# Physical Design (Layout)

**Nov. 18, 2015**

Course Webpage:

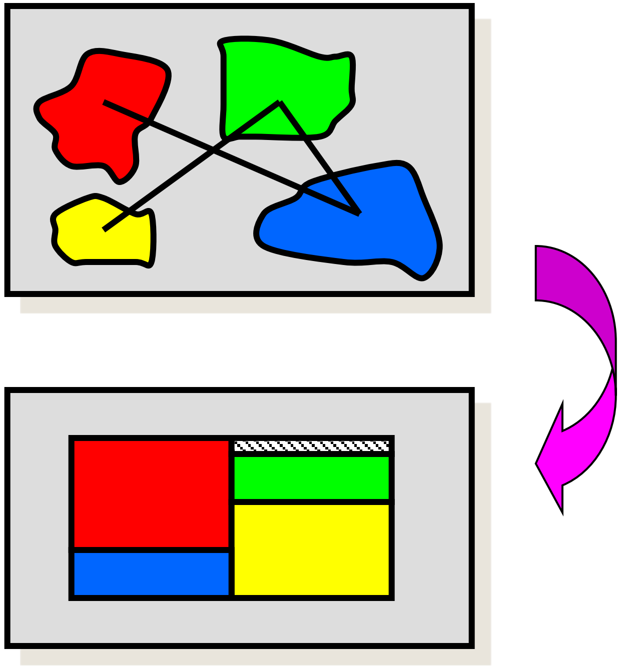
http://staff.ustc.edu.cn/~songch/da-ug.htm

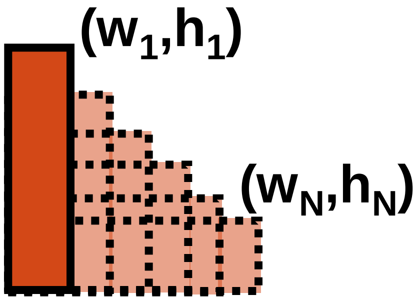
# Outline

- Floorplanning
- Placement
  - Global Placement
  - Detailed Placement
- Routing
  - Global Routing
  - Detailed Routing
  - Clock Routing

# Floorplanning

- Problem
  - Given circuit modules (or cells) and their connections, determine the *approximate* location of circuit elements
  - Consistent with a hierarchical / building block design/IP-based methodology
  - Modules (result of partitioning, IP blocks):
    - Fixed area, generally rectangular
    - Fixed aspect ratio → hard macro (aka fixed-shaped blocks)
      fixed / floating terminals (pins)
      Rotation might be allowed / denied
    - Flexible shape → soft macro (aka soft modules)



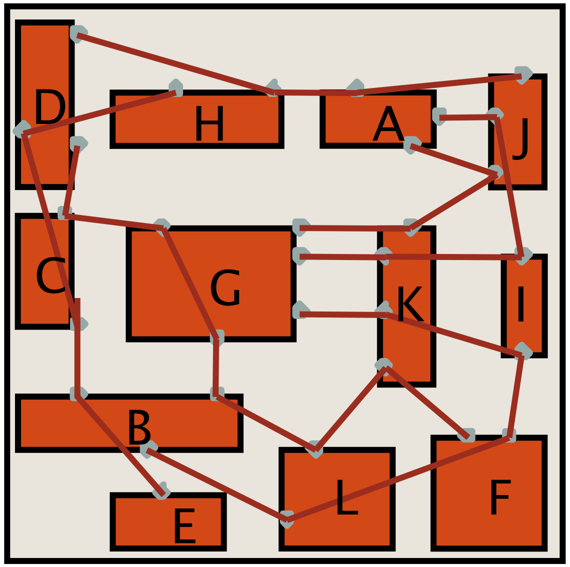


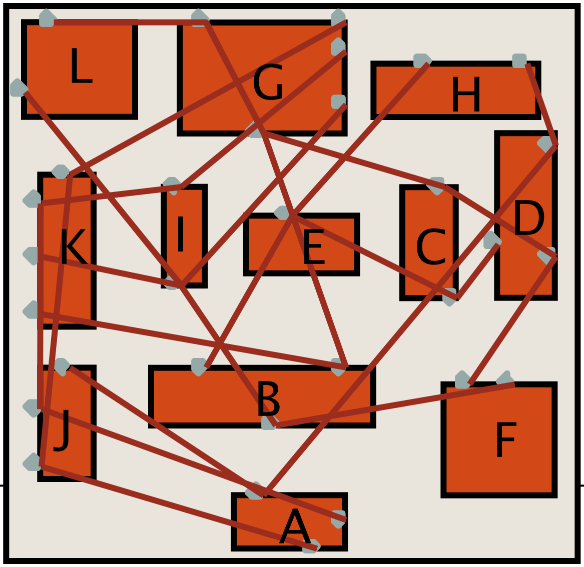
# Floorplanning (cont.)

- Objectives:
  - Minimize area
  - Determine best shape of soft modules
  - Minimize total wire length
    - to make subsequent routing phase easy
      (short wire length roughly translates into routability)
  - Additional cost components:
    - Wire congestion (exact routability measure), Wire delays, Power consumption, System throughput (e.g., CPI of a processor)
- Possible additional constraints:
  - Fixed location for some modules
  - Fixed die, or range of die aspect ratio

# Floorplanning: Why Important?

- Early stage of physical design
  - Determines the location of large blocks
    ➔ detailed placement easier (divide and conquer!)
  - Estimates of area, delay, power
    ➔ important design decisions
  - Impact on subsequent design steps (e.g., routing, heat dissipation analysis and optimization)





# Sequence Pair (SP)

A floorplan is represented by a pair of permutations of the module names:

e.g. 1 3 2 4 5

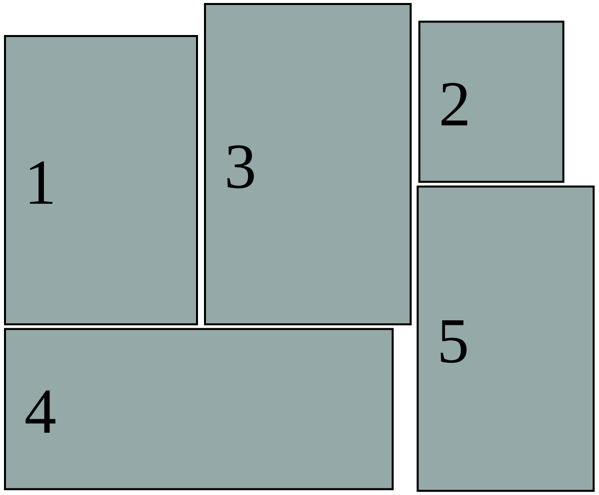
3 5 4 1 2

A sequence pair $(s_1, s_2)$ of $n$ modules can represent all possible floorplans formed by the $n$ modules by specifying the pair-wise relationship between the modules.

# Example

Consider the sequence pair:

(13245,41352 )



Any other SP that is also valid for this packing?

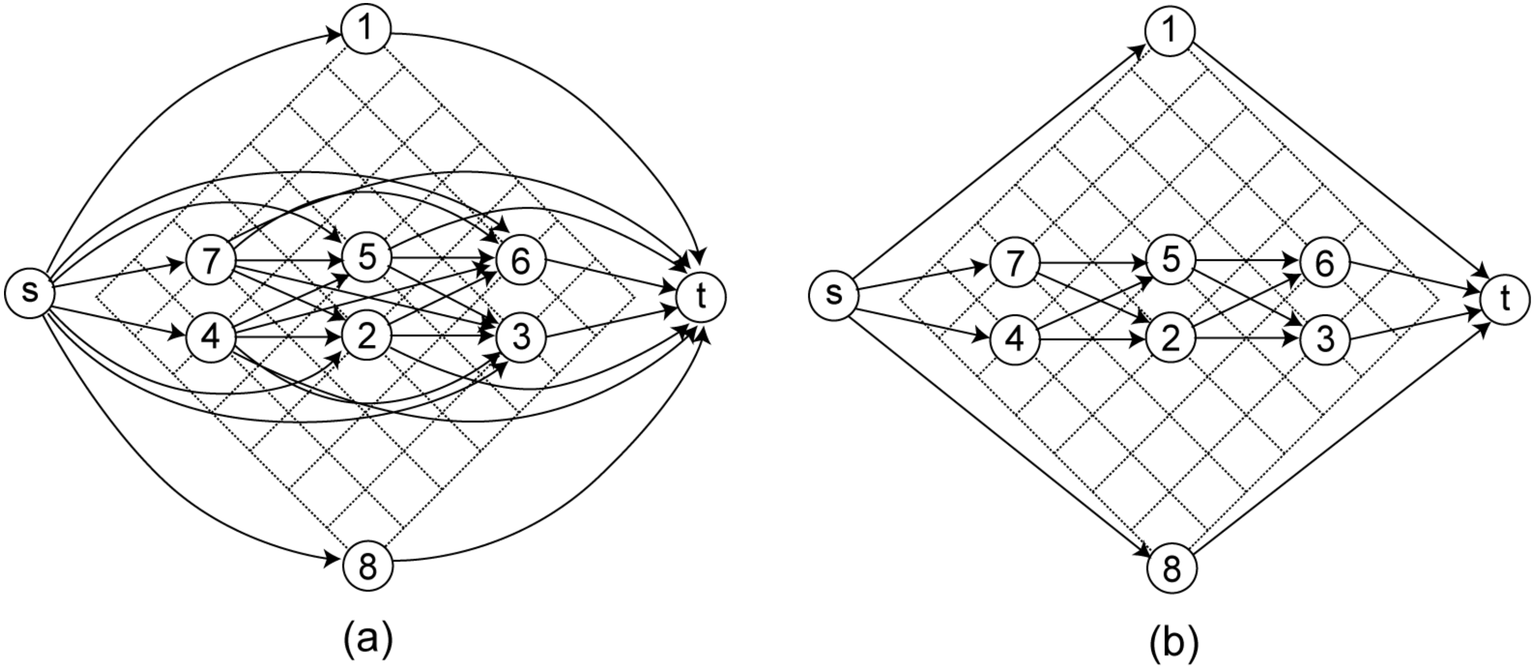
# Sequence Pair To a Floorplan

- Initial SP: $SP_1 = (17452638, 84725361)$
  - Dimensions: (2,4), (1,3), (3,3), (3,5), (3,2), (5,3), (1,2), (2,4)
  - Based on $SP_1$ we build the following table:

| module | right-of | left-of | above | below |
|---|---|---|---|---|
| 1 | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\{2, 3, 4, 5, 6, 7, 8\}$ |
| 2 | $\{3, 6\}$ | $\{4, 7\}$ | $\{1, 5\}$ | $\{8\}$ |
| 3 | $\emptyset$ | $\{2, 4, 5, 7\}$ | $\{1, 6\}$ | $\{8\}$ |
| 4 | $\{2, 3, 5, 6\}$ | $\emptyset$ | $\{1, 7\}$ | $\{8\}$ |
| 5 | $\{3, 6\}$ | $\{4, 7\}$ | $\{1\}$ | $\{2, 8\}$ |
| 6 | $\emptyset$ | $\{2, 4, 5, 7\}$ | $\{1\}$ | $\{3, 8\}$ |
| 7 | $\{2, 3, 5, 6\}$ | $\emptyset$ | $\{1\}$ | $\{4, 8\}$ |
| 8 | $\emptyset$ | $\emptyset$ | $\{1, 2, 3, 4, 5, 6, 7\}$ | $\emptyset$ |

