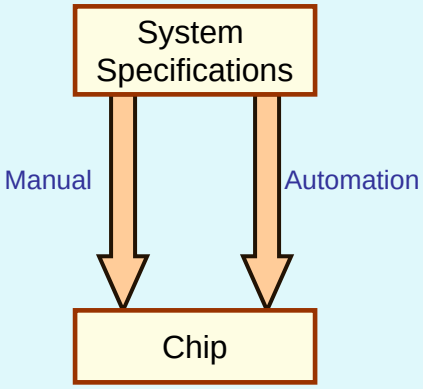
BACKEND DESIGN

Basic Concepts in VLSI Physical Design Automation

VLSI Design Cycle

- Large number of devices
- Optimization requirements for high performance
- Time-to-market competition
- Cost

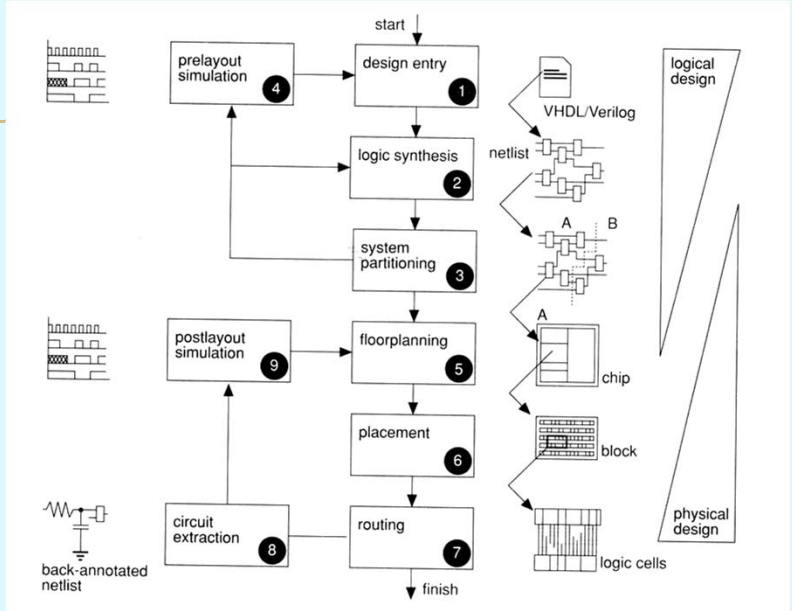


VLSI Design Cycle (contd.)

1. System specification
2. Functional design
3. Logic design
4. Circuit design
5. ***Physical design***
6. Design verification
7. Fabrication
8. Packaging, testing, and debugging

Physical Design

- Converts a circuit description into a geometric description.
  - This description is used for fabrication of the chip.
- Basic steps in the physical design cycle:
  1. Partitioning
  2. Floorplanning
  3. Placement
  4. Routing – global and detailed
  5. Clock and power routing
  6. Others ….



Floorplanning

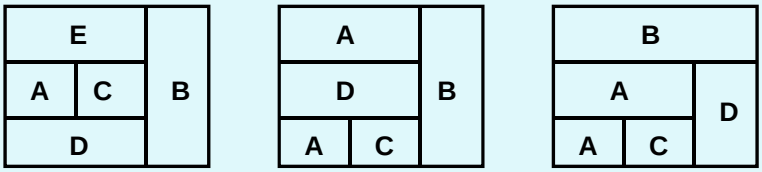
Problem Definition

- **Input:**
  - A set of blocks, both fixed and flexible.
    - Area of the block $A_i = w_i \times h_i$
    - Constraint on the shape of the block (rigid/flexible)
  - Pin locations of fixed blocks.
  - A netlist.
- **Requirements:**
  - Find locations for each block so that no two blocks overlap.
  - Determine shapes of flexible blocks.
- **Objectives:**
  - Minimize area.
  - Reduce wire-length for critical nets.

An Example for Rigid Blocks

Module	Width	Height
A	1	1
B	1	3
C	1	1
D	1	2
E	2	1

Some of the Feasible Floorplans



Design Style Specific Issues

- Full Custom
  - All the steps required for general cells.
- Standard Cell
  - Dimensions of all cells are fixed.
  - Floorplanning problem is simply the placement problem.
  - For large netlists, two steps:
    - First do global partitioning.
    - Placement for individual regions next.
- Gate Array
  - Floorplanning problem same as placement problem.

Estimating Cost of a Floorplan

- The number of feasible solutions of a floorplanning problem is very large.
  - Finding the best solution is NP-hard.
- Several criteria used to measure the quality of floorplans:
  a) Minimize area
  b) Minimize total length of wire
  c) Maximize routability
  d) Minimize delays
  e) Any combination of above

Contd.

- How to determine area?
  - Not difficult.
  - Can be easily estimated because the dimensions of each block is known.
  - Area *A* computed for each candidate floorplan.
- How to determine wire length?
  - A coarse measure is used.
  - Based on a model where all I/O pins of the blocks are merged and assumed to reside at its center.
  - Overall wiring length $L = \Sigma_{i,j} (c_{ij} * d_{ij})$

    where c_{ij} : connectivity between blocks i and j

    d_{ij} : Manhattan distances between the centres of rectangles of blocks i and j

Contd.

- **Typical cost function used:**

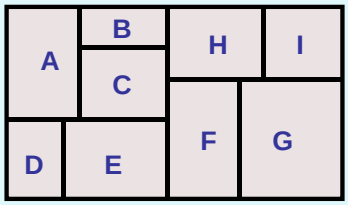
  Cost = w1 * A + w2 * L

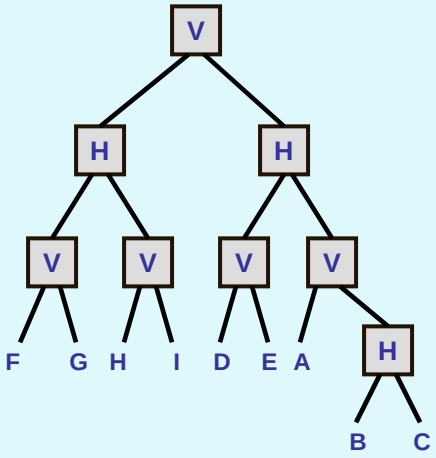
  where w1 and w2 are user-specified parameters.

