
Routing

Virendra Singh
Indian Institute of Science
Bangalore

Routing

- Problem

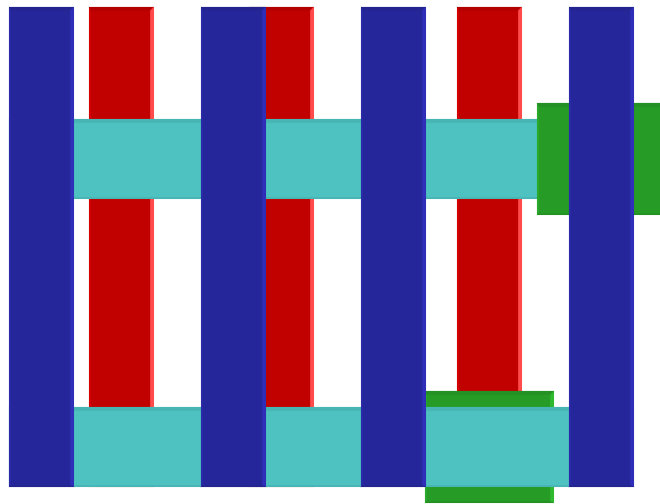
- Given a placement, and a fixed number of metal layers, find a valid pattern of horizontal and vertical wires that connect the terminals of the nets
- Levels of abstraction:
 - Global routing
 - Detailed routing

- Objectives

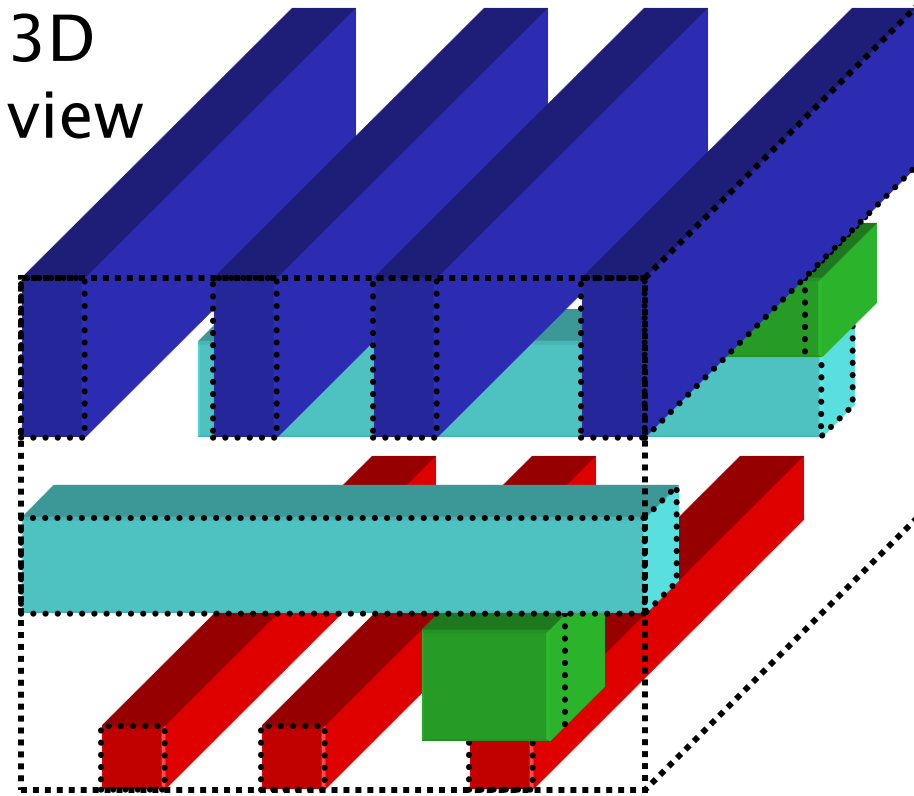
- Cost components:
 - Area (channel width) – min congestion in prev levels helped
 - Wire delays – timing minimization in previous levels
 - Number of layers (fewer layers → less expensive)
 - Additional cost components: number of bends, vias