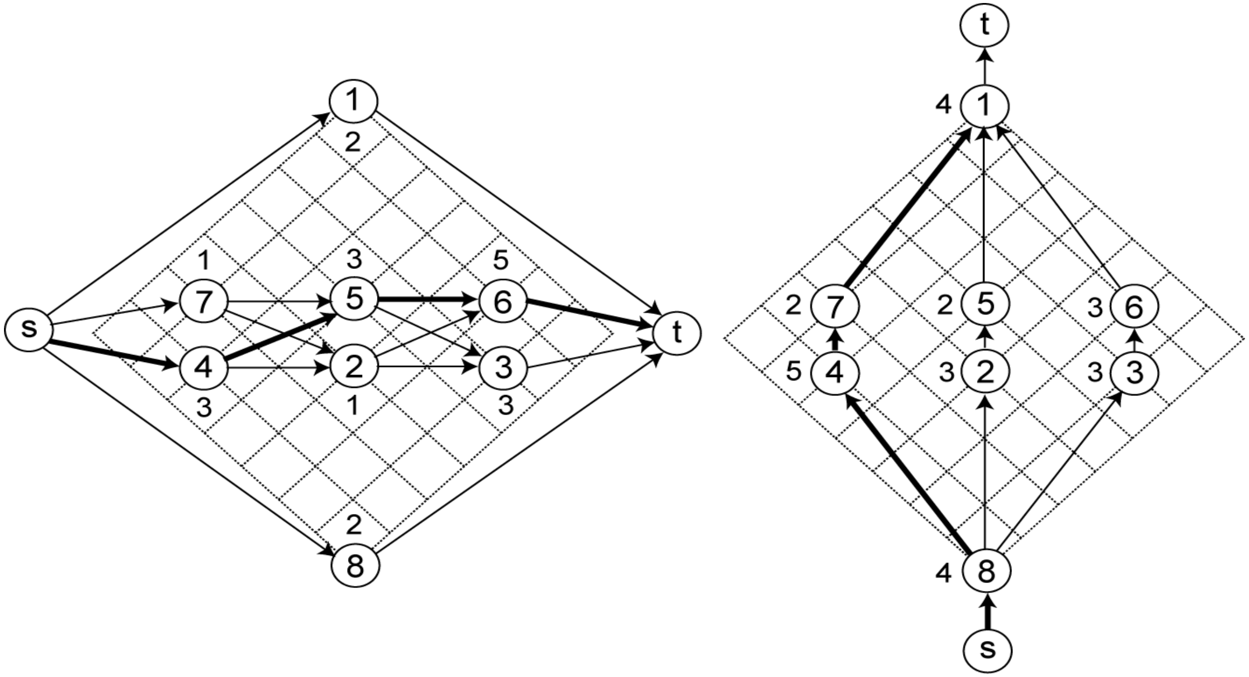
# Constraint Graphs

- Horizontal constraint graph (HCG)
  - Before and after removing transitive edges



# Computing Chip Width and Height

- Longest source-sink path length in:
  - HCG = chip width, (Vertical) VCG = chip height
  - Node weight = module width/height



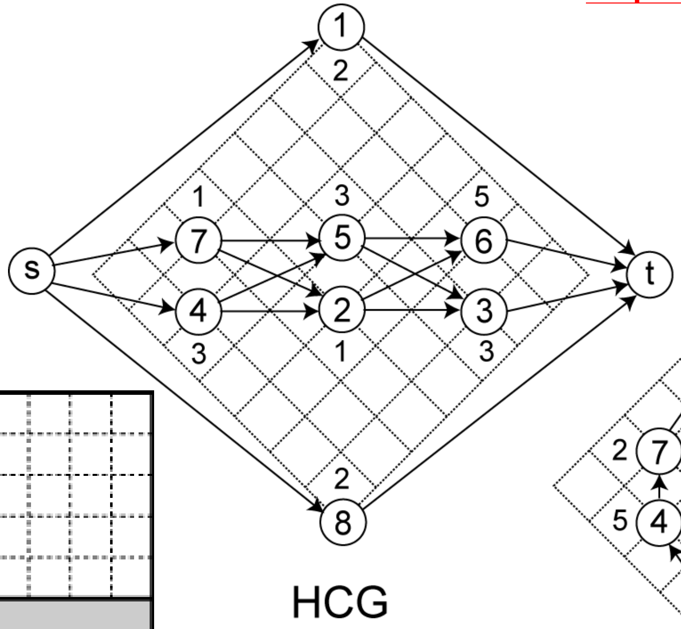
(a)

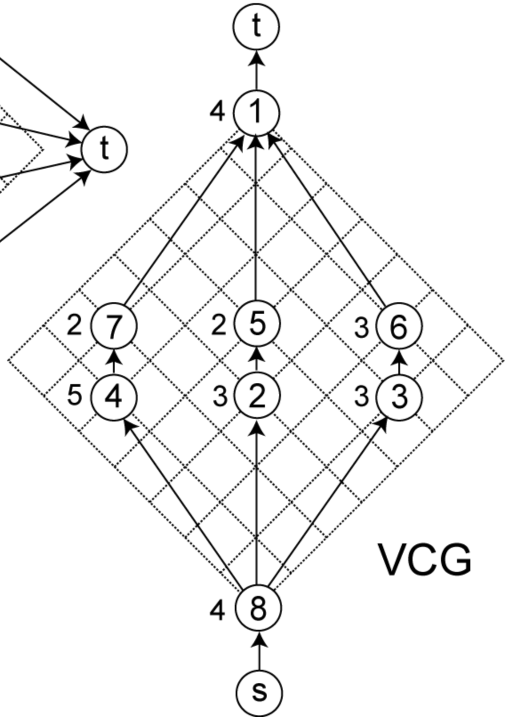
(b)

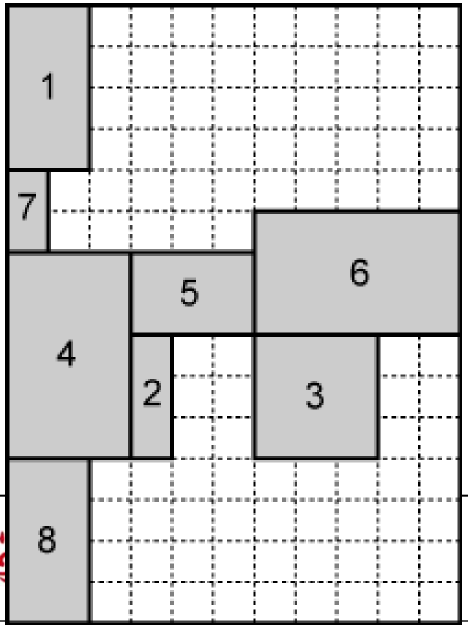
# Computing Module Location

- Use longest source-module path length in HCG/VCG
  - Lower-left corner location = source to module input path length

| module | HCV | VCG |
|---|---|---|
| 1 | 0 | 11 |
| 2 | 3 | 4 |
| 3 | 6 | 4 |
| 4 | 0 | 4 |
| 5 | 3 | 7 |
| 6 | 6 | 7 |
| 7 | 0 | 9 |
| 8 | 0 | 0 |



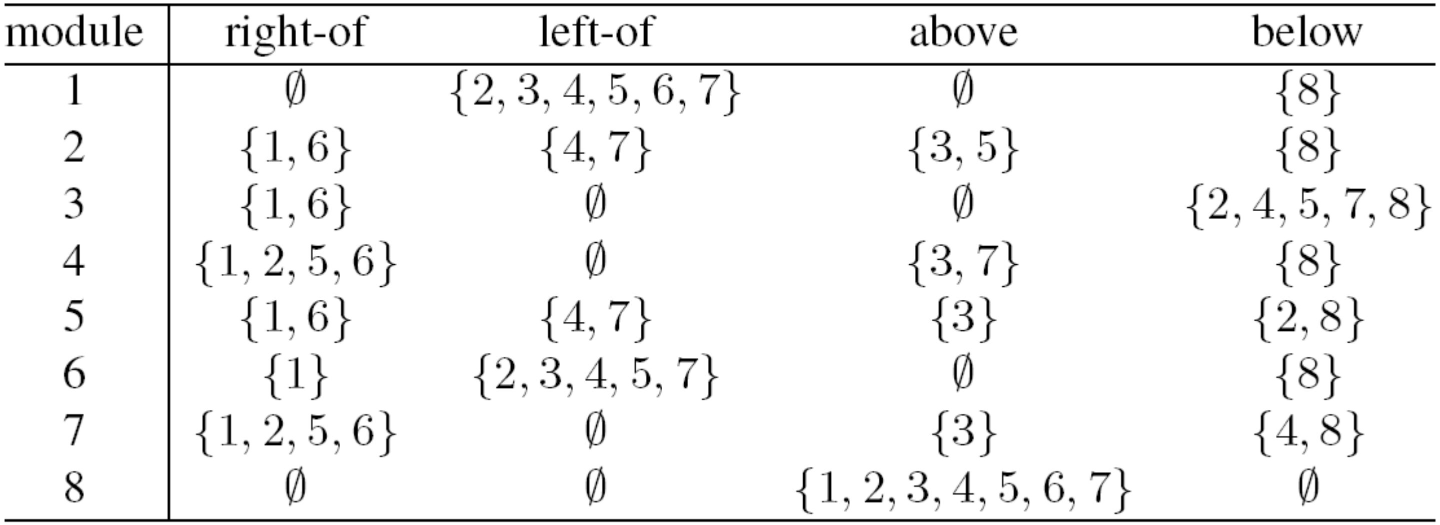




Dim: 11 × 15

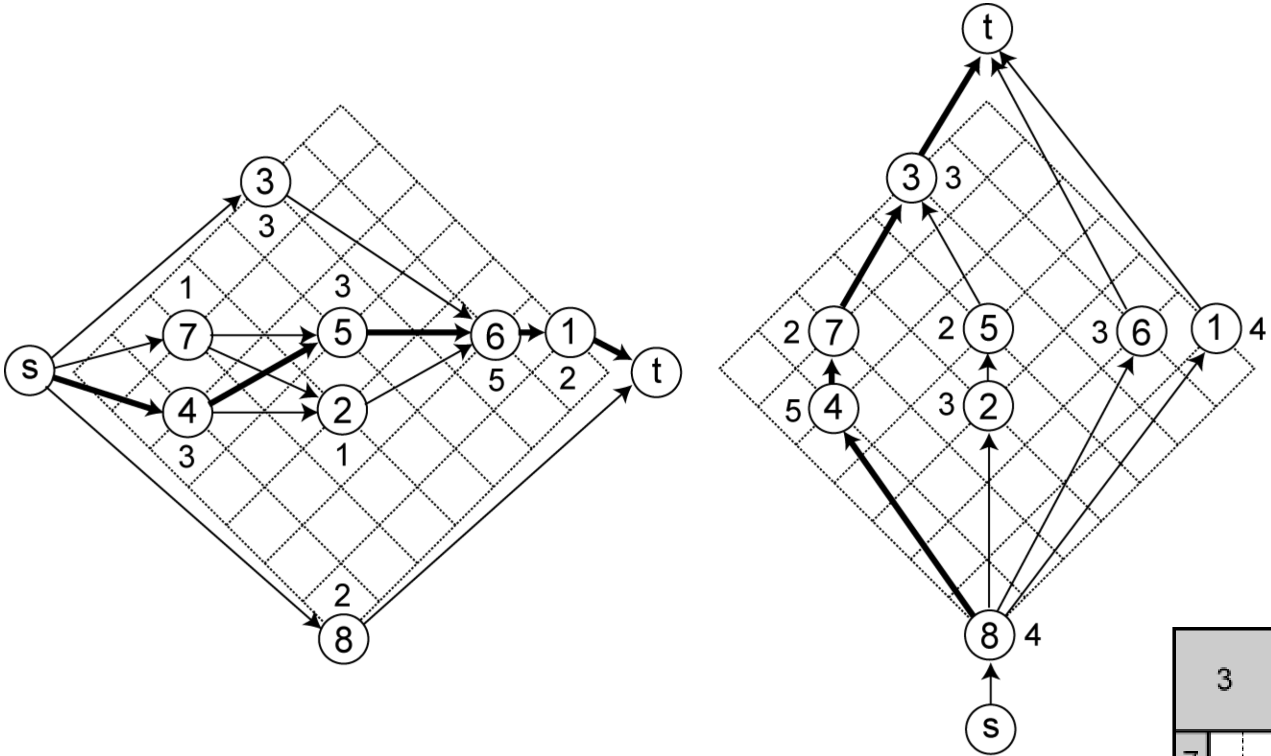
# Move I (purturbation)

