CAD Algorithms

Floorplanning

Mohammad Tehranipoor

ECE Department

Floorplanning

- After the circuit partitioning phase:
  - The area occupied by each block can be estimated.
  - Possible shapes of the blocks can be ascertained.
  - Number of terminals required by each block is known.
  - The netlist specifying the connections between the blocks is available.
- In order to complete the layout, a specific shape needs to be assigned to a block and arrange the blocks on the layout surface.

Physical Design Automation

- **Steps left:**
  - **Floorplanning phase**: Planning and sizing of blocks and interconnects
  - **Placement phase**: Assigning a specific location to blocks.
  - **Routing phase**: Completing interconnections

Floorplanning Phase

- The blocks with known dimensions are called ***fixed*** blocks.
  - Standard cells
- The blocks for which dimensions are yet to be determined are called ***flexible*** blocks.
  - Thus we need to determine an appropriate shape for each block, location of each block on the layout surface, and location of pins on the boundary of the blocks.
- **Placement Problem**: Problem of assigning location of fixed blocks on a layout surface.
- **Floorplanning Problem**: Problem of assigning location of flexible blocks on a layout surface.
- The placement problem is a restricted version of the floorplanning problem.

Floorplanning vs. Placement

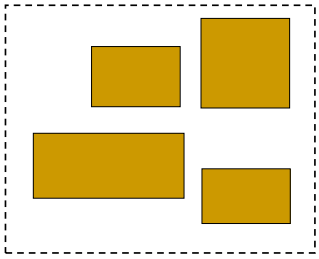
- Both determine block positions to optimize the circuit performance.
- **Floorplanning:**
  - Details like shapes of blocks, pin assignments, etc. are not yet fixed (blocks with flexible shape are also called **soft blocks**).
- **Placement:**
  - Details like module shapes and I/O pin positions are fixed (blocks with no flexibility in shape are also called **hard blocks**).

Floorplanning Phase

- **Input**:
  - Netlist
  - Number of terminals for each block
  - Set of blocks
  - Area of each block
  - Possible shapes of each block
- **Output**:
  - Shapes and location of blocks

Hierarchical Design

- Q: How to put the blocks together without knowing their shapes and the positions of the I/O pins?
- If we design the blocks first, those blocks may not be able to form a tight packing.
- During floorplanning, we know that area is fixed but dimensions are not known.



Floorplanning Problem

- The floorplanning problem is to plan the *positions* and *shapes* of the modules at the beginning of the design cycle to **optimize the performance of the circuits**:
  - chip area
  - total wirelength
  - delay of critical paths
  - routability
  - others, e.g., noise, heat dissipation, etc.
- In floorplanning several alternatives for each block are considered.

Floorplanning Problem Formulation

- **Given**,
  - A set S of n rectangular modules
    - $S=\{1,2,\ldots,i,\ldots,n\}$
  - An interconnection matrix $C_{nxn}=[c_{i,j}]$, $1 \le i,j \le n$
    - Where $c_{i,j}$ indicates connectivity between module i and j.
  - A list of n triples $(A_1,r_1,s_1),\ldots,(A_i,r_i,s_i),\ldots,(A_n,r_n,s_n)$, where A_i is the area of block i, and r_i and s_i are the lower and upper bound constraints on the shape of i.
  - Two additional integers p and q which are lower and upper bound constraints on the shape of the rectangle enveloping the n blocks.

Floorplanning Problem Formulation

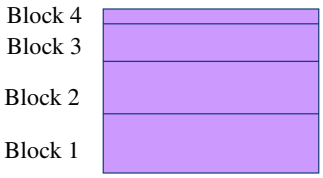
- Output
  - Area constraint $A_i=w_i \times h_i \quad , \quad 1 \le i \le n$
  - $r_i \le h_i/w_i \le s_i$
  - $p \le H/W \le q$

where h/w is called ***aspect ratio*** of a block.

Then, r_i and s_i are called aspect ratio of block i.