Slicing Structure

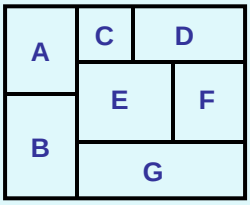
- **Definition**
  - **A rectangular dissection that can be obtained by repeatedly splitting rectangles by horizontal and vertical lines into smaller rectangles.**
- **Slicing Tree**
  - **A binary tree that models a slicing structure.**
  - **Each node represents a *vertical cut line* (V), or a *horizontal cut line* (H).**
    - **A third kind of node called *Wheel (W)* appears for non-sliceable floorplans (discussed later).**
  - **Each leaf is a basic block (rectangle).**

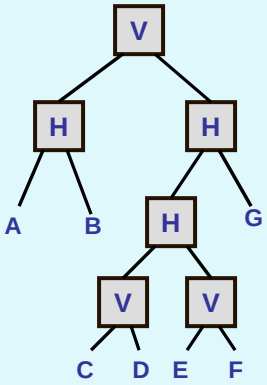
A Slicing Floorplan and its Slicing Tree

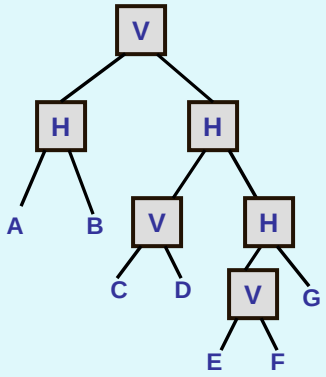




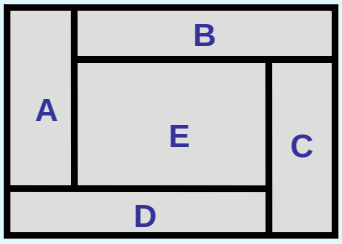
Slicing Tree is not Unique

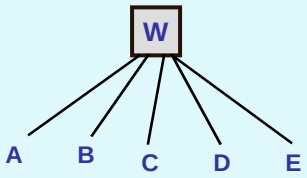






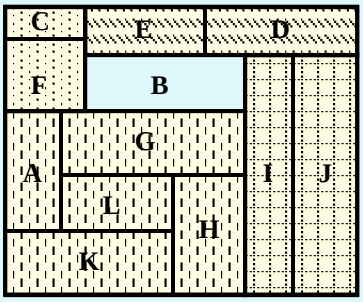
A Non-Slicing Floorplan





Also called “WHEEL”

A Hierarchical Floorplan



The dissection tree will contain “wheel”.

Floorplanning Algorithms

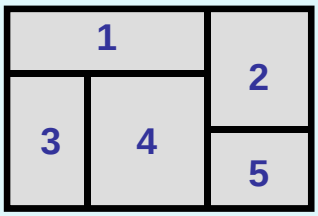
- Several broad classes of algorithms:
  - Integer programming based
  - Rectangular dual graph based
  - Hierarchical tree based
  - Simulated annealing based
  - Other variations

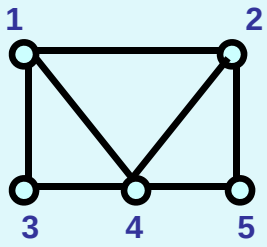
Rectangular Dual-Graph Approach

- Basic Concept:
  - Output of partitioning algorithms represented by a graph.
  - Floorplans can be obtained by converting the graph into its rectangular dual.
- The rectangular dual of a graph satisfies the following properties:
  - Each vertex corresponds to a distinct rectangle.
  - For every edge, the corresponding rectangles are adjacent.

A Rectangular Floorplan & its Dual Graph

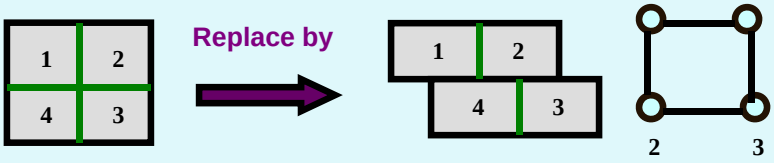
- Without loss of generality, we assume that a rectangular floorplan contains no cross junctions.
- Under this assumption, the dual graph of a rectangular floorplan is a ***planar triangulated graph (PTG).***



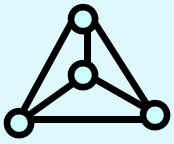


Contd.

- Every dual graph of a rectangular floorplan (without cross junction) is a PTG.



- However, not every PTG corresponds to a rectangular floorplan.



Complex triangle

Drawbacks

- A new approach to floorplanning, in which many sub-problems are still unsolved.
- The main problem concerns the existence of the rectangular dual, i.e. the elimination of complex triangles.
  - Select a minimum set E of edges such that each complex triangle has at least one edge in E.
  - A vertex can be added to each edge of E to eliminate all complex triangles.
  - The weighted complex triangle elimination problem has been shown to be NP-complete.
    - Some heuristics are available.

Placement

The Placement Problem

- **Inputs:**
  - A set of modules with
    - well-defined shapes
    - fixed locations of pins.
  - A netlist.
- **Requirements:**
  - Find locations for each module so that no two modules overlap.
  - The placement is routable.
- **Objectives:**
  - Minimize layout area.
  - Reduce the length of critical nets.
  - Completion of routing.

Problem Formulation

- **Notations:**

$B_1, B_2, \ldots, B_n$: modules/blocks to be placed

w_i, h_i : width and height of B_i, $1 \le i \le n$

$N=\{N_1, N_2, \ldots, N_m\}$: set of nets (i.e. the netlist)

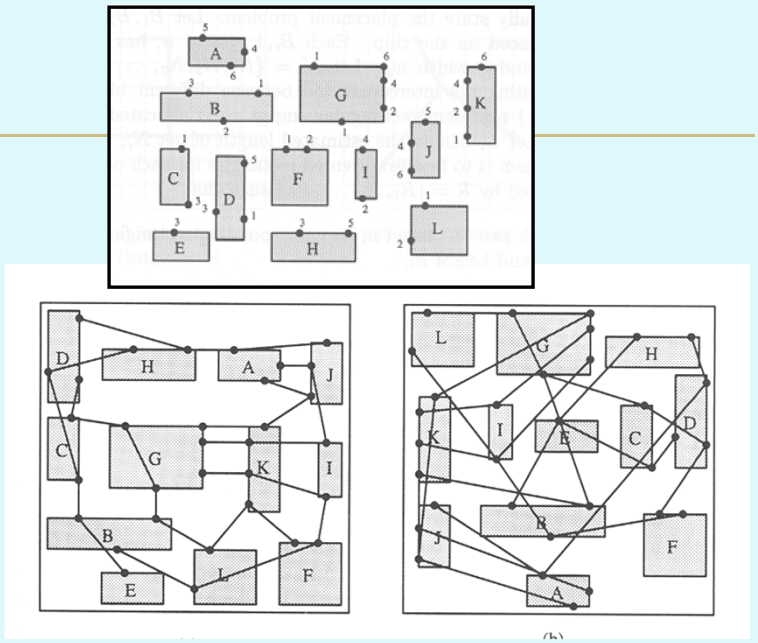
$Q=\{Q_1, Q_2, \ldots, Q_k\}$: rectangular empty spaces for routing

L_i : estimated length of net N_i, $1 \le i \le m$

Contd.

- **The problem**

  Find rectangular regions $R=\{R_1,R_2,...R_n\}$ for each of the blocks such that
  - **Block B_i can be placed in region R_i.**
  - **No two rectangles overlap, $R_i \cap R_j = \Phi$.**
  - **Placement is routable (Q is sufficient to route all nets).**
  - **Total area of rectangle bounding R and Q is minimized.**
  - **Total wire length ΣL_i is minimized.**
  - **For high performance circuits, max $\{L_i \mid i=1,2,\ldots,m\}$ is minimized.**
- **General problem is NP-complete.**
- **Algorithms used are heuristic in nature.**



Interconnection Topologies

- The actual wiring paths are not known during placement.
  - For making an estimation, a placement algorithm needs to model the topology of the interconnection nets.
    - An interconnection graph structure is used.
    - Vertices are terminals, and edges are interconnections.

Estimation of Wirelength

- The speed and quality of estimation has a drastic effect on the performance of placement algorithms.
  - For 2-terminal nets, we can use Manhattan distance as an estimate.
  - If the end co-ordinates are (x_1,y_1) and (x_2,y_2), then the wire length

    $L = |x_1 - x_2| + |y_1 - y_2|$
- How to estimate length of multi-terminal nets?