- Swap 1 and 3 in positive sequence of $SP_1$
  - $SP_1$ = (17452638, 84725361)
  - $SP_2$ = (37452618, 84725361)



| module | right-of | left-of | above | below |
|---|---|---|---|---|
| 1 | $\emptyset$ | $\{2,3,4,5,6,7\}$ | $\emptyset$ | $\{8\}$ |
| 2 | $\{1,6\}$ | $\{4,7\}$ | $\{3,5\}$ | $\{8\}$ |
| 3 | $\{1,6\}$ | $\emptyset$ | $\emptyset$ | $\{2,4,5,7,8\}$ |
| 4 | $\{1,2,5,6\}$ | $\emptyset$ | $\{3,7\}$ | $\{8\}$ |
| 5 | $\{1,6\}$ | $\{4,7\}$ | $\{3\}$ | $\{2,8\}$ |
| 6 | $\{1\}$ | $\{2,3,4,5,7\}$ | $\emptyset$ | $\{8\}$ |
| 7 | $\{1,2,5,6\}$ | $\emptyset$ | $\{3\}$ | $\{4,8\}$ |
| 8 | $\emptyset$ | $\emptyset$ | $\{1,2,3,4,5,6,7\}$ | $\emptyset$ |

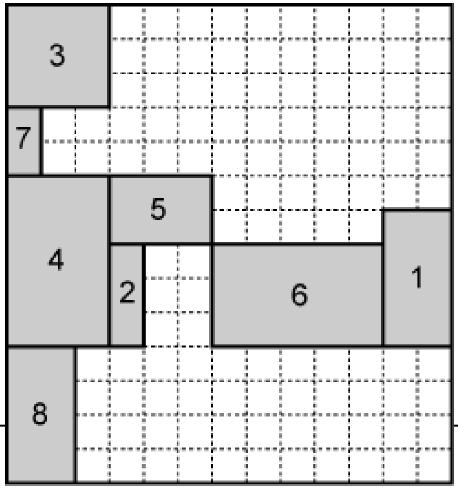
# Constraint Graphs and floorplan



(a)

(b)

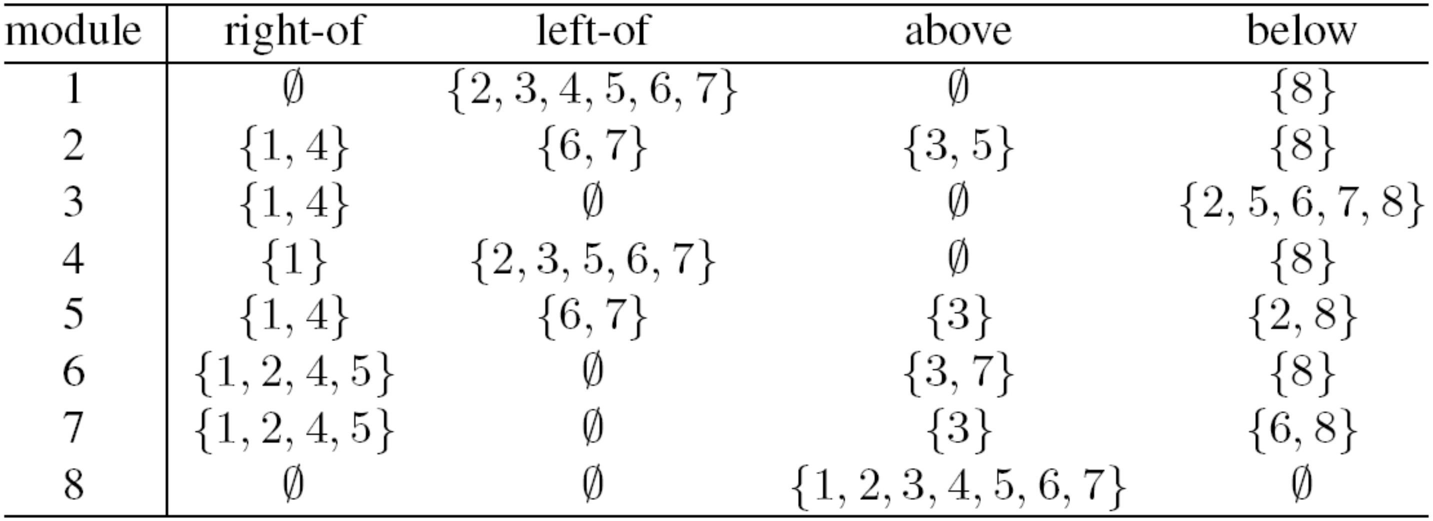
| module | HCV | VCG |
|---|---|---|
| 1 | 11 | 4 |
| 2 | 3 | 4 |
| 3 | 0 | 11 |
| 4 | 0 | 4 |
| 5 | 3 | 7 |
| 6 | 6 | 4 |
| 7 | 0 | 9 |
| 8 | 0 | 0 |



Dim: 13 × 14

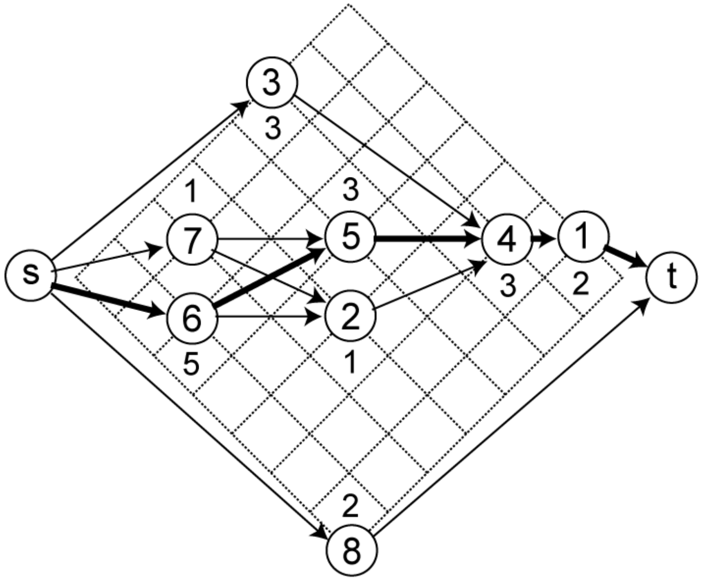
# Move II

- Swap 4 and 6 in both sequences of $SP_2$
  - $SP_2 = (37\underline{4}52\underline{6}18, 8\underline{4}7253\underline{6}1)$
  - $SP_3 = (37\underline{6}52\underline{4}18, 8\underline{6}7253\underline{4}1)$

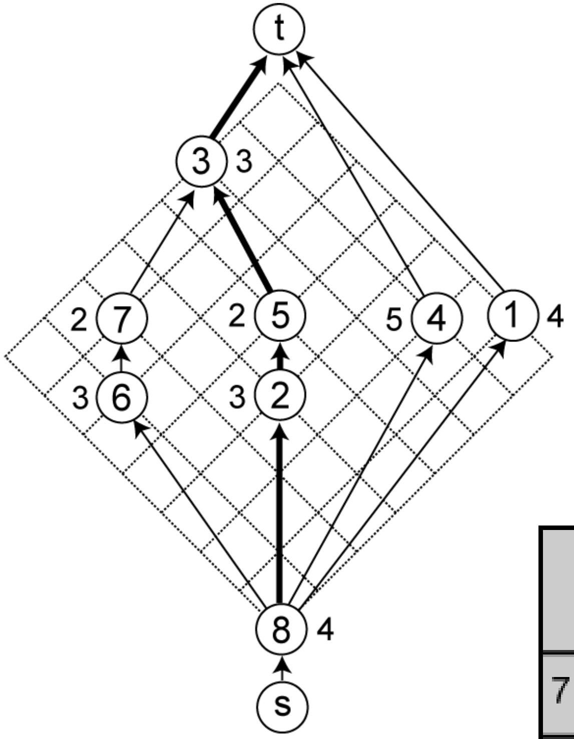


| module | right-of | left-of | above | below |
|---|---|---|---|---|
| 1 | $\emptyset$ | $\{2,3,4,5,6,7\}$ | $\emptyset$ | $\{8\}$ |
| 2 | $\{1,4\}$ | $\{6,7\}$ | $\{3,5\}$ | $\{8\}$ |
| 3 | $\{1,4\}$ | $\emptyset$ | $\emptyset$ | $\{2,5,6,7,8\}$ |
| 4 | $\{1\}$ | $\{2,3,5,6,7\}$ | $\emptyset$ | $\{8\}$ |
| 5 | $\{1,4\}$ | $\{6,7\}$ | $\{3\}$ | $\{2,8\}$ |
| 6 | $\{1,2,4,5\}$ | $\emptyset$ | $\{3,7\}$ | $\{8\}$ |
| 7 | $\{1,2,4,5\}$ | $\emptyset$ | $\{3\}$ | $\{6,8\}$ |
| 8 | $\emptyset$ | $\emptyset$ | $\{1,2,3,4,5,6,7\}$ | $\emptyset$ |

# Constraint Graphs and floorplan

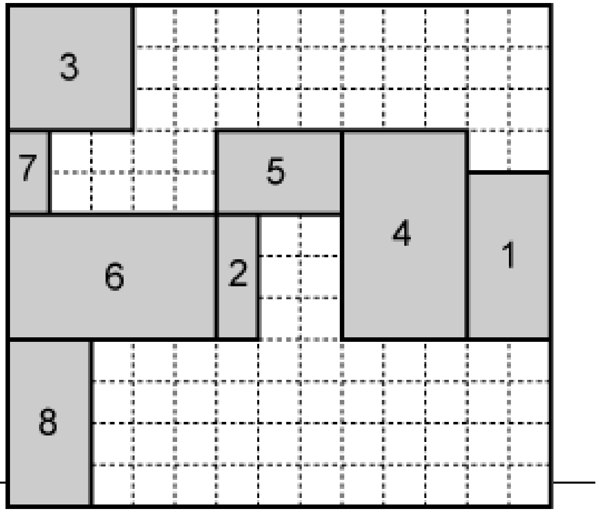


(a)



(b)

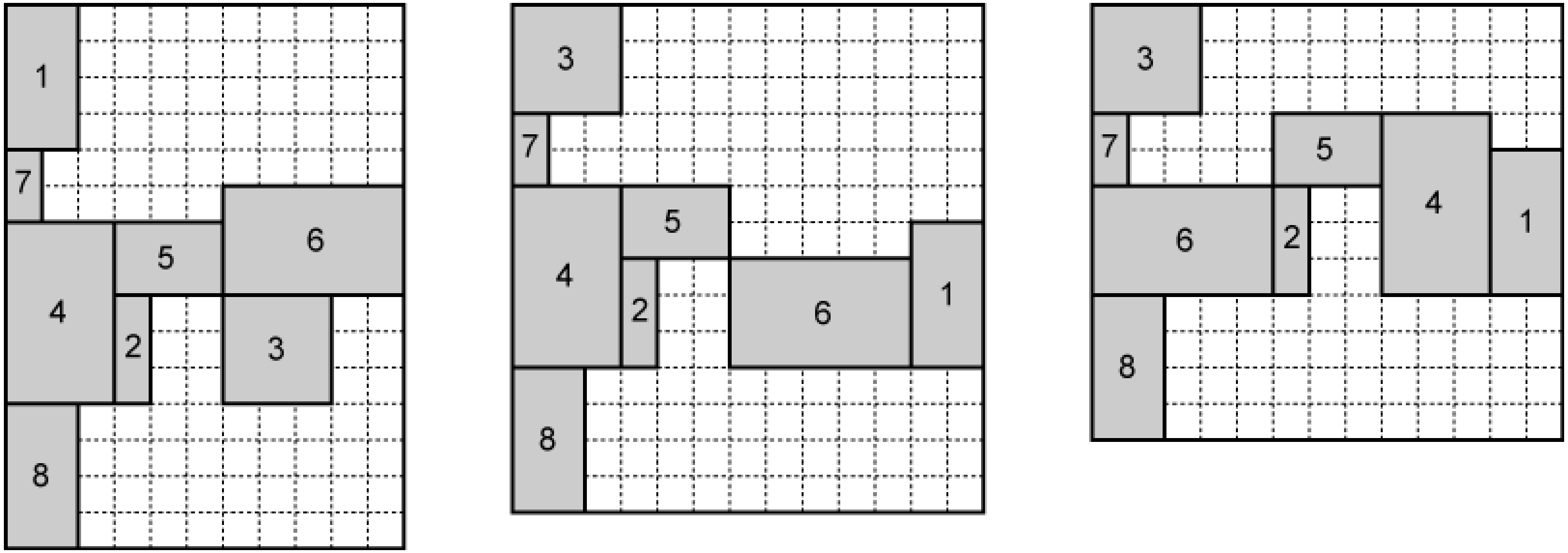
| module | HCV | VCG |
|---|---|---|
| 1 | 11 | 4 |
| 2 | 3 | 4 |
| 3 | 0 | 11 |
| 4 | 0 | 4 |
| 5 | 3 | 7 |
| 6 | 6 | 4 |
| 7 | 0 | 9 |
| 8 | 0 | 0 |