Bounds on Aspect Ratios

- If there is no bound on the aspect ratios, we may be able to pack very tightly:

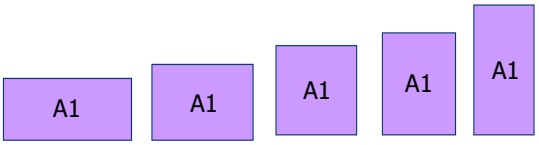


- But laying out the blocks as long strips may not necessarily result in best routability and rectangle final layout, so for each block i :

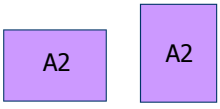
$$r_i \le h_i / w_i \le s_i \quad \text{and} \quad p \le H/W \le q$$

Bounds on Aspect Ratios

- We can allow several shapes for each soft block:



- For hard blocks, only the orientations can be changed:



Objective Funtion

- **Wirelength and Area**
- A commonly used objective function is a weighted sum of area and wirelength:

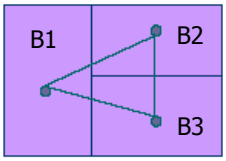
$$Cost = \alpha A + \beta L$$

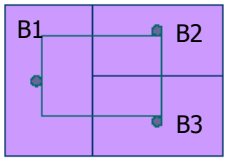
- where A is the total area of the packing, L is the total wirelength, and α and β are constants.
  - α and β define different weights for A and L.
  - Minimize Area: $\alpha=1$ and $\beta=0$
  - Minimize Wirelength: $\alpha=0$ and $\beta=1$
  - Tradeoff: $\alpha=0.5$ and $\beta=0.5$

- Total Area of packing: $A = H \times W$

Wirelength Estimation

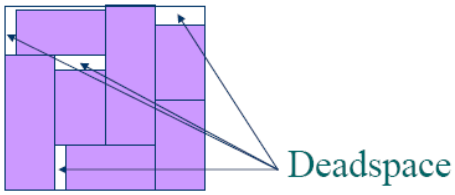
- Exact wirelength of each net is not known until routing is done.
- In floorplanning, even pin positions are not known.
  - The process of identifying pin location is called *pin assignment*.
- A possible wirelength estimation:
  - Center-to-center estimation





Deadspace

- Deadspace is the space that is wasted:



  - Minimizing area is the same as minimizing deadspace.
  - Deadspace percentage is computed as

$$((A - \sum_i Ai) / \sum_i Ai) \times 100\%$$

$A = H \cdot W$ $\quad$ $A_i = w_i \times h_i$

Design Style Specific Floorplanning Problems

- Floorplanning is not carried out for some design styles.
- Some design styles have blocks with fixed dimentions.
  - Full Custom:
    - Floorplanning needs to be performed.
  - Standard Cell:
    - Dimensions of cells are fixed. Therefore, floorplanning is simply a placement problem. Neither floorplanning nor pin assignment is required.
    - For large standard cells, floorplanning may be required if the cell is partitioned into several blocks. (Cell-level floorplanning)
  - Gate Array:
    - Floorplanning problem is placement problem.

Classification of Algorithms

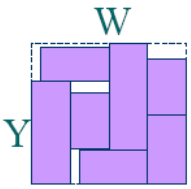
- Floorplanning methods can be classified as:
  - Constraint Based methods
    - Construct a floorplan of **optimal area** that satisfies a given set of constraints.
  - (Integer) Linear Programming Methods
  - Rectangular Dualization Based Methods
  - Hierarchical Tree Based Methods
  - Simulated Annealing and Genetic Algorithms
  - Timing Driven Floorplanning Algorithms

Integer Programming Based Floorplanning

- **LP**:
  - The objective is a linear function.
  - All constraints are linear functions.
  - Some variables are real numbers and some are integers, i.e., "*mixed integer*".
- It is almost like a linear program, except that some variables are integers.

Problem Formulation

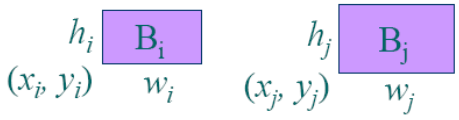
- **Minimize the packing area:**
  - Assume that one dimension W is fixed.
  - Minimize the other dimension Y.
- Need to have constraints
  - Overlap constraints
    - Prevent any two blocks from overlapping
  - Routability constraints
    - Estimate the routing area required between the blocks
- Associate each block B_i with 4 variables:
  - x_i and y_i: coordinates of its lower left corner.
  - w_i and h_i: width and height.



Non-overlapping Constraints for Fixed Blocks

- For two non-overlapping blocks B_i and B_j, at least one of the following four linear constraints must be satisfied:

(1)	$x_i + w_i \le x_j$	if B_i is to the left of B_j
or (2)	$x_i - w_j \ge x_j$	if B_i is to the right of B_j
or (3)	$y_i + h_i \le y_j$	if B_i is below B_j
or (4)	$y_i - h_j \ge y_j$	if B_i is above B_j



The "or" condition is not understood in LP.