Routing Anatomy

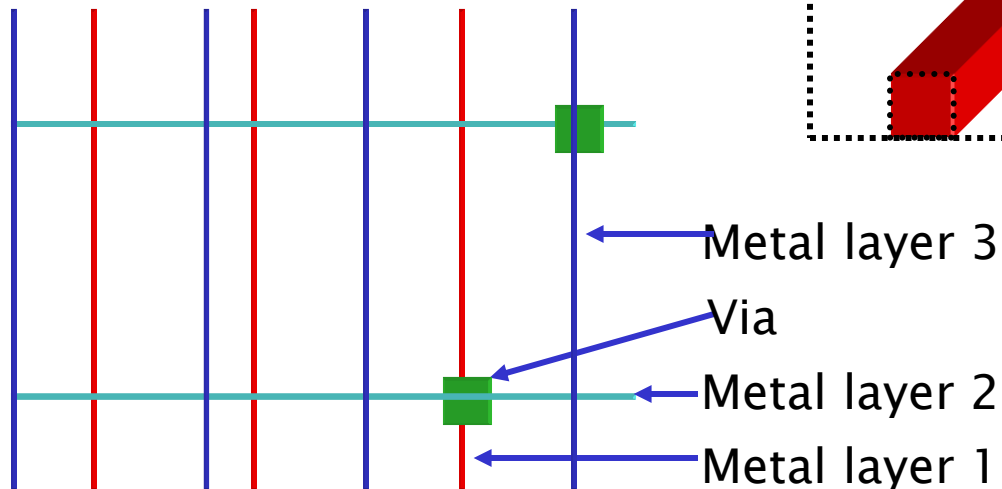
Top view



3D view



Symbolic Layout

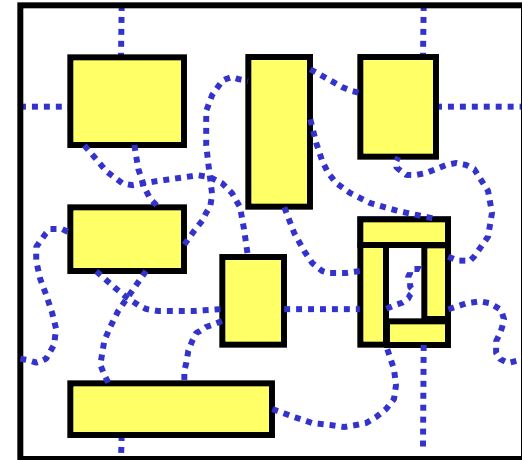


Note: Colors used in this slide are not standard

Global vs. Detailed Routing

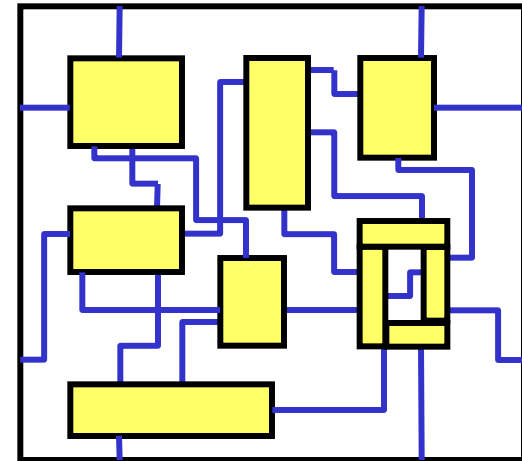
- Global routing

- 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



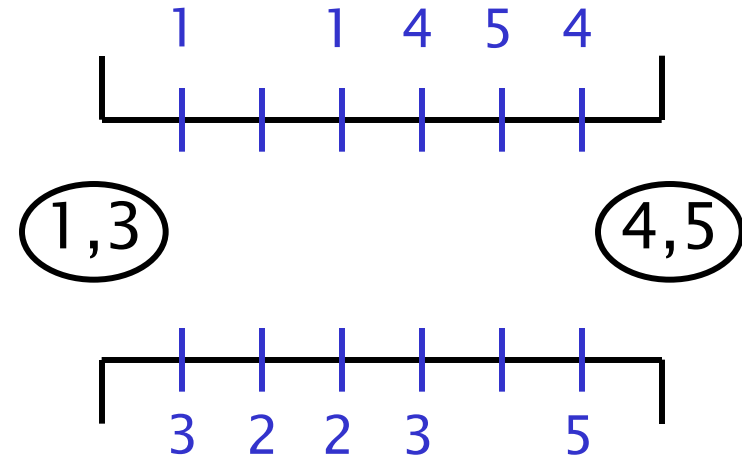
Figs. [©Sherwani]