Dim: 13 × 12

# Summary (SP)

- Impact of the moves:
  - Floorplan dimension changes from 11 × 15 to 13 × 14 to 13 × 12

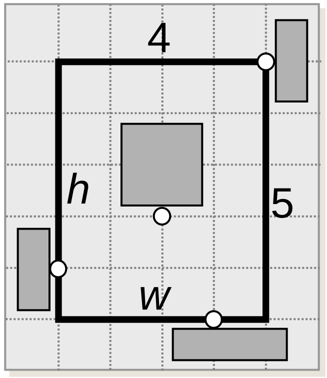


# Placement

- Problem
  - Given a netlist, and fixed-shape cells (small, standard cell), find the exact location of the cells to minimize area and wire-length
  - Consistent with the standard-cell design methodology
    - Row-based, no hard-macros
  - Modules:
    - Usually fixed, equal height
    - Some fixed (I/O pads)
    - Connected by edges or hyperedges
- Objectives
  - Cost components: area, wire length
    - Additional cost components: timing, congestion

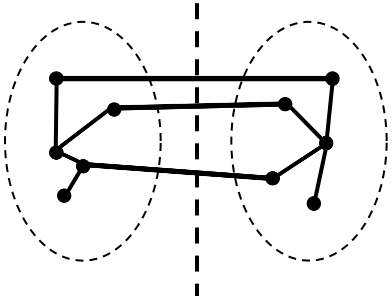
# Optimization Objectives

Total Wirelength

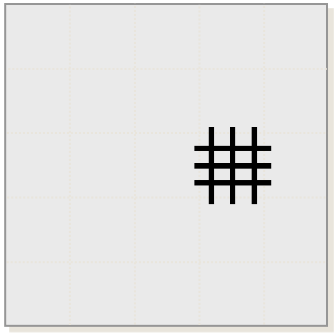


HPWL = 9

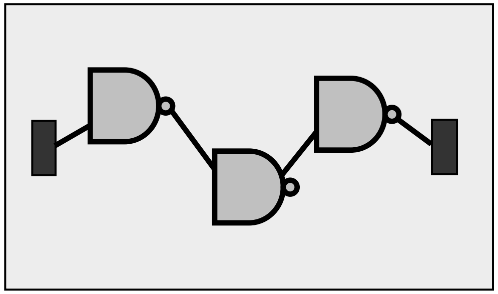
Number of Cut Nets



Wire Congestion



Signal Delay



# Placement Cost Components

- Area
  - Would like to pack all the modules very tightly
- Wire length (half-perimeter of the net bounding box)
  - Minimize average wire length
  - Would result in tight packing of modules with high connectivity
- Overlap
  - Could be prohibited by the moves, or used as penalty
  - Keep the cells from overlapping (moves cells apart)
- Timing
  - Not a 1-1 correspondence with wire length minimization, but consistent on average
- Congestion
  - Measure of routability
  - Tends to move cells apart

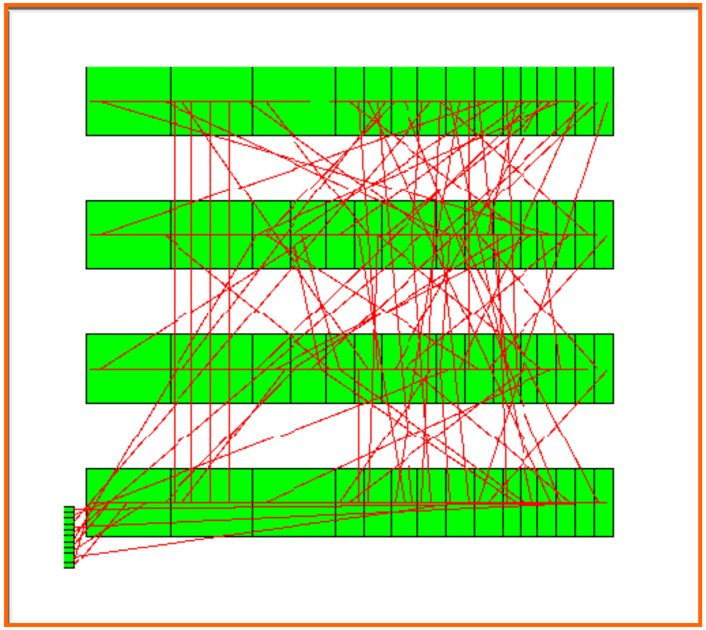
# Importance of Placement

- Placement: fundamental problem in physical design
  - Serious interconnect issues (delay, routability, noise) in nanometer design
    - Placement determines interconnect to the first order
    - Need placement information even in early design stages (e.g., logic synthesis)
    - Need to have a good placement solution
  - Placement problem becomes significantly larger
- Glue of the physical synthesis

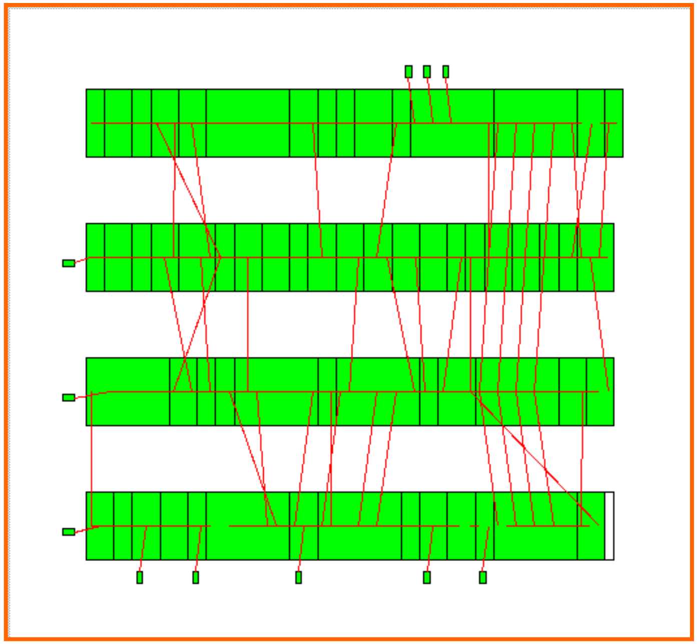
# Design Types

- ASICs
  - Lots of fixed I/Os, few macros, millions of standard cells
  - Placement densities : 40-80% (IBM)
  - Flat and hierarchical designs
- SoCs
  - Many more macro blocks, cores
  - Datapaths + control logic
  - Can have very low placement densities : < 20%
- Micro-Processor (μP) Random Logic Macros(RLM)
  - Hierarchical partitions are placement instances (5-30K)
  - High placement densities : 80%-98% (low whitespace)
  - Many fixed I/Os, relatively few standard cells

# VLSI Global Placement Examples



**bad placement**



**good placement**

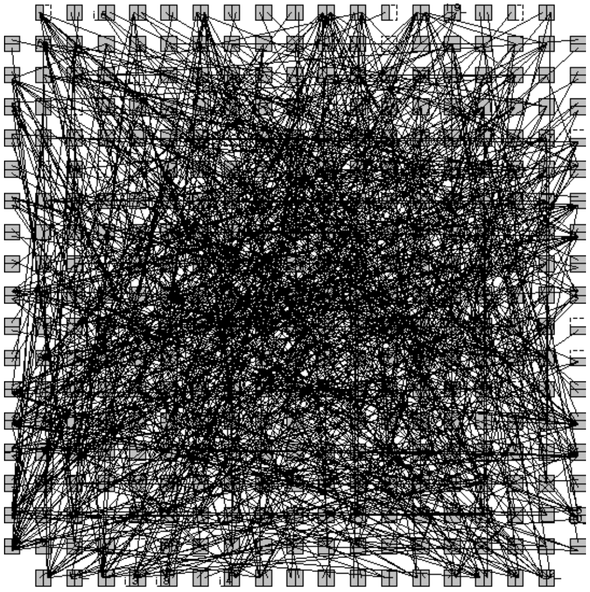
# Placement can Make A Difference

- MCNC Benchmark circuit e64 (contains 230 4-LUT). Placed to a FPGA.