Modeling of Multi-terminal Nets

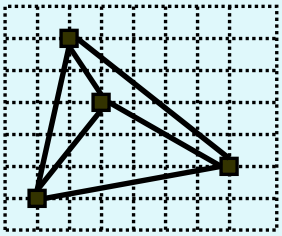
1. **Complete Graph**
   - ${}^{n}C_2 = n(n-1)/2$ edges for a n-pin net.
   - A tree has (n-1) edges which is 2/n times the number of edges of the complete graph.
   - Length is estimated as 2/n times the sum of the edge weights.
2. **Minimum Spanning Tree**
   - Commonly used structure.
   - Branching allowed only at pin locations.
   - Easy to compute.

Contd.

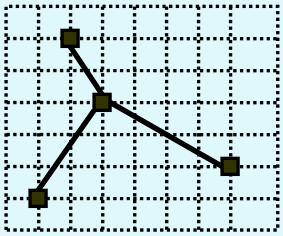
3. **Rectangular Steiner Tree**
   - A Steiner tree is the shortest route for connecting a set of pins.
   - A wire can branch from any point along its length.
   - Problem of finding Steiner tree is NP-complete.
4. **Semi Perimeter**
   - Efficient and most widely used.
   - Finds the smallest bounding rectangle that encloses all the pins of the net to be connected.
   - Estimated wire length is half the perimeter of this rectangle.
   - Always underestimates the wire length for congested nets.

Example

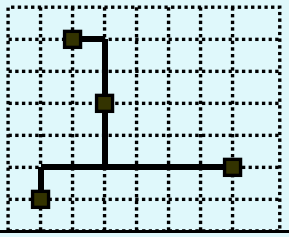
Complete Graph



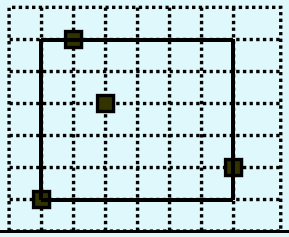
Minimum Spanning Tree



Steiner Tree



Semi Perimeter

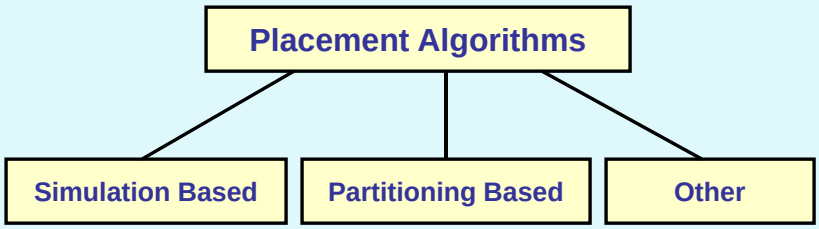


Design Style Specific Issues

- **Full Custom**
  - Placing a number of blocks of various shapes and sizes within a rectangular region.
  - Irregularity of block shapes may lead to unused areas.
- **Standard Cell**
  - Minimization of the layout area means:
    - Minimize sum of channel heights.
    - Minimize width of the widest row.
    - All rows should have equal width.
  - Over-the-cell routing leads to almost “channel-less” standard cell designs.

- **Gate Arrays**
  - The problem of partitioning and placement are the same in this design style.
  - For FPGA's, the partitioned sub-circuit may be a complex netlist.
    - Map the netlist to one or more basic blocks (placement).

Classification of Placement Algorithms



Simulation Based	Partitioning Based	Other
Simulated Annealing	Breuer's Algorithm	Cluster Growth
Simulated Evolution	Terminal Propagation	Force Directed
Force Directed		

Simulated Annealing

- Simulation of the annealing process in metals or glass.
  - Avoids getting trapped in local minima.
  - Starts with an initial placement.
  - Incremental improvements by exchanging blocks, displacing a block, etc.
  - Moves which decrease cost are always accepted.
  - Moves which increase cost are accepted with a probability that decreases with the number of iterations.
- Timberwolf is one of the most successful placement algorithms based on simulated annealing.

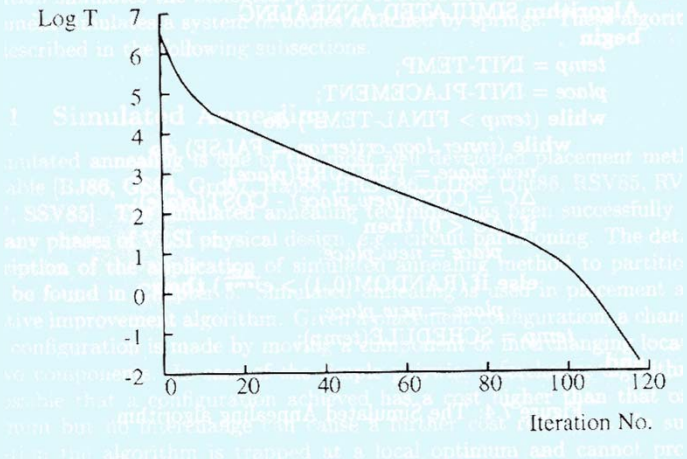
Simulated Annealing Algorithm

```
Algorithm  SA_Placement
begin
   T = initial_temperature;
   P = initial_placement;
   while  ( T > final_temperature)  do
      while  (no_of_trials_at_each_temp not yet completed) do
         new_P = PERTURB (P);
         ΔC = COST (new_P) – COST (P);
         if  (ΔC < 0)  then
            P = new_P;
        else if  (random(0,1) > exp(ΔC/T))  then
                 P = new_P;
      T = SCHEDULE (T);       /** Decrease temperature **/
end
```

TimberWolf

- One of the most successful placement algorithms.
  - Developed by Sechen and Sangiovanni-Vincentelli.
- Parameters used:
  - Initial_temperature = 4,000,000
  - Final_temperature = 0.1
  - SCHEDULE(T) = α(T) x T
    - α(T) specifies the cooling rate which depends on the current temperature.
    - α(T) is 0.8 when the cooling process just starts.
    - α(T) is 0.95 in the medium range of temperature.
    - α(T) is 0.8 again when temperature is low.

The SCHEDULE Function



The PERTURB Function

- New configuration is generated by making a weighted random selection from one of the following:
  a) The displacement of a block to a new location.
  b) The interchange of locations between two blocks.
  c) An orientation change for a block.
     – Mirror image of the block's x-coordinate.
     – Used only when a new configuration generated using alternative (a) is rejected.

The COST Function

- The cost of a solution is computed as:

  COST = cost1 + cost2 + cost3

  where cost1 : weighted sum of estimated length of all nets
  cost2 : penalty cost for overlapping
  cost3 : penalty cost for uneven length among standard cell rows.

  – Overlap is not allowed in placement.
  – Computationally complex to remove all overlaps.
  – More efficient to allow overlaps during intermediate placements.
    • Cost function (cost2) penalizes the overlapping.