Integer Variables

- Use integer (0 or 1) variables x_{ij} and y_{ij}:
  - $x_{ij}=0$ and $y_{ij}=0$ if (1) is true.
  - $x_{ij}=0$ and $y_{ij}=1$ if (2) is true.
  - $x_{ij}=1$ and $y_{ij}=0$ if (3) is true.
  - $x_{ij}=1$ and $y_{ij}=1$ if (4) is true.
- Let W and H be upper bounds on the total width and height. Non-overlapping constraints:

$$
\begin{aligned}
&(1') \quad x_i + w_i \le x_j + W(x_{ij} + y_{ij}) \\
&(2') \quad x_i - w_j \ge x_j - W(1 + x_{ij} - y_{ij}) \\
&(3') \quad y_i + h_i \le y_j + H(1 - x_{ij} + y_{ij}) \\
&(4') \quad y_i - h_j \ge y_j - H(2 - x_{ij} - y_{ij})
\end{aligned}
$$

- Only one of the above equations will be active and other equations will be true depending on the value of x_{ij} and y_{ij}.

Formulation

$$
\begin{aligned}
\text{Min.} \quad & Y \\
\text{s.t.} \quad & 0 \le x_i,\ x_i + w_i \le W \\
& 0 \le y_i,\ y_i + h_i \le Y \\
& x_i + w_i \le x_j + W(x_{ij} + y_{ij}) \\
& x_i - w_j \ge x_j - W(1 + x_{ij} - y_{ij}) \\
& y_i + h_i \le y_j + H(1 - x_{ij} + y_{ij}) \\
& y_i - h_j \ge y_j - H(2 - x_{ij} - y_{ij}) \\
& x_{ij} = 0 \text{ or } 1 \\
& y_{ij} = 0 \text{ or } 1
\end{aligned}
$$

Formulation with Hard Blocks

- If the blocks can be rotated, use a 0-1 integer variable z_i for each block B_i, $z_i = 0$ if B_i is in the original orientation and $z_i = 1$ if B_i is rotated 90° .

Min. Y

s.t.
$$0 \le x_i,\ x_i + z_i h_i + (1 - z_i) w_i \le W$$
$$0 \le y_i,\ y_i + z_i w_i + (1 - z_i) h_i \le Y$$
$$x_i + z_i h_i + (1 - z_i) w_i \le x_j + W(x_{ij} + y_{ij})$$
$$x_i - z_j h_j - (1 - z_j) w_j \ge x_j - W(1 + x_{ij} - y_{ij})$$
$$y_i + z_i w_i + (1 - z_i) h_i \le y_j + H(1 - x_{ij} + y_{ij})$$
$$y_i - z_j w_j - (1 - z_j) h_j \ge y_j - H(2 - x_{ij} - y_{ij})$$
$$x_{ij} = 0 \text{ or } 1$$
$$y_{ij} = 0 \text{ or } 1$$

Non-overlapping Constraints for Flexible Blocks

- Flexible blocks can take rectangular shapes within a limited aspect ratio range.
- If B_i is a soft block, $w_i h_i = A_i$. But this constraint is quadratic!
- This non-linear area relation is linearized by taking the first two terms of the Taylor expression of $h_i = A_i / w_i$ at w_{imax} (maximum width of block B_i).

$$h_i = h_{imin} + \Delta w_i \lambda_i$$
$$h_i = h_{imin} + \lambda_i (w_{imax} - w_i)$$

where $h_{imin} = A_i / w_{imax}$ and $\lambda_i = A_i / w_{imax}^2$

Formulation with Soft Blocks

- If B_i is soft and B_j is hard:

$$
\begin{aligned}
&(1) \quad x_i + w_i \le x_j + W(x_{ij} + y_{ij}) \\
&(2) \quad x_i - w_j \ge x_j - W(1 + x_{ij} - y_{ij}) \\
&(3) \quad y_i + h_{imin} + \lambda_i(w_{imax} - w_i) \le y_j + H(1 - x_{ij} + y_{ij}) \\
&(4) \quad y_i - h_j \ge y_j - H(2 - x_{ij} - y_{ij})
\end{aligned}
$$

- If both B_i and B_j are soft:

$$
\begin{aligned}
&(1) \quad x_i + w_i \le x_j + W(x_{ij} + y_{ij}) \\
&(2) \quad x_i - w_j \ge x_j - W(1 + x_{ij} - y_{ij}) \\
&(3) \quad y_i + h_{imin} + \lambda_i(w_{imax} - w_i) \le y_j + H(1 - x_{ij} + y_{ij}) \\
&(4) \quad y_i - h_{jmin} - \lambda_j(w_{jmax} - w_j) \ge y_j - H(2 - x_{ij} - y_{ij})
\end{aligned}
$$

Solving Linear Program

- Linear Programming (LP) can be solved by classical optimization techniques in *polynomial time*.
- CPLEX can be used to solve the problem.