Routing Environment

- Routing regions

- Channel

- Fixed height ?
(→ fixed number of tracks)
 - Fixed terminals on top and bottom
 - More constrained problem: switchbox.
Terminals on four sides fixed



- Area routing

- Wires can pass through any region not occupied by cells
(exception: over-the-cell routing)

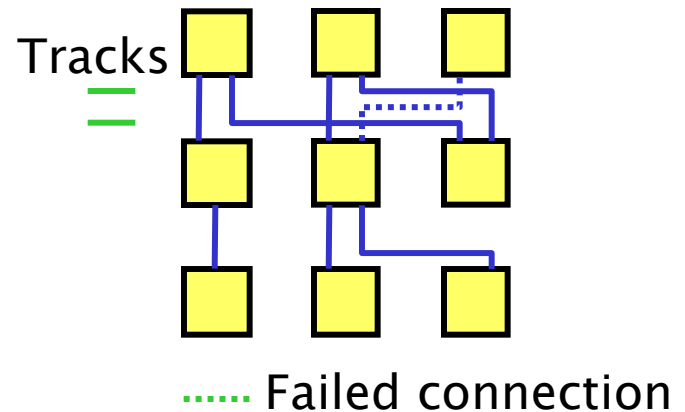
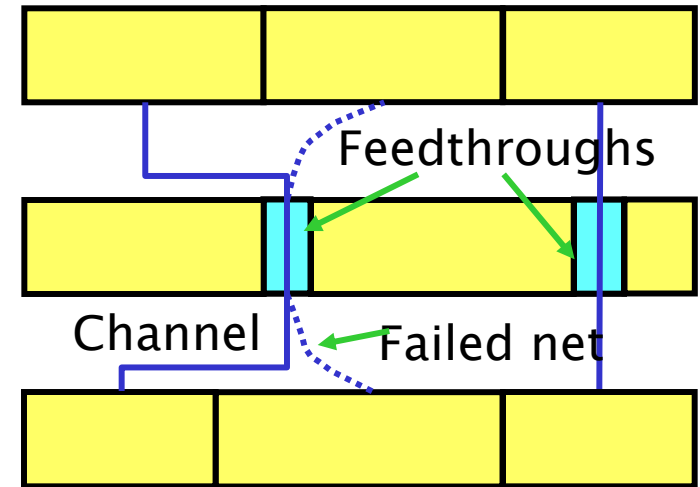
- Routing layers

- Could be pre-assigned (e.g., M1 horizontal, M2 vert.)
 - Different weights might be assigned to layers