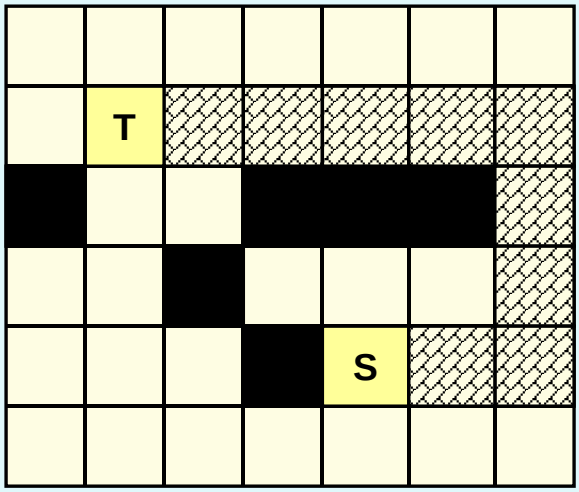
Grid Routing

Introduction

- In the VLSI design cycle, *routing* follows *cell placement*.
- During routing, precise paths are defined on the layout surface, on which conductors carrying electrical signals are run.
- Routing takes up almost 30% of the design time, and a large percentage of layout area.
- We first take up the problem of grid routing.

What is Grid Routing?

- The layout surface is assumed to be made up of a rectangular array of grid cells.
- Some of the grid cells act as obstacles.
  - Blocks that are placed on the surface.
  - Some nets that are already laid out.
- Objective is to find out a path (sequence of grid cells) for connecting two points belonging to the same net.
- Two broad class of algorithms:
  - Maze routing algorithms.
  - Line search algorithms.



Problem Definition

- The general routing problem is defined as follows.
- Given:
  - A set of blocks with pins on the boundaries.
  - A set of signal nets.
  - Locations of blocks on the layout floor.
- Objective:
  - Find suitable paths on the available layout space, on which wires are run to connect the desired set of pins.
  - Minimize some given objective function, subject to given constraints.

Contd.

- Types of constraints:
  - Minimum width of routing wires.
  - Minimum separation between adjacent wires.
  - Number of routing layers available.
  - Timing constraints.

Grid Routing Algorithms

1. Maze running algorithm
   - Lee's algorithm
   - Hadlock's algorithm
2. Line search algorithm
   - Mikami-Tabuchi's algorithm
   - Hightower's algorithm
3. Steiner tree algorithm

Maze Running Algorithms

- The entire routing surface is represented by a 2-D array of grid cells.
  - All pins, wires and edges of bounding boxes that enclose the blocks are aligned with respect to the grid lines.
  - The segments on which wires run are also aligned.
  - The size of grid cells is appropriately defined.
    - Wires belonging to different nets can be routed through adjacent cells without violating the width and spacing rules.
- Maze routers connect a single pair of points at a time.
  - By finding a sequence of adjacent cells from one point to the other.

Lee's Algorithm

- The most common maze routing algorithm.
- Characteristics:
  - If a path exists between a pair of points S and T, it is definitely found.
  - It always finds the shortest path.
  - Uses breadth-first search.
- Time and space complexities are $O(N^2)$ for a grid of dimension $N \times N$.

Phase 1 of Lee's Algorithm

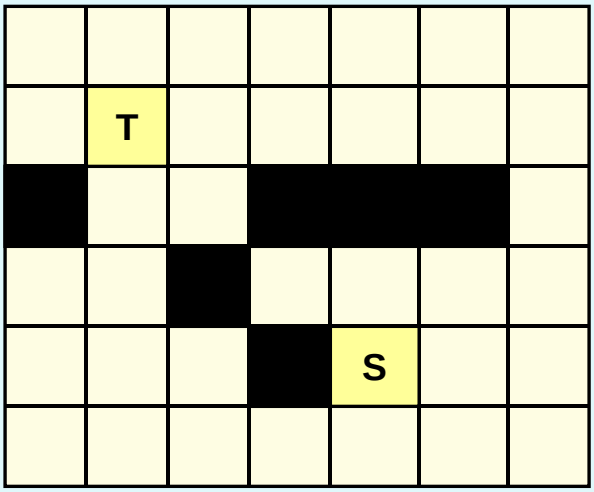
- Wave propagation phase
  - Iterative process.
  - During step i, non-blocking grid cells at Manhattan distance of i from grid cell S are all labeled with i.
  - Labeling continues until the target grid cell T is marked in step L.
    - L is the length of the shortest path.
  - The process fails if:
    - T is not reached and no new grid cells can be labeled during step i.
    - T is not reached and i equals M, some upper bound on the path length.

Phase 2 of Lee's Algorithm

- **Retrace phase**
  - Systematically backtrack from the target cell T back towards the source cell S.
  - If T was reached during step i, then at least one grid cell adjacent to it will be labeled i-1, and so on.
  - By tracing the numbered cells in descending order, we can reach S following the shortest path.
    - There is a choice of cells that can be made in general.
    - In practice, the rule of thumb is not to change the direction of retrace unless one has to do so.
    - Minimizes number of bends.

Phase 3 of Lee's Algorithm

- **Label clearance**
  - All labeled cells except those corresponding to the path just found are cleared.
  - Search complexity is as involved as the wave propagation step itself.



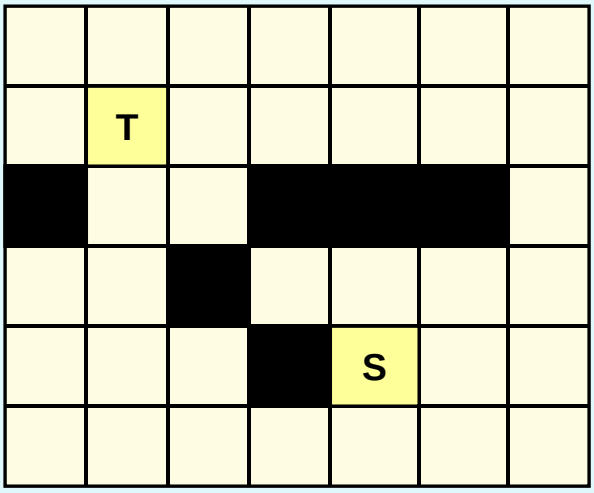
- **Memory Requirement**
  - Each cell needs to store a number between 1 and L, where L is some bound on the maximum path length.
  - One bit combination to denote empty cell.
  - One bit combination to denote obstacles.

$\log_2(L+2)$ bits per cell

- **Improvements**:
  - Instead of using the sequence 1,2,3,4,5,….. for numbering the cells, the sequence 1,2,3,1,2,3,… is used.
    - For a cell, labels of predecessors and successors are different. So tracing back is easy.

      $\log_2(3+2) = 3$ bits per cell.
  - Use the sequence 0,0,1,1,0,0,1,1,…..
    - Predecessors and successors are again different.

      $\log_2(2+2) = 2$ bits per cell.



Reducing Running Time

- **Starting point selection**
  - Choose the starting point as the one that is farthest from the center of the grid.
- **Double fan-out**
  - Propagate waves from both the source and the target cells.
  - Labeling continues until the wavefronts touch.
- **Framing**
  - An artificial boundary is considered outside the terminal pairs to be connected.
  - 10-20% larger than the smallest bounding box.

Global Routing

Basic Idea

- The routing problem is typically solved using a two-step approach:
  - Global Routing
    - Define the routing regions.
    - Generate a tentative route for each net.
    - Each net is assigned to a set of routing regions.
    - Does not specify the actual layout of wires.
  - Detailed Routing
    - For each routing region, each net passing through that region is assigned particular routing tracks.
    - Actual layout of wires gets fixed.
    - Associated subproblems: channel routing and switchbox routing.