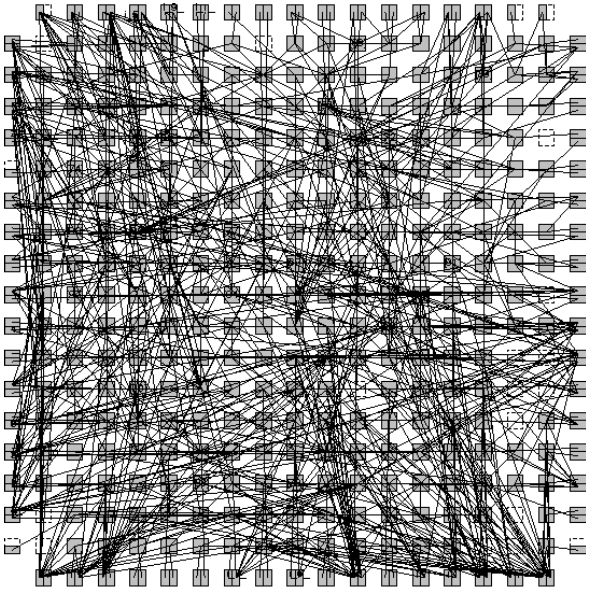
Random Initial Placement

Final Placement

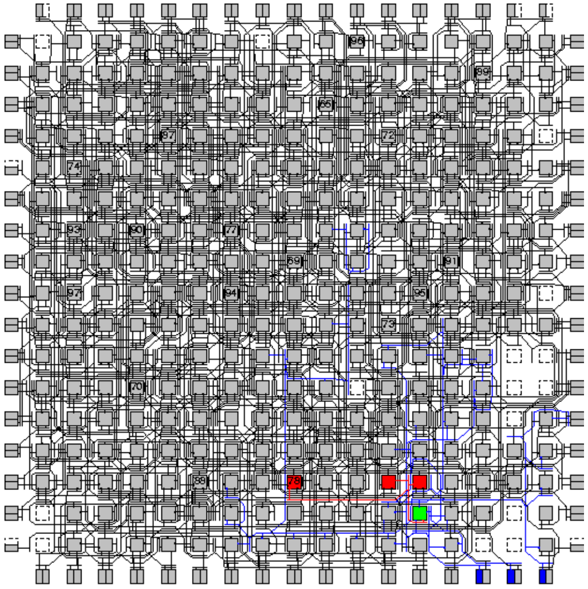
After Detailed Routing



Initial Placement. Cost: 74.5562. Channel Factor: 100

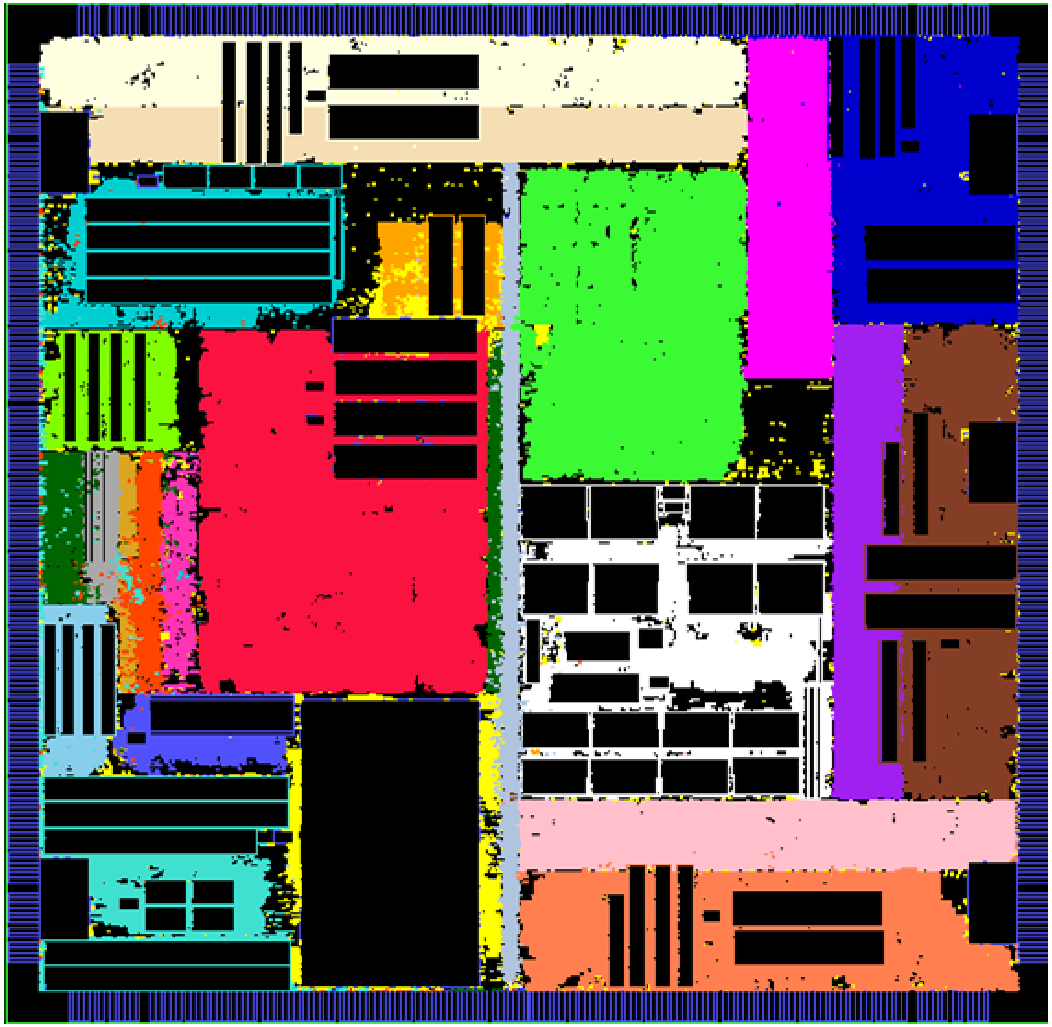


Final Placement. Cost: 28.5384. Channel Factor: 100

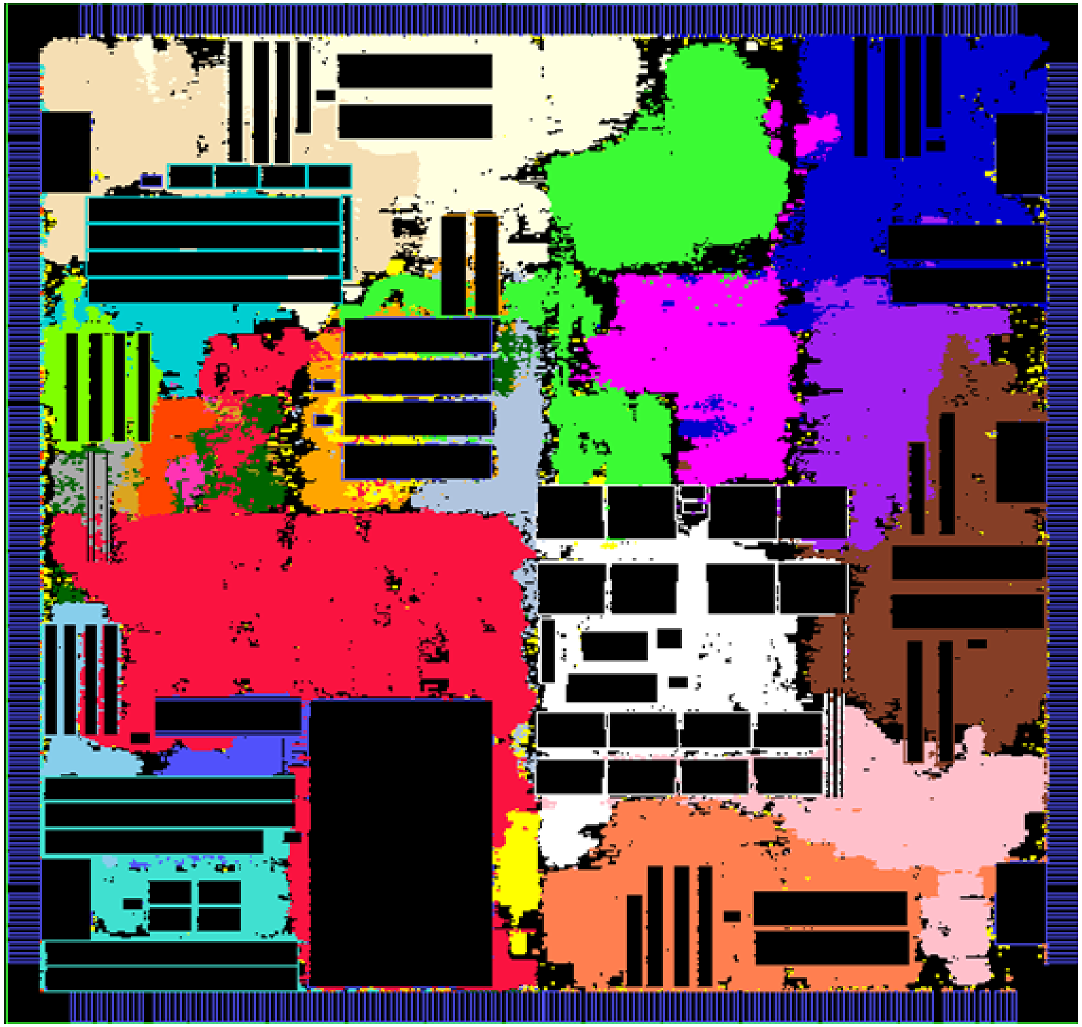


Routing succeeded with a channel width factor of 7.

# Floorplaned Placement



# Unconstrained Placement

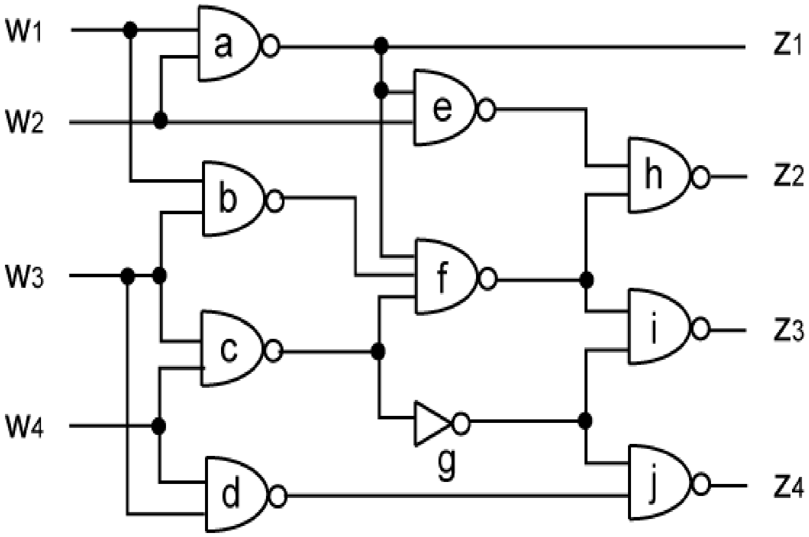


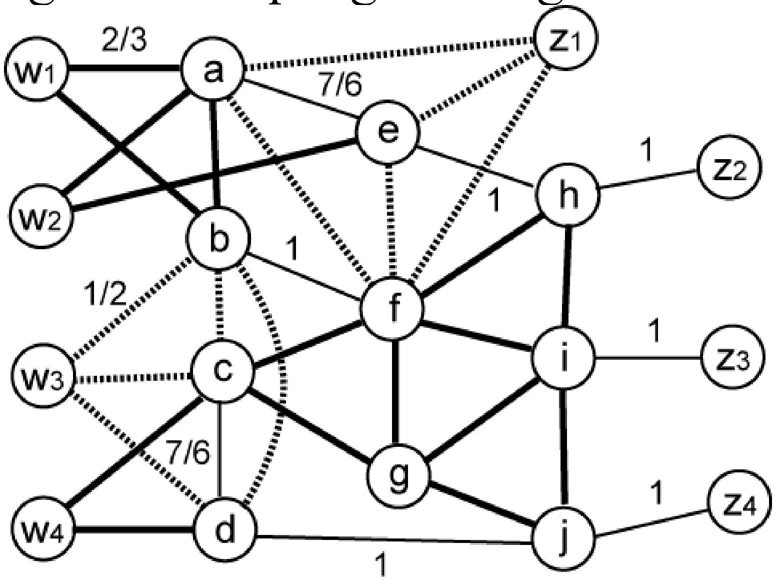
# GORDIAN Placement

- Minimize the squared wire-length among the cells

$$\phi(x) = \frac{1}{2}x^T C x + d_x{}^T x \quad \phi(y) = \frac{1}{2}y^T C y + d_y{}^T y$$

- Perform GORDIAN placement
  - Uniform area and net weight, area balance factor = 0.5
  - Undirected graph model: each edge in *k*-clique gets weight 2/*k*



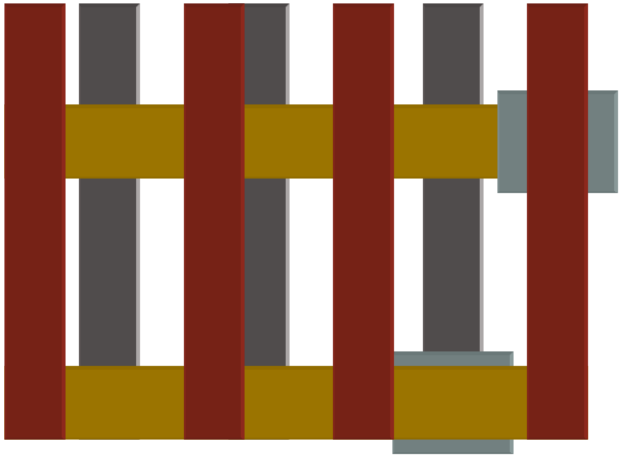


# Routing

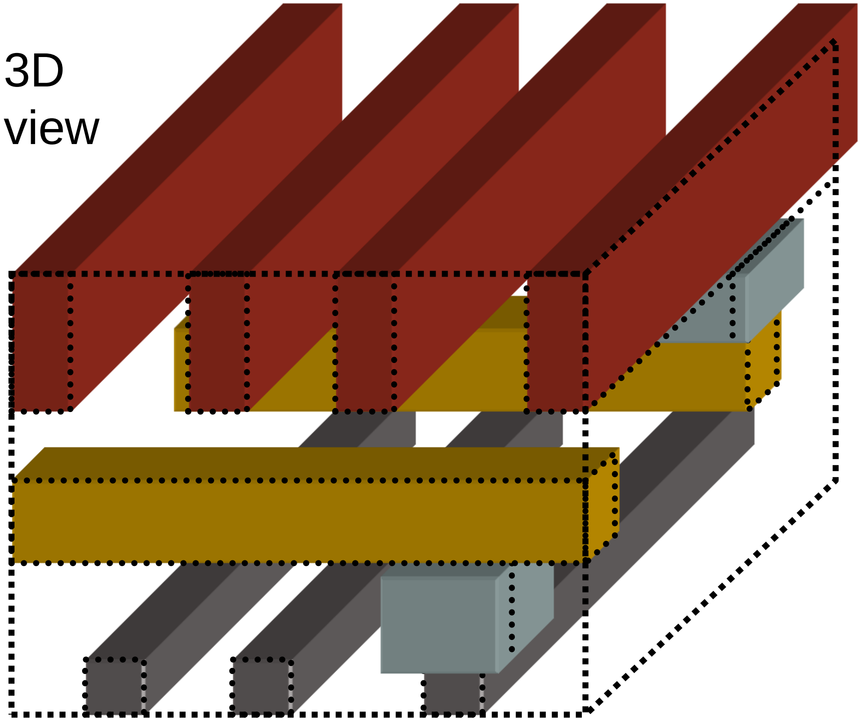
- Problem
  - Given a placement, and a fixed number of metal layers, find a valid pattern of horizontal and vertical wires that connect the terminals of the nets
  - Levels of abstraction:
    - Global routing
    - Detailed routing
- Objectives
  - Cost components:
    - Area (channel width) – min congestion in prev levels helped
    - Wire delays – timing minimization in previous levels
    - Number of layers (fewer layers → less expensive)
    - Additional cost components: number of bends, vias