Routing Environment

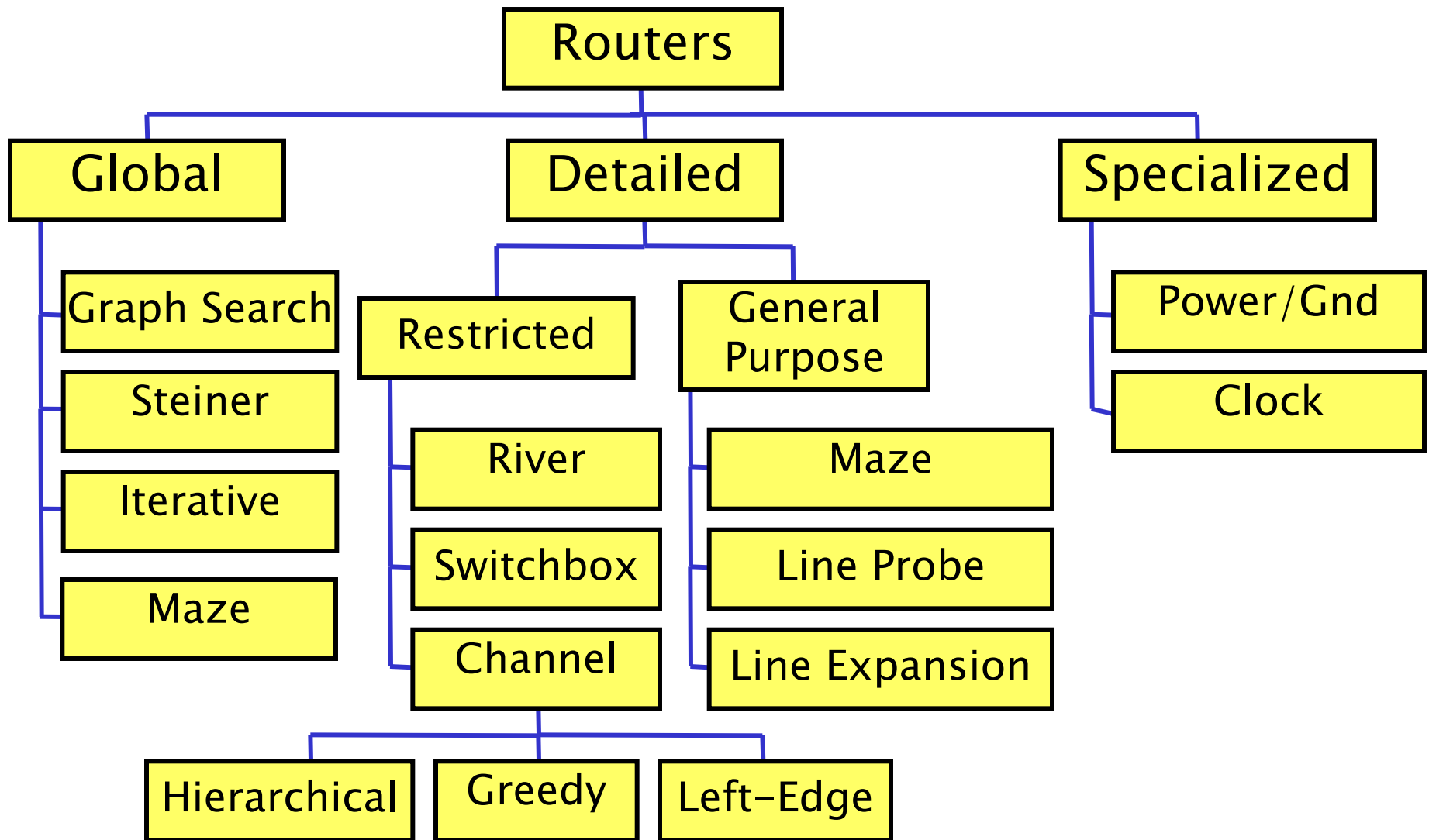
- Chip architecture

- Full-custom:
 - No constraint on routing regions
- Standard cell:
 - Variable channel height?
 - Feed-through cells connect channels
- FPGA:
 - Fixed channel height
 - Limited switchbox connections
 - Prefabricated wire segments have different weights



Figs. [©Sherwani]