Routing Regions

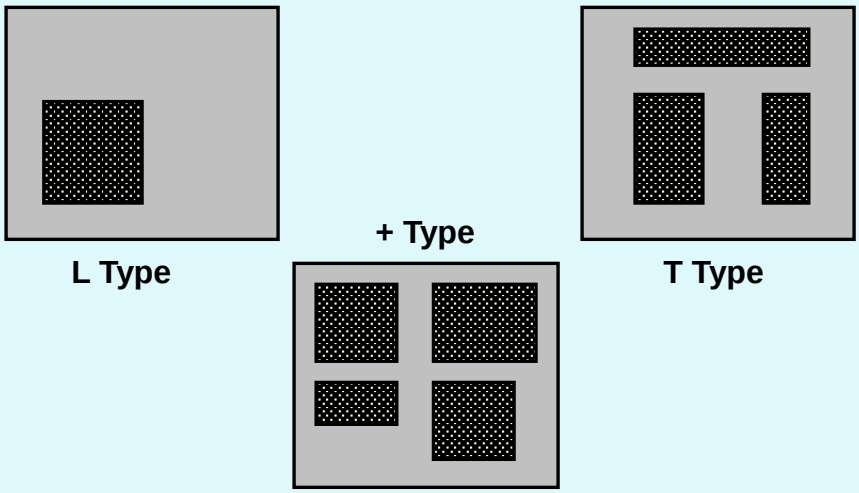
- Regions through which interconnecting wires are laid out.
- How to define these regions?
  - Partition the routing area into a set of non-intersecting rectangular regions.
  - Types of routing regions:
    - Horizontal channel: parallel to the x-axis with pins at their top and bottom boundaries.
    - Vertical channel: parallel to the y-axis with pins at their left and right boundaries.
    - Switchbox: rectangular regions with pins on all four sides.

- **Points to note:**
  - Identification of routing regions is a crucial first step to global routing.
  - Routing regions often do not have pre-fixed capacities.
  - The order in which the routing regions are considered during detailed routing plays a vital part in determining overall routing quality.

Types of Channel Junctions

- **Three types of channel junctions may occur:**
  - **L-type:**
    - Occurs at the corners of the layout surface.
    - Ordering is not important during detailed routing.
    - Can be routed using channel routers.
  - **T-type:**
    - The leg of the "T" must be routed before the shoulder.
    - Can be routed using channel routers.
  - **+-type:**
    - More complex and requires switchbox routers.
    - Advantageous to convert +-junctions to T-junctions.

Illustrations



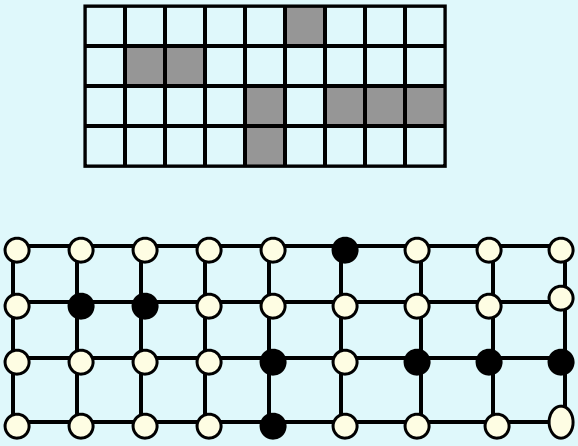
Graph Models used in Global Routing

- **Global routing is typically studied as a graph problem.**
  - **Routing regions and their relationships modeled as graphs.**
- **Three important graph models:**
  1. **Grid Graph Model**
     - Most suitable for area routing
  2. **Channel Intersection Graph Model**
     - Most suitable for global routing

Grid Graph Model

- A layout is considered to be a collection of unit side square cells (grid).
- Define a graph:
  - Each cell c_i is represented as a vertex v_i.
  - Two vertices v_i and v_j are joined by an edge if the corresponding cells c_i and c_j are adjacent.
  - A terminal in cell c_i is assigned to the corresponding vertex v_i.
  - The occupied cells are represented as filled circles, whereas the others as clear circles.
  - The capacity and length of each edge is set to one.
- Given a 2-terminal net, the routing problem is to find a path between the corresponding vertices in the grid graph.

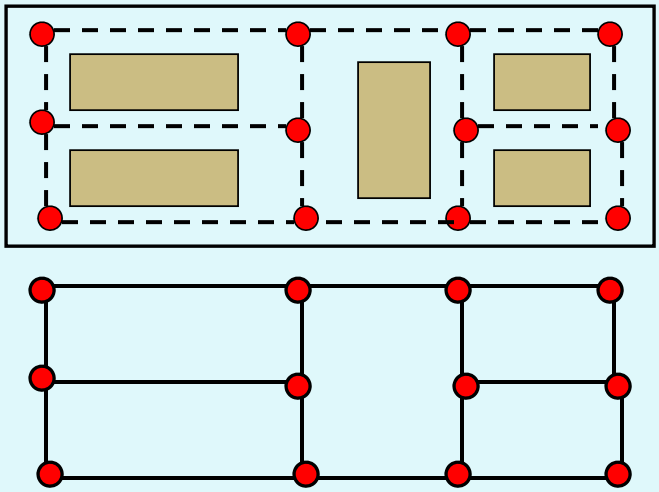
Grid Graph Model :: Illustration



Channel Intersection Graph

- Most general and accurate model for global routing.
- Define a graph:
  - Each vertex v_i represents a channel intersection CI_i.
  - Channels are represented as edges.
  - Two vertices v_i and v_j are connected by an edge if there exists a channel between CI_i and CI_j.
  - Edge weight represents channel capacity.

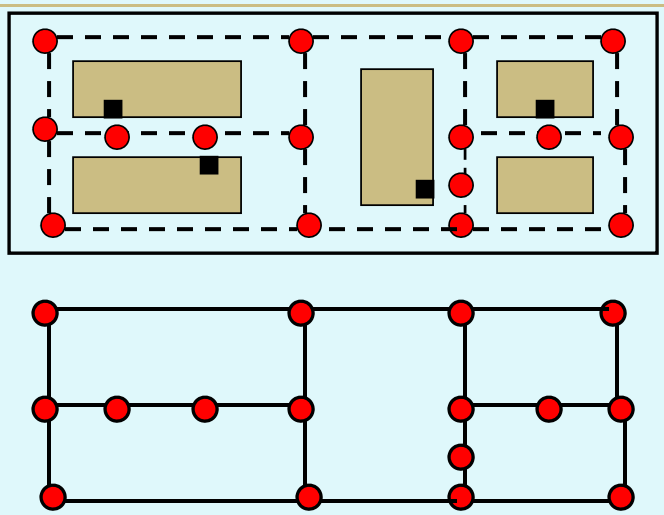
Illustration



Extended Channel Intersection Graph

- Extension of the channel intersection graph.
  - Includes the pins as vertices so that the connections between the pins can be considered.
- The global routing problem is simply to find a path in the channel intersection graph.
  - The capacities of the edges must not be violated.
  - For 2-terminal nets, we can consider the nets sequentially.
  - For multi-terminal nets, we can have an approximation to minimum Steiner tree.