- Floorplanning is a mixed integer linear programming (MILP) problem.
- MILP is NP-Complete.
  - The run time of the best known algorithm is exponential to the number of variables and equations.

Complexity

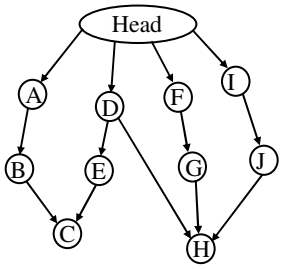
- For a problem with n blocks, and for the simplest case, *i.e.*, all blocks are hard:
  - $4n$ continuous variables (x_i, y_i, w_i, h_i)
  - $n(n-1)$ integer variables (x_{ij}, y_{ij})
  - $2n^2$ linear constraints
- Practically, this method can only solve small problems ($n \approx 10$).

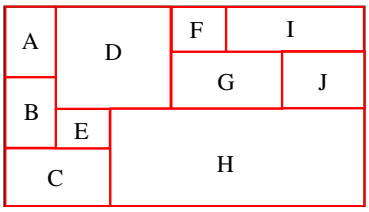
Rectangular Dualization

- It's a **heuristic**.
- The output from a partitioning algorithm can be represented as a graph G=(V,E).
  - The vertices of the graph correspond to the subcircuits and the edges represent the interconnections between the subcircuits.
- Rectangular dualization:
  - The floorplan can be obtained by converting the graph into its rectangular dual.
- **Advantage:**
  - Use of rectangular dualization maximizes adjacency of blocks that are heavily connected.

Rectangular Dualization

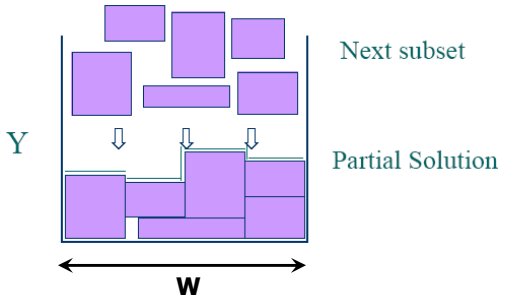
If there is no connections between E and J, then they should not be next to each other in the layout surface.





Successive Augmentation

- A classical **greedy** approach to keep the problem size small: Repeatedly pick a small subset of blocks to formulate a MILP, solve it together with the previously picked blocks with fixed locations and shapes:

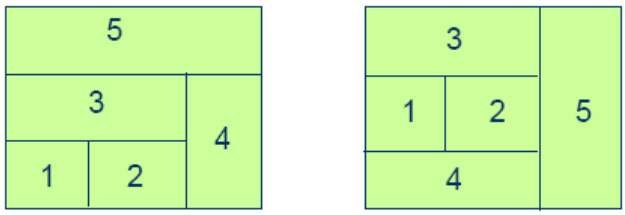


Floorplanning Using SA and GA

- Floorplanning using simulated annealing and genetic algorithms requires correct problem representation.
- Few things to think about before implementing SA or GA:
  - **GA**:
    - Fitness **Encoding,** Crossover, Mutation
  - **SA**
    - **Encoding,** Fitness, Boltzman Distribution, Cooling Schedule, Neighborhood Function

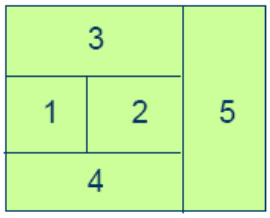
Terminology

- **Rectangular Dissection**
  - It is a subdivision of a given rectangle by a finite number of horizontal and vertical line segments into a finite number of non-overlapping rectangles.



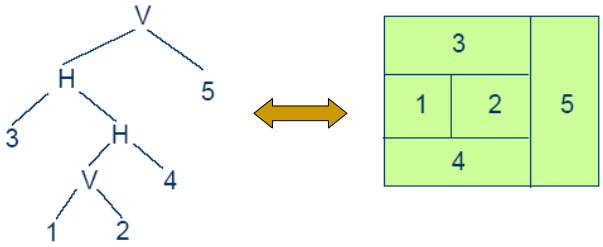
Terminology

- **Slicing Structure**
  - A rectangular dissection that can be obtained by repetitively subdividing rectangles horizontally or vertically.



