.             (*To be generated by cross
    $P_\mu$= Mutation probability.           (*Also called mutation rate
    $\mathcal{P} \leftarrow \Xi(\mathrm{M})$ (*Construct initial populati
    For $j = 1$ to M                         (*Evaluate fitnesses of all i
            Evaluate $f(\mathcal{P}[j])$     (*Evaluate fitness of $\mathcal{P}$.*)
    EndFor
    For $i = 1$ to $N_g$
            For $j = 1$ to $N_o$
                $(x, y) \leftarrow \phi(\mathcal{P})$        (*Select two parents $x$ and
                offspring[$j$] $\leftarrow \chi(x, y)$       (*Generate offsprings by cr
                Evaluate $f$(offspring[$j$])                 (*Evaluate fitness of each o
            EndFor

            For $j = 1$ to $N_o$                             (*With probability $P_\mu$ apply
                mutated[$j$] $\leftarrow \mu(y)$
                Evaluate $f$(mutated[j])
            EndFor
    $\mathcal{P} \leftarrow Select(\mathcal{P}$, offsprings)  (*Select best M solutions f
    EndFor
    Return highest scoring configuration in $\mathcal{P}$.
End
```