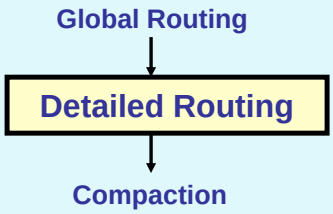
Illustration



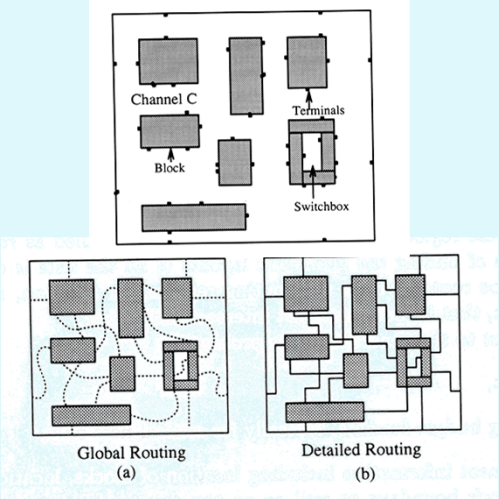
Detailed Routing

Detailed Routing

- Find actual geometric layout of each net within assigned routing regions.
- No layouts of two different nets should intersect on the same layer.
- Problem is solved incrementally, one region at a time in a predefined order.



A Routing Example

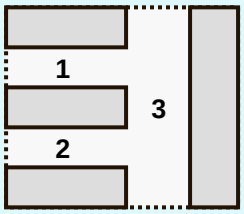


After Global Routing

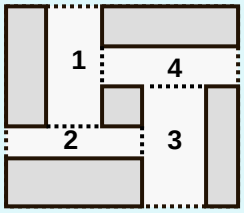
- The two-stage routing method is a powerful technique for routing in VLSI circuits.
- During the global routing stage
  - The routing region is partitioned into a collection of rectangular regions.
  - To interconnect each net, a sequence of sub-regions to be used is determined.
  - All nets crossing a given boundary of a routing region are called ***floating terminals***.
  - Once the sub-region is routed, these floating terminals become fixed terminals for subsequent regions.

Order of Routing Regions

- Slicing placement topology
- Nets can be routed by considering channels 1, 2 and 3 in order.



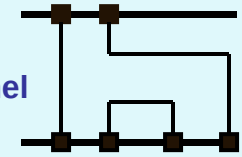
- Non-slicing placement topology.
- Channels with cyclic constraints.
- Some of the routing regions are to be considered as switchboxes.

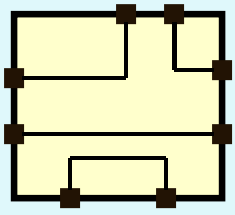


Channels and Switchboxes

- There are normally two kinds of rectilinear regions.
  - Channels: routing regions having two parallel rows of fixed terminals.
  - Switchboxes: generalizations of channels that allow fixed terminals on all four sides of the region.

Channel



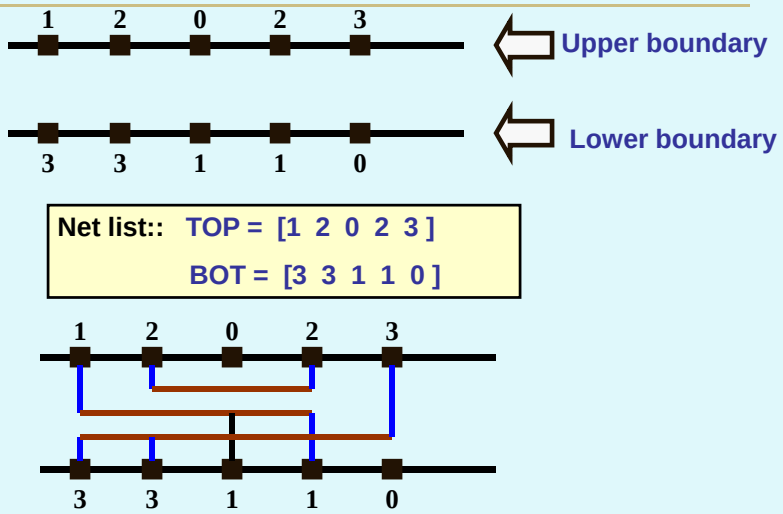


Switchbox

Channel Routing

- In channel routing, interconnections are made within a rectangular region having no obstructions.
  - A majority of modern-day ASIC's use channel routers.
  - Algorithms are efficient and simple.
  - Guarantees 100% completion if channel width is adjustable.
- Some terminologies:
  - Track: horizontal row available for routing.
  - Trunk: horizontal wire segment.
  - Branch: vertical wire segment connecting trunks to terminals.
  - Via: connection between a branch and a trunk.

Channel Routing Problem :: Terminologies



Problem Formulation

- The channel is defined by a rectangular region with two rows of terminals along its top and bottom sides.
  - Each terminal is assigned a number between 0 and N.
  - Terminals having the same label i belong to the same net i.
  - A ‘0’ indicates no connection.
- The netlist is usually represented by two vectors TOP and BOT.
  - TOP(k) and BOT(k) represents the labels on the grid points on the top and bottom sides of the channel in column k, respectively.

Contd.

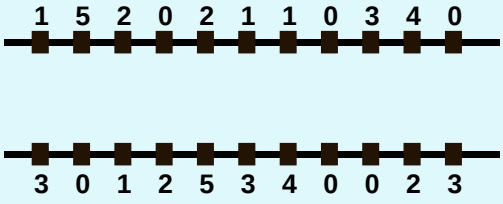
- The task of the channel router is to:
  - Assign horizontal segments of nets to tracks.
  - Assign vertical segments to connect
    - Horizontal segments of the same net in different tracks.
    - The terminals of the net to horizontal segments of the net.
- Channel height should be minimized.
- Horizontal and vertical constraints must not be violated.

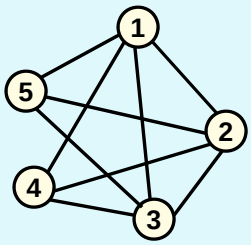
Contd.

- **Horizontal constraints between two nets:**
  - The horizontal span of two nets overlaps each other.
  - The nets must be assigned to separate tracks.
- **Vertical constraints between two nets:**
  - There exists a column such that the terminal i on top of the column belongs to one net, and the terminal j on bottom of the column belongs to the other net.
  - Net i must be assigned a track above that for net j.

Horizontal Constraint Graph (HCG)

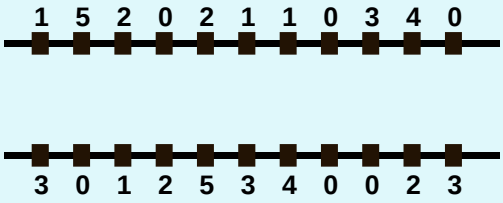
- **It is a graph where vertices represent nets, and edges represent horizontal constraints.**

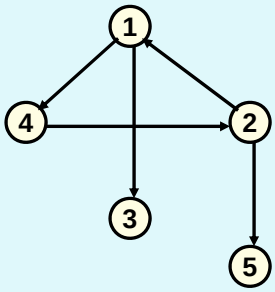




Vertical Constraint Graph (VCG)

- It is a directed graph where vertices represent nets, and edges represent vertical constraints.