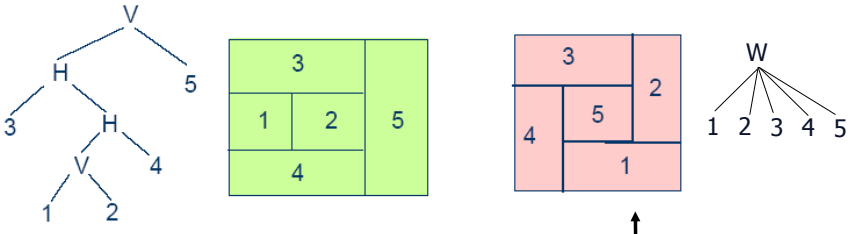
Terminology

- **Slicing Tree**
  - A slicing structure can be modeled by a binary tree with *n* leaves and *n-1* nodes, where each node represents a vertical cut line or a horizontal cut line, and each leaf a basic rectangle.



Slicing and Non-Slicing Floorplans

- A floorplan that corresponds to a slicing structure is called a *slicing floorplan*.

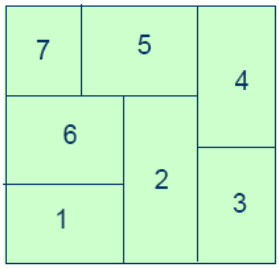


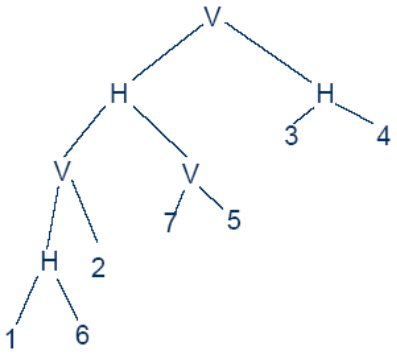
- Otherwise it is a *non-slicing floorplan*.
  - Often called as ***wheel***.

Terminology

- An expression, $E=e_1,e_2,\ldots,e_{2n-1}$, *where* $e_i \epsilon \{1,2,\ldots,n,H,V\}$, $1 \leq i \leq 2n-1$, is a *Polish Expression* of length $2n-1$ *iff*
  - every operand j, $1 \leq j \leq n$, appears exactly once in the expression, and
  - the expression E has the balloting property, *i.e.*, for every sub-expression $E_i=e_1,e_2,\ldots,e_i$, $1 \leq i \leq 2n-1$, the number of operands is greater than the number of operators.

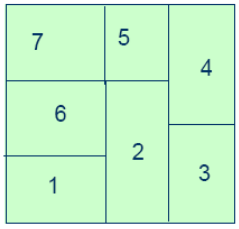
Example

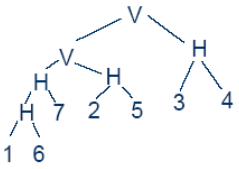




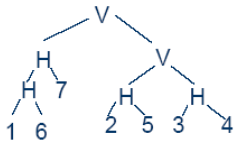
Polish Expression E=16H2V75VH34HV

Example

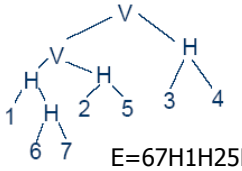




E=16H7H25HV34HV



E=34H25HV16H7HV



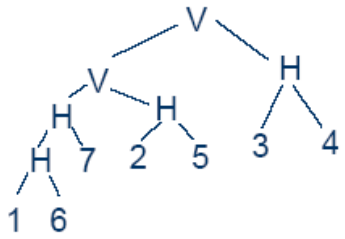
E=67H1H25HV34HV

One Layout and many slicing trees !!

Normalized Polish Expressions

- An expression, $E=e_1,e_2,\ldots,e_{2n-1}$, *where* $e_i \in \{1,2,\ldots,n,H,V\}$, $1 \le i \le 2n-1$, is a *Normalized Polish* expression of length *iff E* has no consecutive *H*'s and *V*'s.