# Routing Anatomy

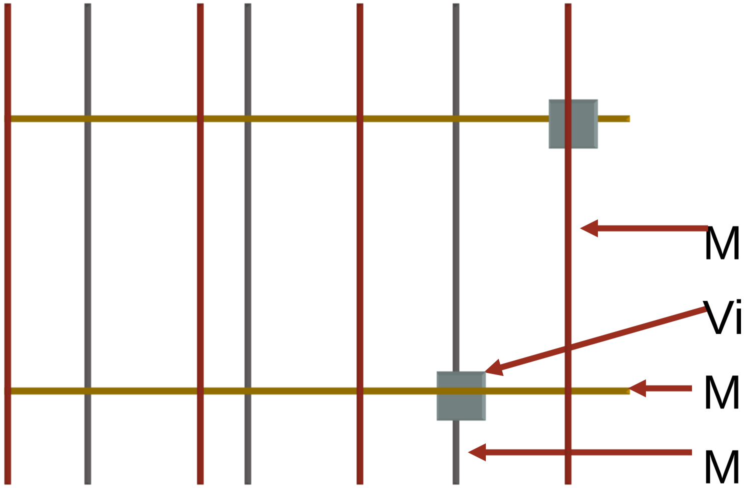
Top view



3D view

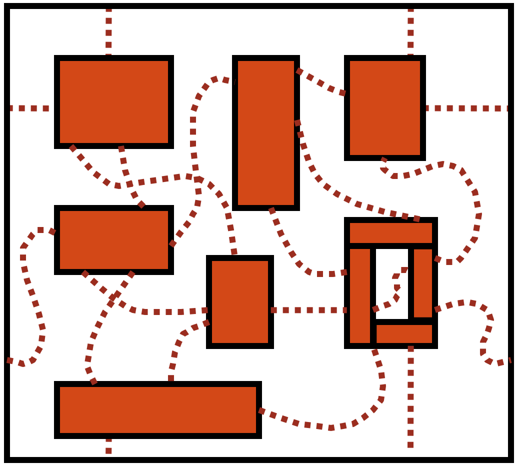


Symbolic Layout

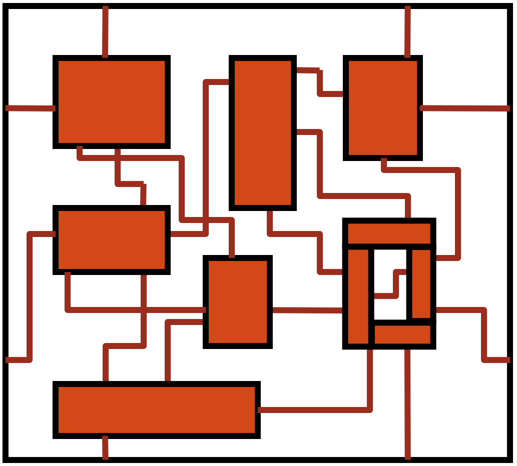


Note: Colors used in this slide are not standard

# Global vs. Detailed Routing

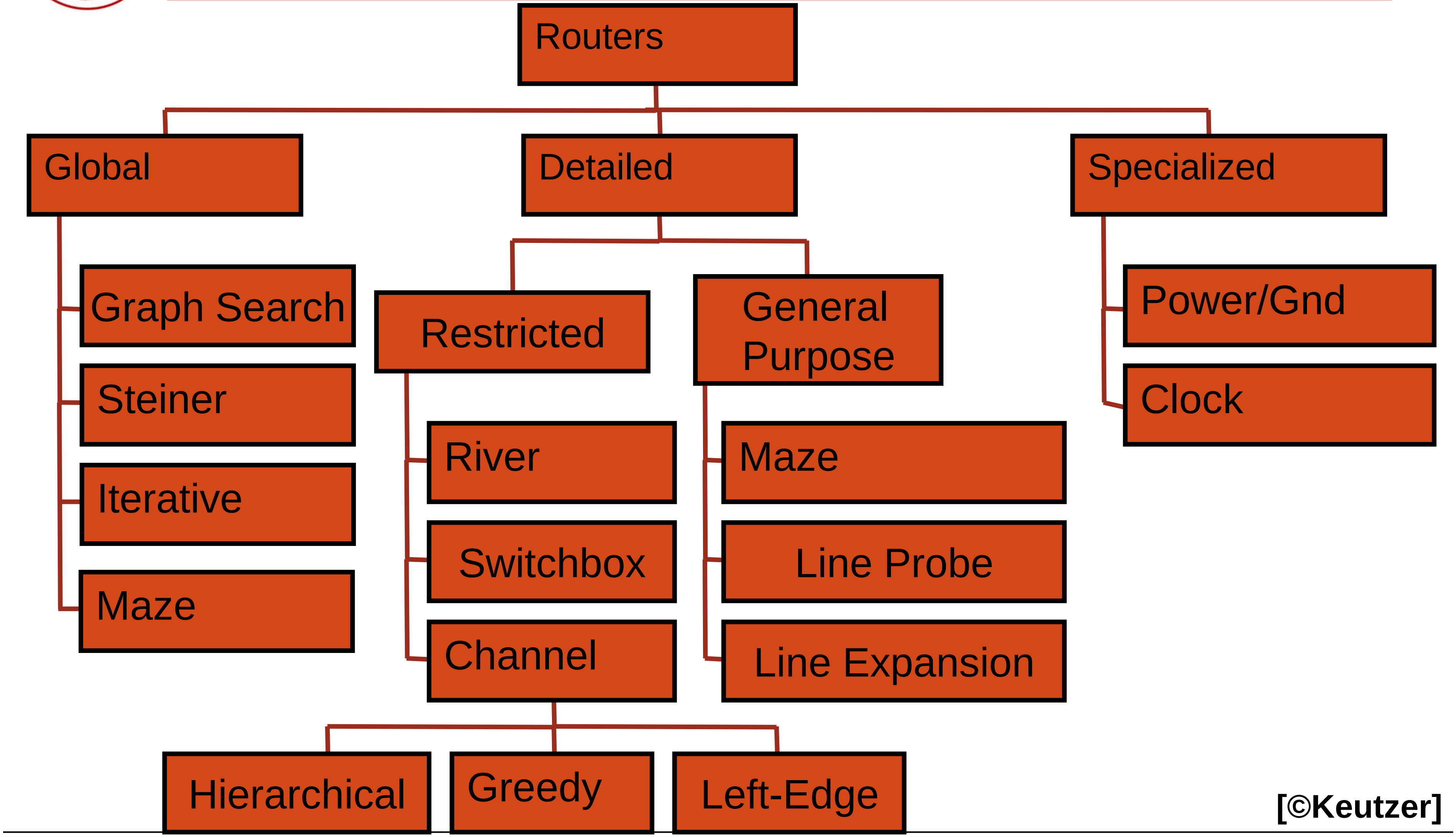


- Global routing
  - Input: detailed placement, with exact terminal locations
  - Determine "channel" (routing region) for each net
  - Objective: minimize area (congestion), and timing (approximate)
- Detailed routing
  - Input: channels and approximate routing from the global routing phase
  - Determine the exact route and layers for each net
  - Objective: valid routing, minimize area (congestion), meet timing constraints
  - Additional objectives: min via, power



Figs. [©Sherwani]

# Taxonomy of VLSI Routers



Routers

Global

Graph Search

Steiner

Iterative

Maze

Detailed

Restricted

River

Switchbox

Channel

Hierarchical

Greedy

Left-Edge

General Purpose

Maze

Line Probe

Line Expansion

Specialized

Power/Gnd

Clock

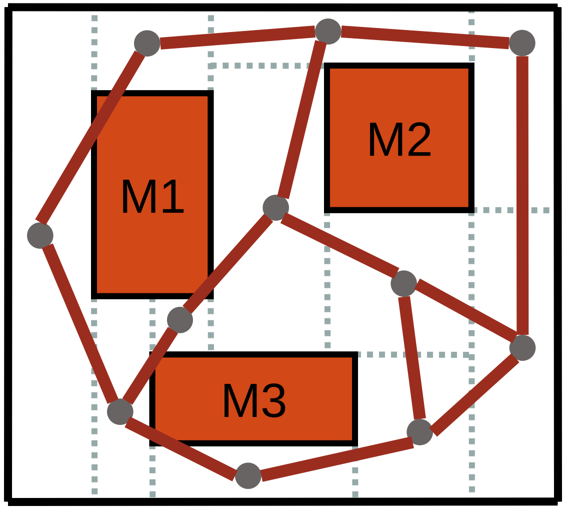
**[©Keutzer]**

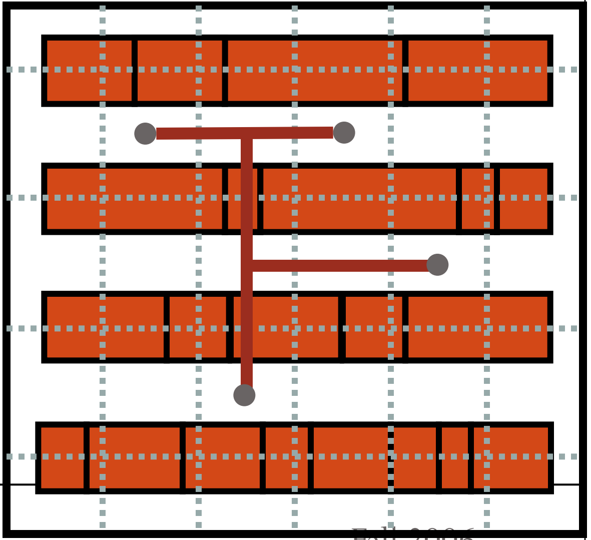
# Global Routing

- Stages
  - Routing region definition
  - Routing region ordering
  - Steiner-tree / area routing
- Grid
  - Tiles super-imposed on placement
  - Regular or irregular
  - Smaller problem to solve, higher level of abstraction
  - Terminals at center of grid tiles
- Edge capacity
  - Number of nets that can pass a certain grid edge (aka congestion)
  - On edge $E_{ij}$,

$$Capacity(E_{ij}) \geq Congestion(E_{ij})$$

[©Sarrafzadeh]



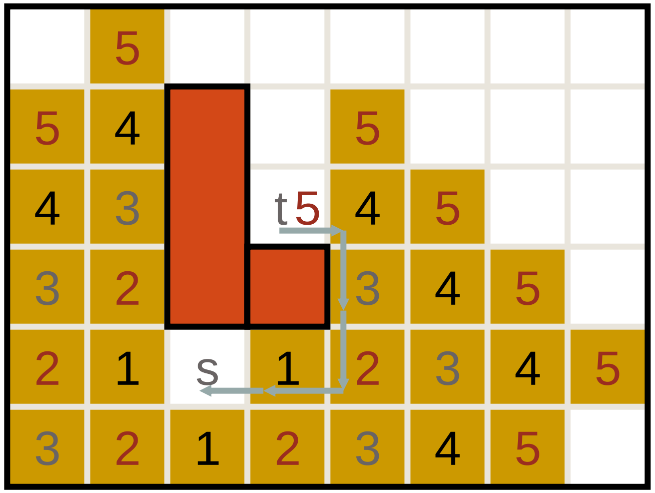


# Global Routing – Graph Search

- Good for two-terminal nets
- Build grid graph (Coarse? Fine?)
- Use graph search algorithms, e.g., Dijkstra
- Iterative: route nets one by one
- How to handle:
  - Congestion?
  - Critical nets?
- Order of the nets to route?
  - Net criticality
  - Half-perimeter of the bounding box
  - Number of terminals