Two-layer Channel Routing

- Left-Edge Algorithms (LEA)
  - Basic Left-Edge Algorithm
  - Left-Edge Algorithm with Vertical Constraints
  - Dogleg Router
- Constraint-Graph Based Algorithm
  - Net Merge Channel Router
  - Gridless Channel Router
- Greedy Channel Router
- Hierarchical Channel Router

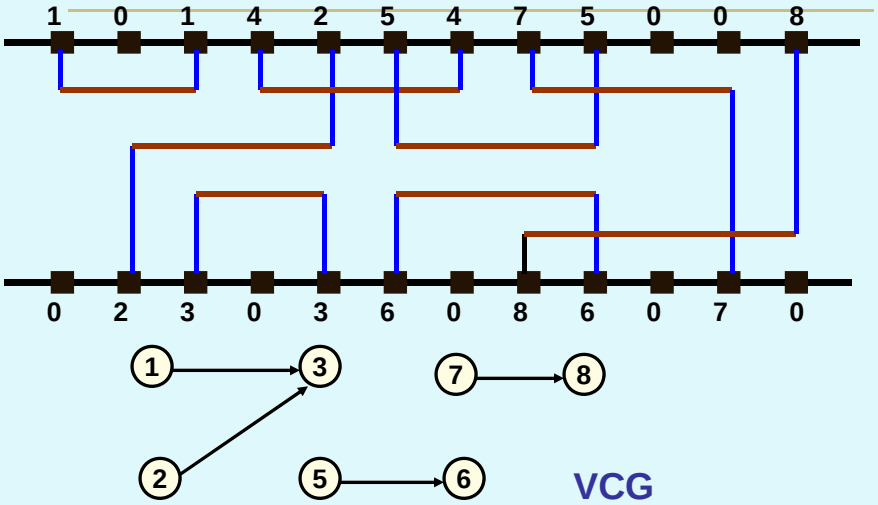
Basic Left Edge Algorithm

- **Assumptions:**
  - Only two-terminal nets.
  - No vertical constraints.
  - HV layer model.
  - Doglegs are not allowed.
- **Basic Steps:**
  - Sort the nets according to the x-coordinate of the leftmost terminal of the net.
  - Route the nets one-by-one according to the order.
  - For a net, scan the tracks from top to bottom, and assign it to the first track that can accommodate it.
- **In the absence of vertical constraints, the algorithm produces a minimum-track solution.**

Contd.

- **Extension to Left-Edge Algorithm**
  - Vertical constraints may exist, but there are no directed cycles in the VCG.
  - Select a net for routing if
    - The x-coordinate of the leftmost terminal is the least.
    - There is no edge incident on the vertex corresponding to that net in the VCG.
  - After routing a net, the corresponding vertex and the incident edges are deleted from the VCG.
  - Other considerations same as the basic left-edge algorithm.

Illustration



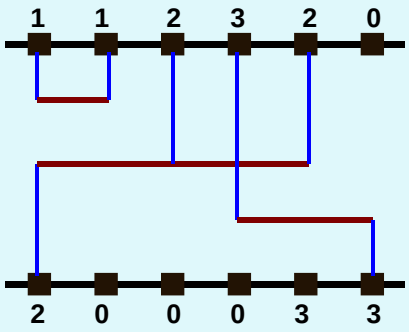
Dogleg Router

- Drawback of LEA
  - The entire net is on a single track.
  - Sometimes leads to routing with more tracks than necessary.
- Doglegs are used to place parts of the same net on different tracks.
  - A dogleg is a vertical segment that connects two trunks located in two different tracks.
  - May lead to a reduction in channel height.

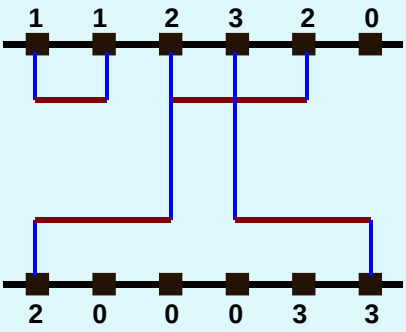
Contd.

- Dogleg router allows multi-terminal nets and vertical constraints.
  - Multi-terminal nets can be broken into a series of two-terminal nets.
- Cannot handle cyclic vertical constraints.

Example



No dogleg
3 tracks

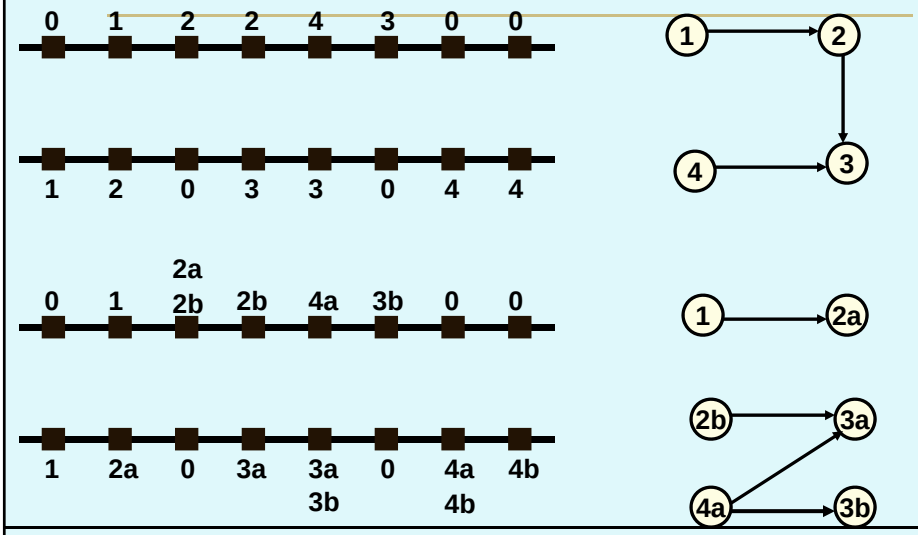


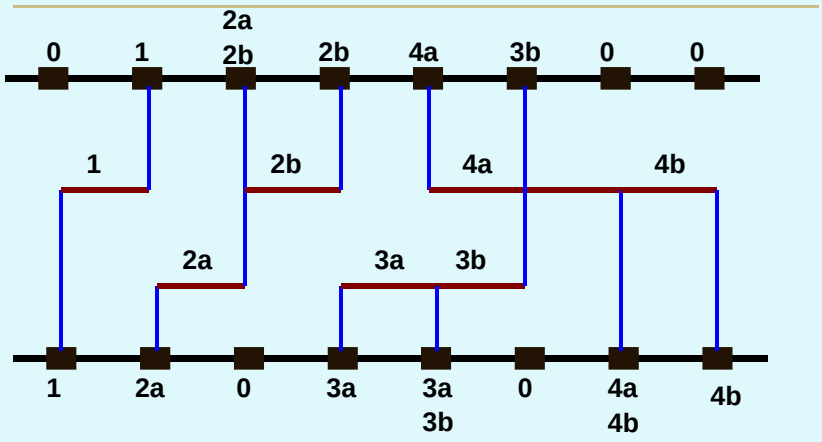
With dogleg
2 tracks

Dogleg Router: Algorithm

- Step 1:
  - If cycle exists in the VCG, return with failure.
- Step 2:
  - Split each multi-terminal net into a sequence of 2-terminal nets.
    - A net 2 .. 2 .. 2 will get broken as 2a .. 2a 2b .. 2b.
  - HCG and VCG gets modified accordingly.
- Step 3:
  - Apply the extended left-edge algorithm to the modified problem.