Example:



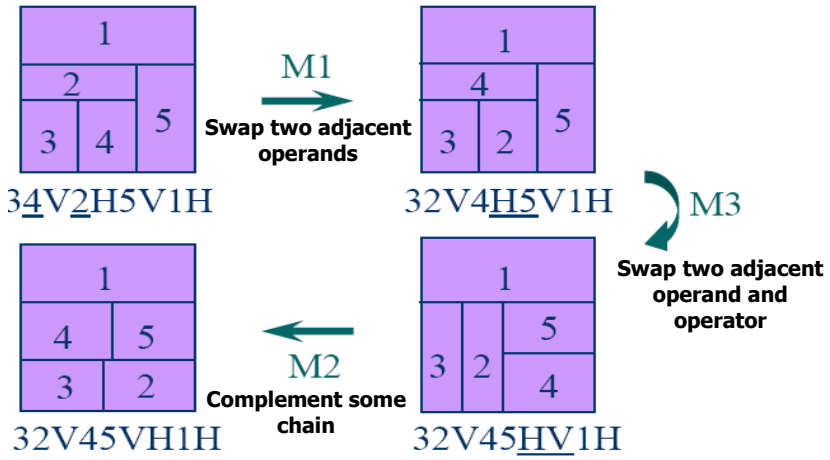
E=16H7H25HV34HV

Simulated Annealing Algorithm

- Represent floorplan by normalized polish expression
- Perturb using moves
  - **M1**: Swap two adjacent operands
  - **M2**: Complement some chain
  - **M3**: Swap two adjacent operand and operator
- Optimize cost

$$\text{Cost} = \alpha.A + \beta L$$

Examples of Moves



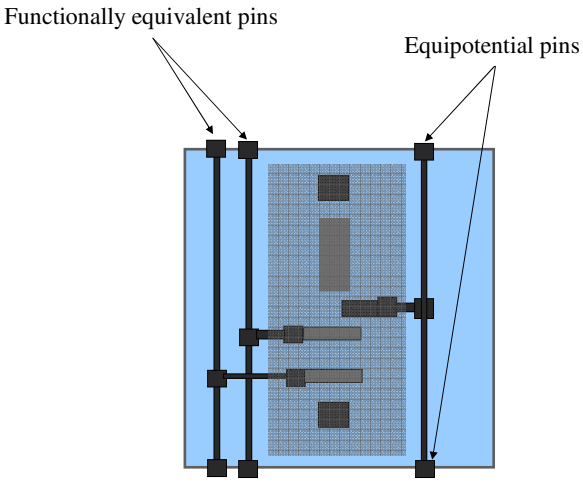
Pin Assignment

- The purpose of pin assignment is to define the signal that each pin will receive.
- Pin assignment may be done during floorplanning, placement or after placement is fixed.
  - If the blocks are not designed then good assignment of nets to pins can improve the placement.
  - If the blocks are already designed, it may be possible to exchange a few pins.

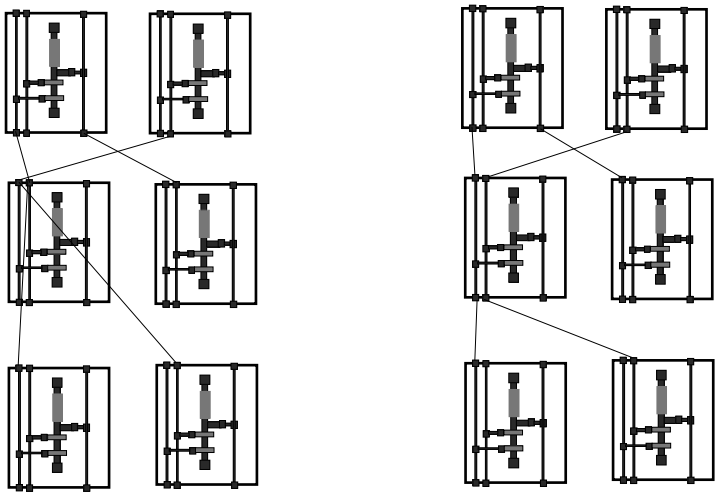
Pin Assignment

- **Functionally equivalent pins:**
  - Two pins are functionally equivalent if exchanging signals does not affect the circuit.
    - E.g. two inputs of a gate are functionally equivalent.
- **Equipotential pins:**
  - Two pins are equipotential pins if both are internally connected and hence represent the same net.
    - E.g. the output of a gate is equipotential pin if it is available on both sides.

Example



Impact of Pin Assignment



Problem Formulation

- Optimize the assignment of nets within a functionally equivalent pin groups or assignment of nets within an equipotentially pin group. That results in:
- Objective:
  - Minimize congestion or the number of crossovers.