# Global Routing – Maze Routing

- Similar to breadth-first search
  - Very simple algorithm
  - Works on grid graph
  - Time complexity: grid size (NxN)
- Algorithm
  - Propagate a "wave" from source until hit the sink (implemented using a queue)
  - Trace back to find the path
- Guaranteed to find the optimal solution
  - Usually multiple optimal solutions exist
- More than two terminals?
  - For the third terminal, use the path between the first two as the source of the wave



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 5 | | | | | | |
| 5 | 4 | | | 5 | | | |
| 4 | 3 | | t5 | 4 | 5 | | |
| 3 | 2 | | | 3 | 4 | 5 | |
| 2 | 1 | s | 1 | 2 | 3 | 4 | 5 |
| 3 | 2 | 1 | 2 | 3 | 4 | 5 | |

# Maze Routing

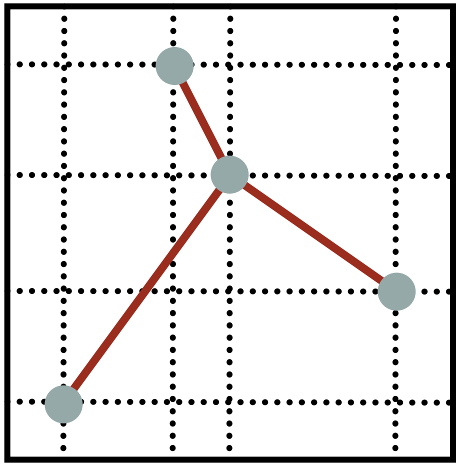
- Key to popularity:
  - Simplicity
  - Guaranteed to find the optimal solution
  - Can realize more complex cost functions too (e.g., number of bends in a path)
- Weakness:
  - Multiple terminals not handled efficiently
  - Dependent on grid, a two dimensional data structure
- Different variations exist
  - Soukup’s alg:
    - First use DFS, when get to an obstacle, use BFS to get around
    - No guarantee to find the shortest path

# Multiple Terminal Nets: Steiner Tree

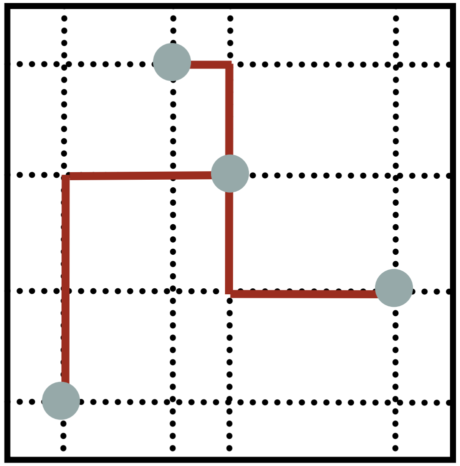
- Steiner tree (aka Rectilinear Steiner Tree – RST):
  - A tree connecting multiple terminals
    - Original points: “demand points” – set D
    - Added points: “Steiner points” – set S
  - Edges horizontal or vertical only
- Steiner Minimum Tree (SMT)
  - Similar to minimum spanning tree (MST)
  - But finding SMT is NP-complete
  - Many good heuristics introduced to find SMT
- Algorithm
  - Find MST
  - Pass horizontal and vertical lines from each terminal to get the Hannan grid (optimal solution is on this grid)
  - Convert each edge of the MST to an L-shaped route on Hannan grid (add a Steiner point at the corner of L)

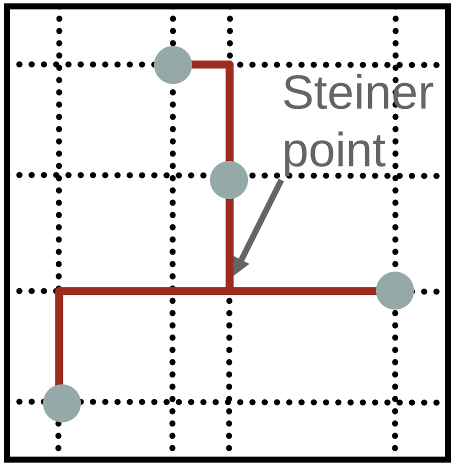
# Steiner Tree

- Hannan grid reduces solution space (smaller grid)
  - For min length RST, Steiner points always on Hannan grid
- Convert MST to rectilinear paths
  - Length bounded by 1.5 times optimal SMT length
- Use alternate "L" routes to find the minimum tree



MSP (length=11)





Steiner tree (len=13)

[©Sherwani]

# Steiner Tree Routing

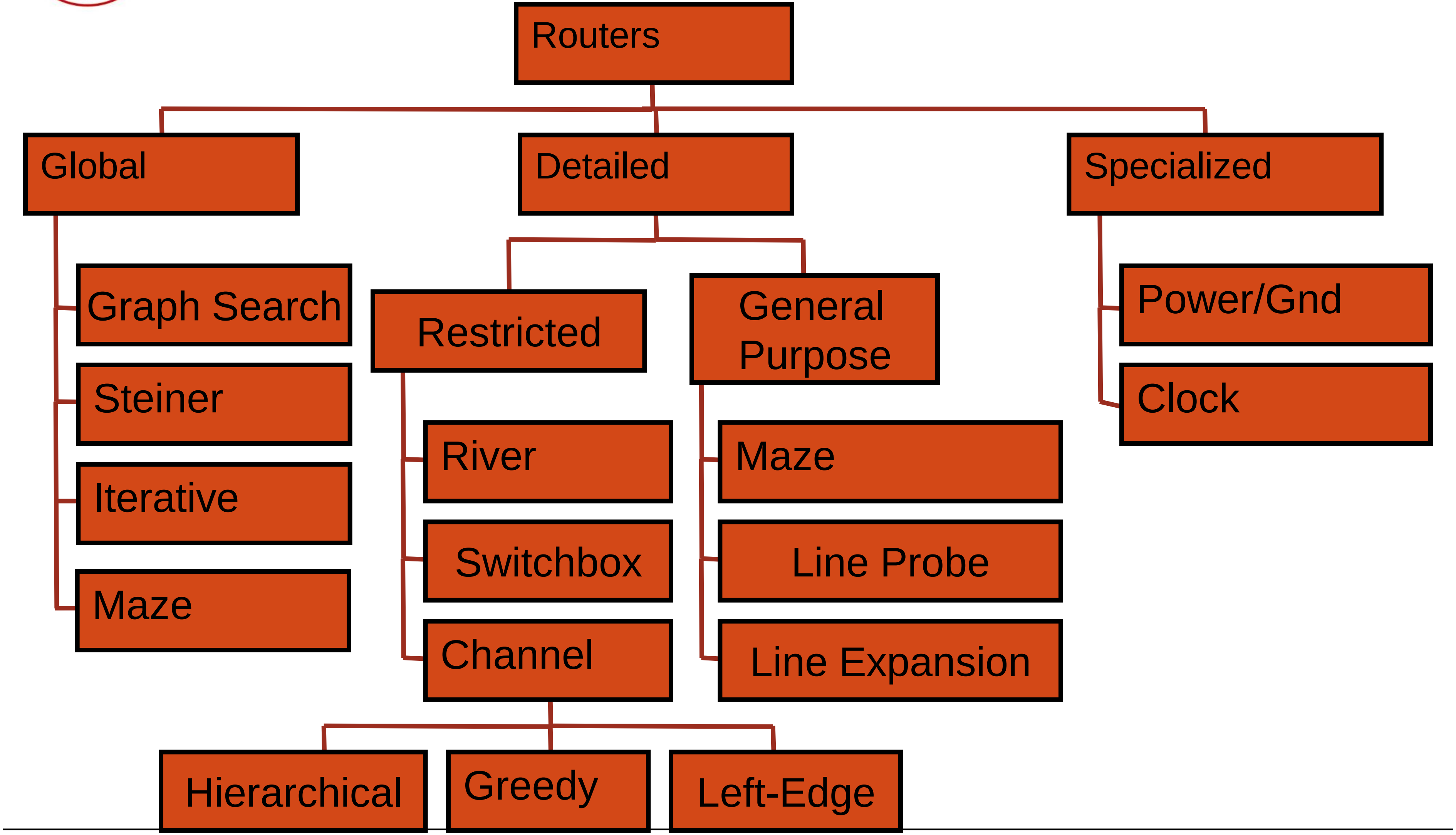
- Can apply different costs to different regions (or horizontal/vertical preference)
- Order of the nets
  - Sequential
    - Use # of terminals, criticality, etc. to determine order
  - Parallel
    - Divide the chip into large regions, perform the routing in parallel
- Key to popularity
  - Fast (not theoretically, but practically)
  - Bounded solution quality
- Shortcomings
  - Difficult to predict or avoid congestion

# Global Routing Approaches

- A combination of different approaches might be used in chip-level routing
  - Route simple nets (2-3 pins in local area) directly (e.g., L-shaped or Z-shaped)
  - Use a "close to optimal" Steiner Tree algorithms to route nets of intermediate length
  - Route remaining "big" nets using a maze router
- Ordering
  - Some ordering is chosen, if can route all, then done, otherwise:
  - Rip-up and Re-route

[©Keutzer]

# Taxonomy of VLSI Routers

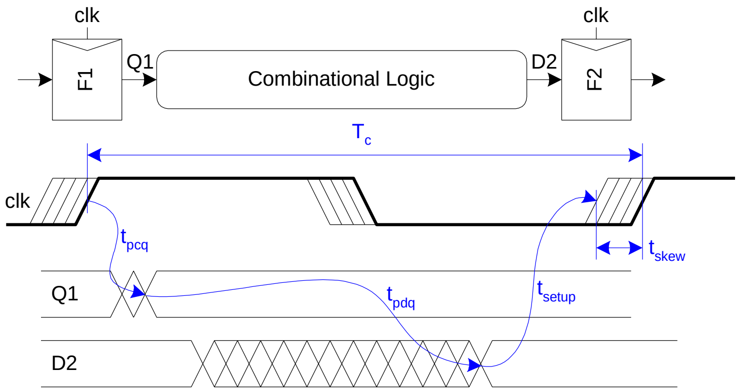


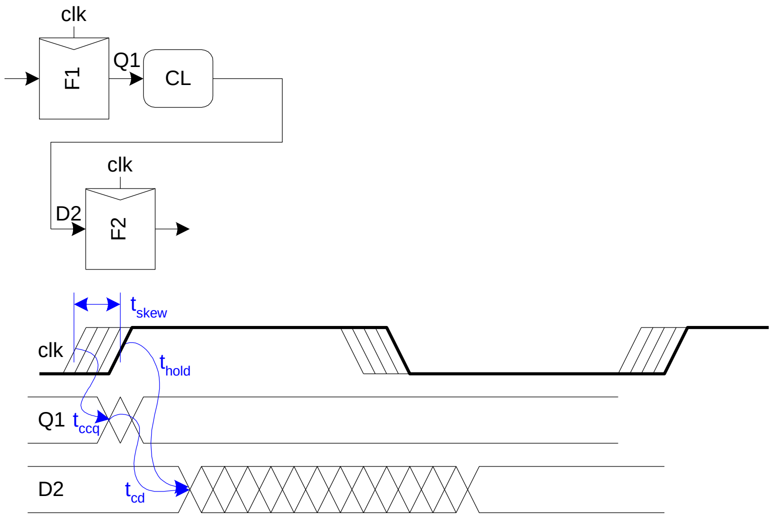
[©Keutzer]