Illustration





Clock Routing

Problem Formulation

- Specialized algorithms are required for clock (and power nets) due to strict specifications for routing such nets.
  - Better to develop specialized routers for these nets.
  - Do not over-complicate the general router.
  - In many designs, both these nets are manually routed.
- Sophisticated and accurate clock routing tools are a must for high-performance designs.

Clock Routing

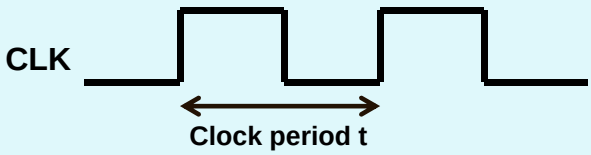
- Clock synchronization is one of the most critical considerations in designing high-performance VLSI circuits.
  - Data transfer between functional elements is synchronized by the clock.
  - It is desirable to design a circuit with the fastest possible clock.
- The clock signal is typically generated external to the chip.
  - Provided to the chip through “clock pin”.

Contd.

- Each functional unit which needs the clock is connected to clock pin by the clock net.
- Ideally, the clock must arrive at all the functional units precisely at the same time.
- In practice, clock skew exists.
  - Maximum difference in the arrival time of a clock at two different components.
  - Forces the designer to be conservative.
    - Use a large time period between clock pulses, i.e. lower clock frequency.

Clocking Schemes

- The clock is a simple pulsating signal alternating between 0 and 1.



- Digital systems use a number of clocking schemes:
  1. Single-phase clocking with latches
  2. Single-phase clocking with flip-flops
  3. Two-phase clocking

Clocking Schemes:: Contd.

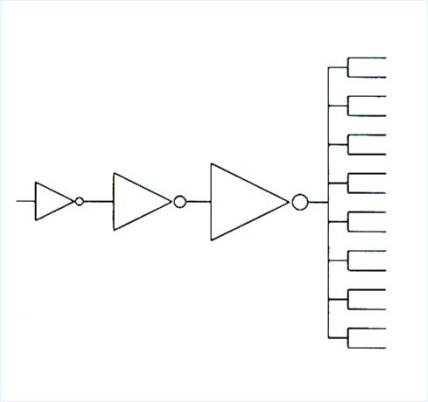
- As a rule of thumb, most systems cannot tolerate a clock skew of more than 10% of the system clock period.
  - A good clock distribution strategy is necessary.
  - Also a requirement for designing high-performance circuits.

Clock Buffering Mechanisms

- Clock signal is global in nature.
  - Clock lines are typically very long.
  - Long wires have large capacitances, which limit the performance of the system.
  - RC delay plays a big factor.
- RC delay cannot be reduced by making the wires wider.
  - Resistance reduces, but capacitance increases.
- To reduce RC delay, buffers are used.
  - Also helps to preserve the clock waveform.
  - Significantly reduces the delay.
  - May occupy as much as 5% of the total chip area.

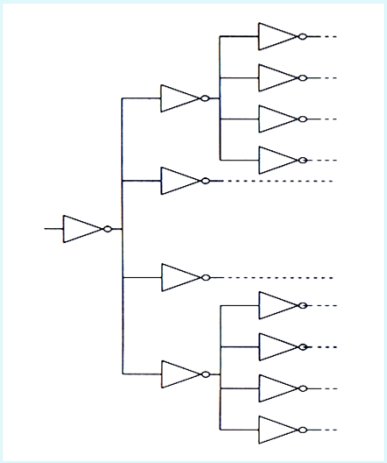
Clock Buffering:: Approach 1

- Use a big, centralized buffer.
  - Better from skew minimization point of view.



Clock Buffering:: Approach 2

- Distribute buffers in the branches of the clock tree.
  - Use identical buffers so that the delay introduced by the buffers is equal in all branches.
- Regular layout of the clock tree, and equalization of the buffer loads help to reduce clock skew.

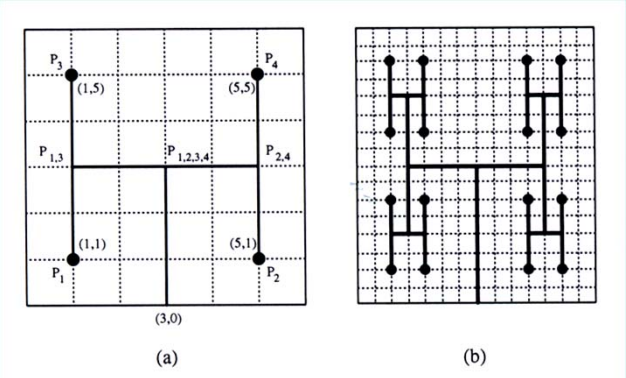


Clock Routing Algorithms

- How to minimize skew?
  - Distribute the clock signal in such a way that the interconnections carrying the clock signal to functional sub-blocks are equal in length.
- Several clock routing algorithms exist which try to achieve this goal.
  - H-tree based algorithm
  - X-tree based algorithm
  - MMM algorithm
  - Weighted center algorithm
  - Zero clock skew routing

H-tree based Algorithm

- Consider that all clock terminals are arranged in a symmetrical fashion, as in the case of gate arrays.



Contd.

- In (a), all points are exactly 7 units from the point P_0, and hence the skew is zero.
- This ensures minimum-delay routing as well.
  - P_0 and P_3 are at a distance 7 (rectilinear distance).
- Can be generalized to n points, where n is a power of 4.

Power and Ground Routing

Basic Problem

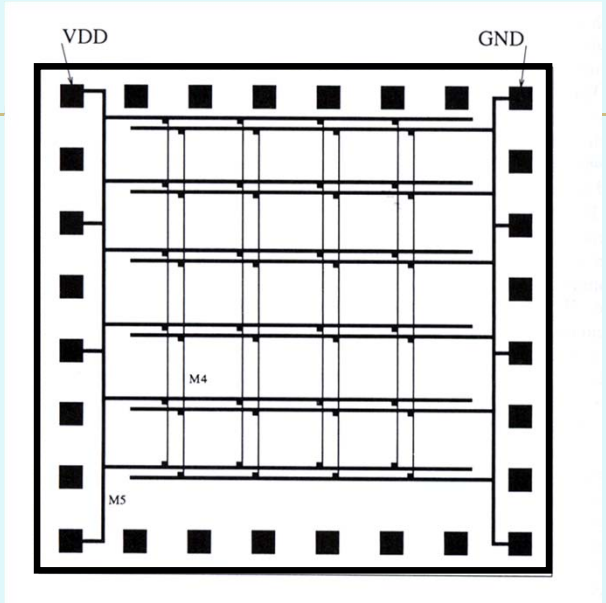
- In a design, almost all blocks require power and ground connections.
- Power and ground nets are usually laid out entirely on the metal layer(s) of the chip.
  - Due to smaller resistivity of metal.
  - Planar single-layer implementation is desirable since contacts (via's) also significantly add to the parasitics.
- Routing of power (VDD) and ground (GND) nets consists of two main tasks:
  - Construction of interconnection topology.
  - Determination of the widths of the various segments.

Contd.

- **Requirement:**
  - Find two non-intersecting interconnection trees.
  - The width of the trees at any particular point must be proportional to the amount of current being drawn by the points in that sub-tree.

Using Grid Structure

- Several rows of horizontal wires for both VDD and GND run parallel to each other on one metal layer.
- The vertical wires run in another metal layer and connect the horizontal wires.
- A block simply connects to the nearest VDD and GND wire.


VDD
GND
M4
M5