# Clock Skew

$$t_{pd} \le T_c - \underbrace{\left(t_{pcq} + t_{\text{setup}} + t_{\text{skew}}\right)}_{\text{sequencing overhead}}$$

$$t_{cd} \ge t_{\text{hold}} - t_{ccq} + t_{\text{skew}}$$





# The Clock Routing Problem

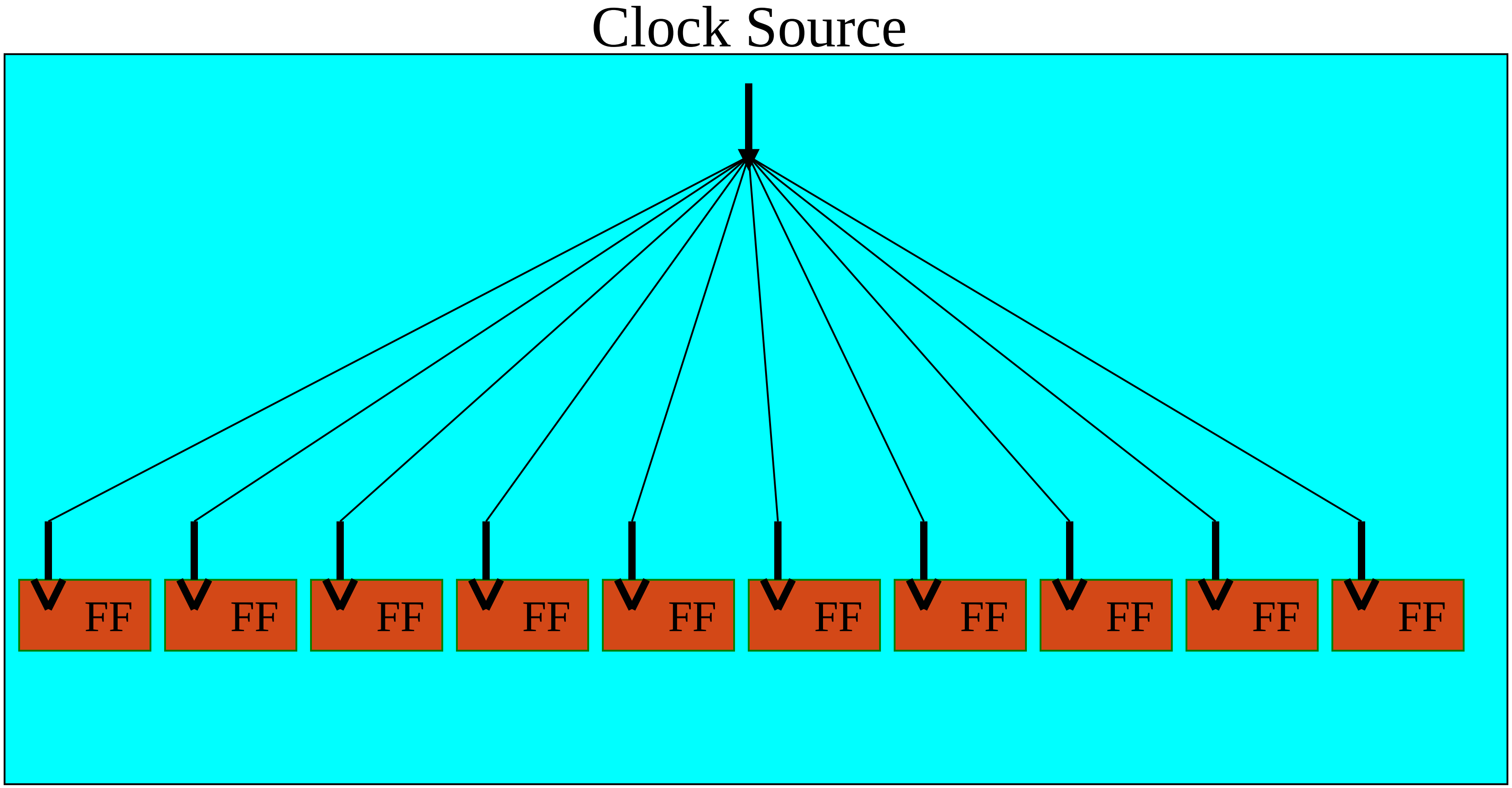
- Given a source and *n* sinks.
- Connect all sinks to the source by an interconnect tree so as to minimize:
  - Clock Skew = $\max_{i,j} |t_i - t_j|$
  - Delay = $\max_i t_i$
  - Total wirelength
  - Noise and coupling effect

# Clock Design Considerations

- Clock signal is global in nature, so clock nets are usually very long.
  - Significant interconnect capacitance and resistance
- So what are the techniques?
  - Routing
    - Clock tree versus clock clock mesh (grid)
    - Balance skew and total wire length
  - Buffer insertion
    - Clock buffers to reduce clock skew, delay, and distortion in waveform.
  - Wire sizing
    - To further tune the clock tree/mesh

# Clock trees

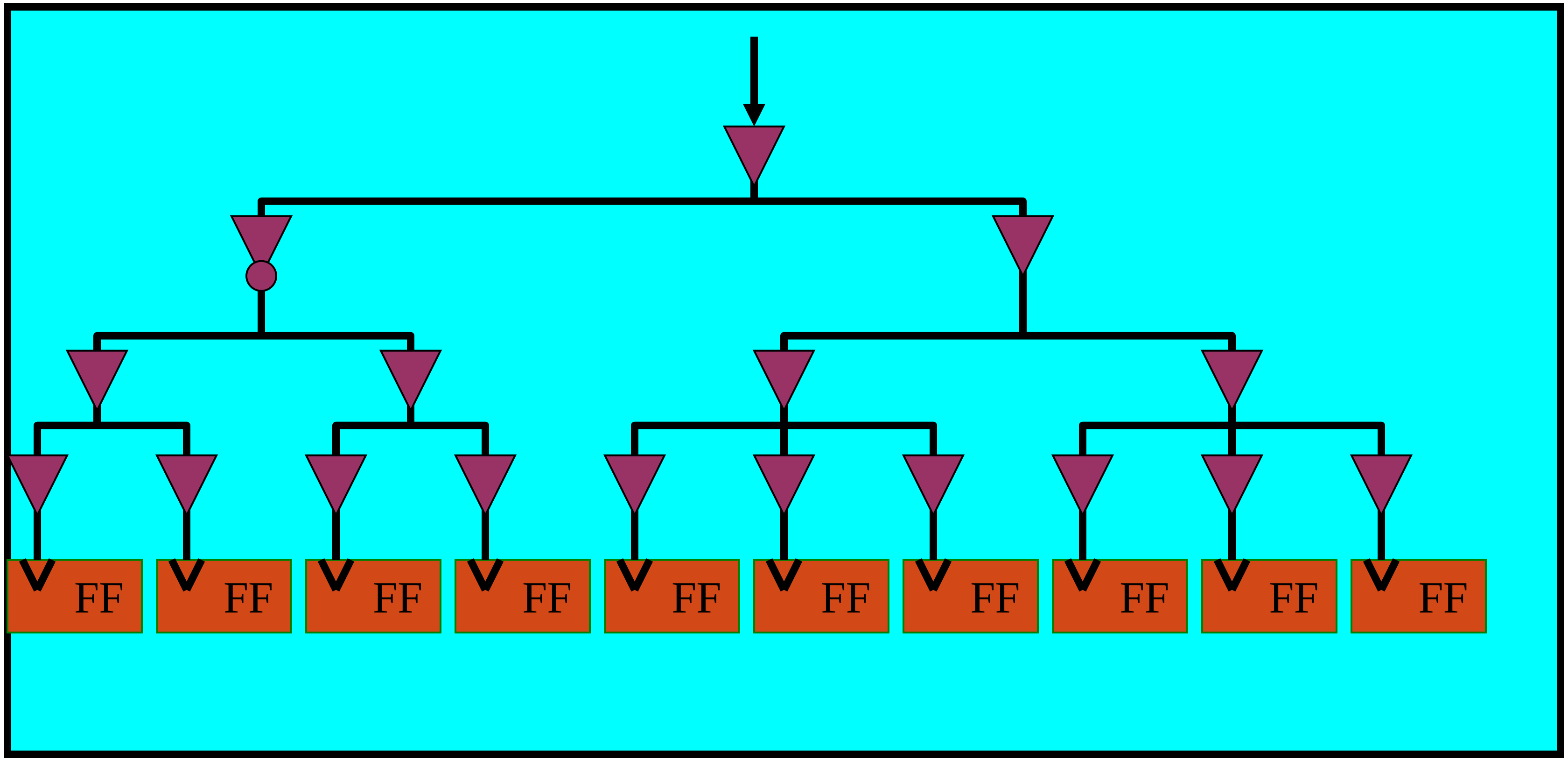
- A path from the clock source to clock sinks



# Clock trees

- A path from the clock source to clock sinks

Clock Source



# An Exact Zero Skew Algorithm [Tsay'93]

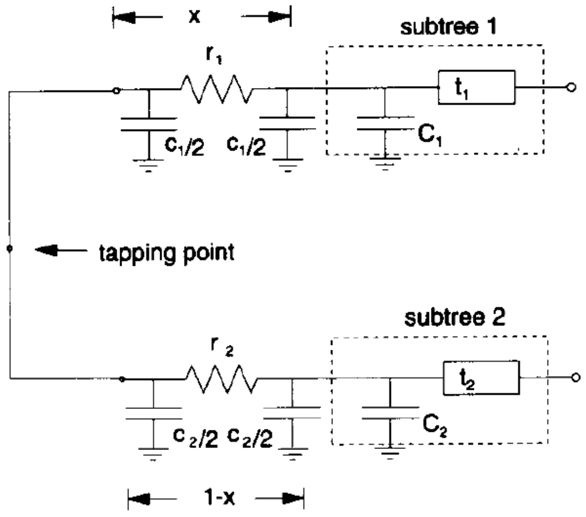
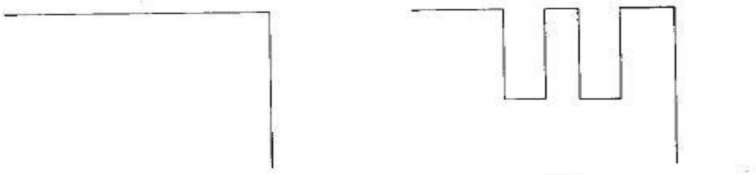


Fig. 5. *Zero-Skew Merge* of two subtrees.

$$r_1(c_1/2 + C_1) + t_1 = r_2(c_2/2 + C_2) + t_2)$$



(a)

(b)

Fig. 6. (a) A regular wire. (b) Elongation of the wire by *snaking*.

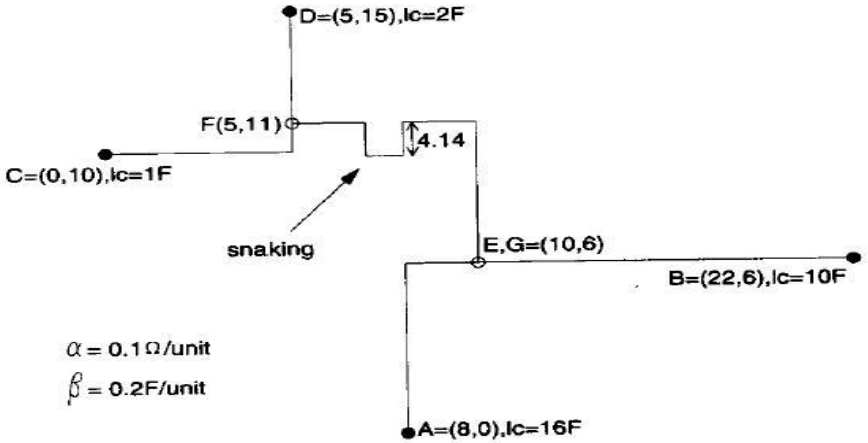


Fig. 7. A zero-skew wiring result of a simple example.

[Tsay'93]: Ren-Song Tsay, "An Exact Zero-Skew Clock Routing Algorithm", IEEE Trans. On CAD of IC, Vol.12, No.2, 1993.

# References

- Reference 1: Chapter 12
- Reference 2: Chapter 5, 6