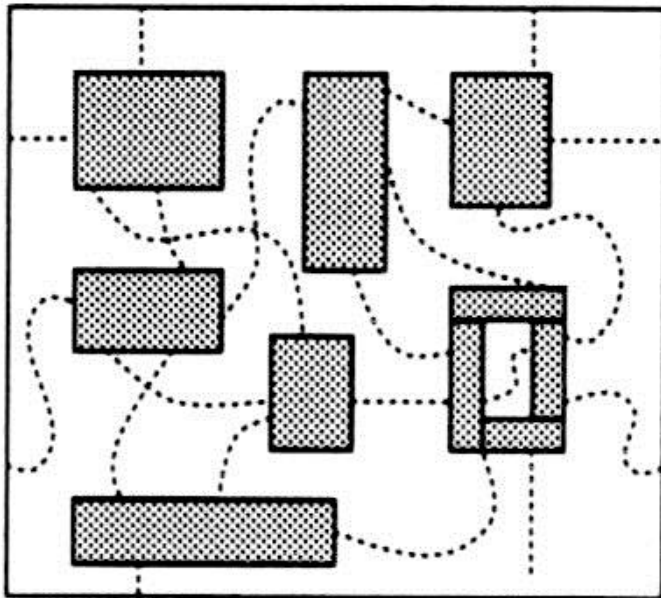
Taxonomy of VLSI Routers



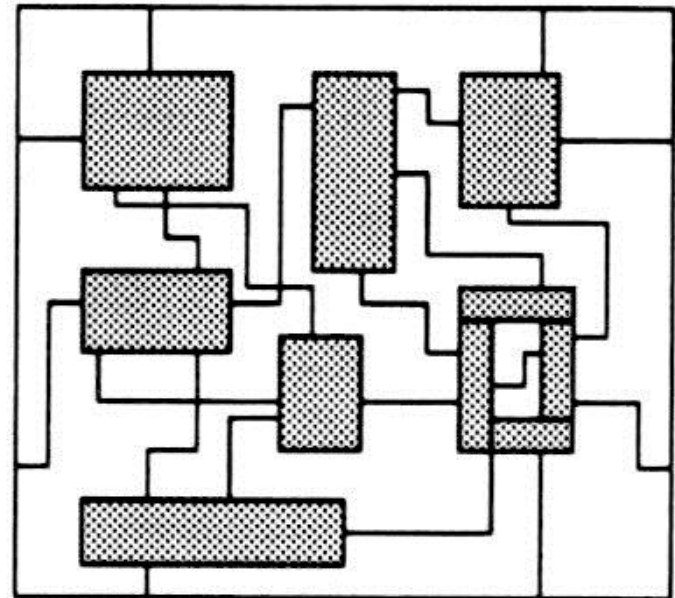
[©Keutzer]

General Routing Problem

Two phases:



Global Routing



Detailed Routing

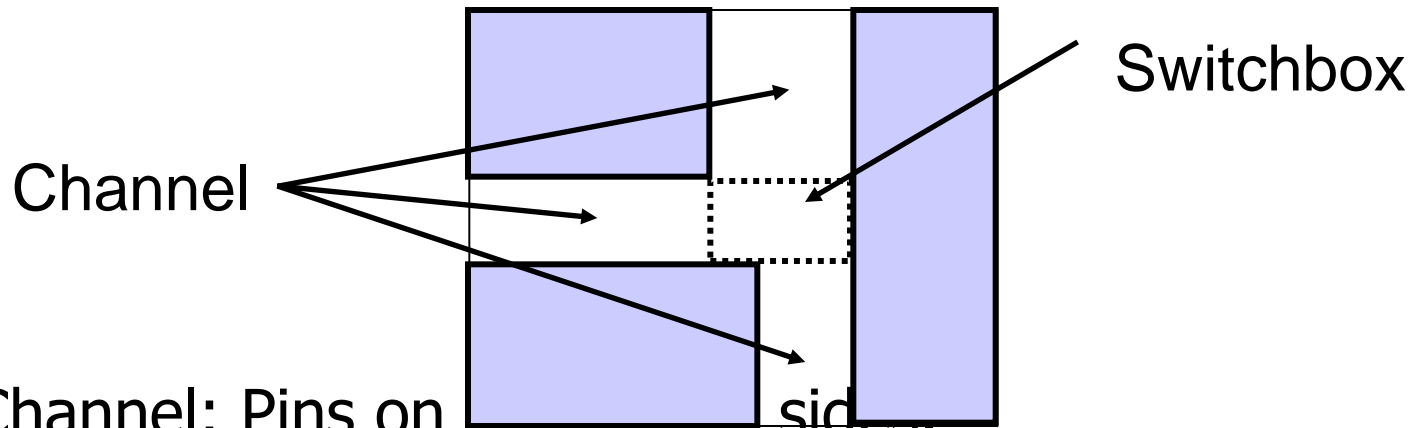
Global Routing

Global routing is divided into 3 phases:

1. Region definition
2. Region assignment
3. Pin assignment to routing regions

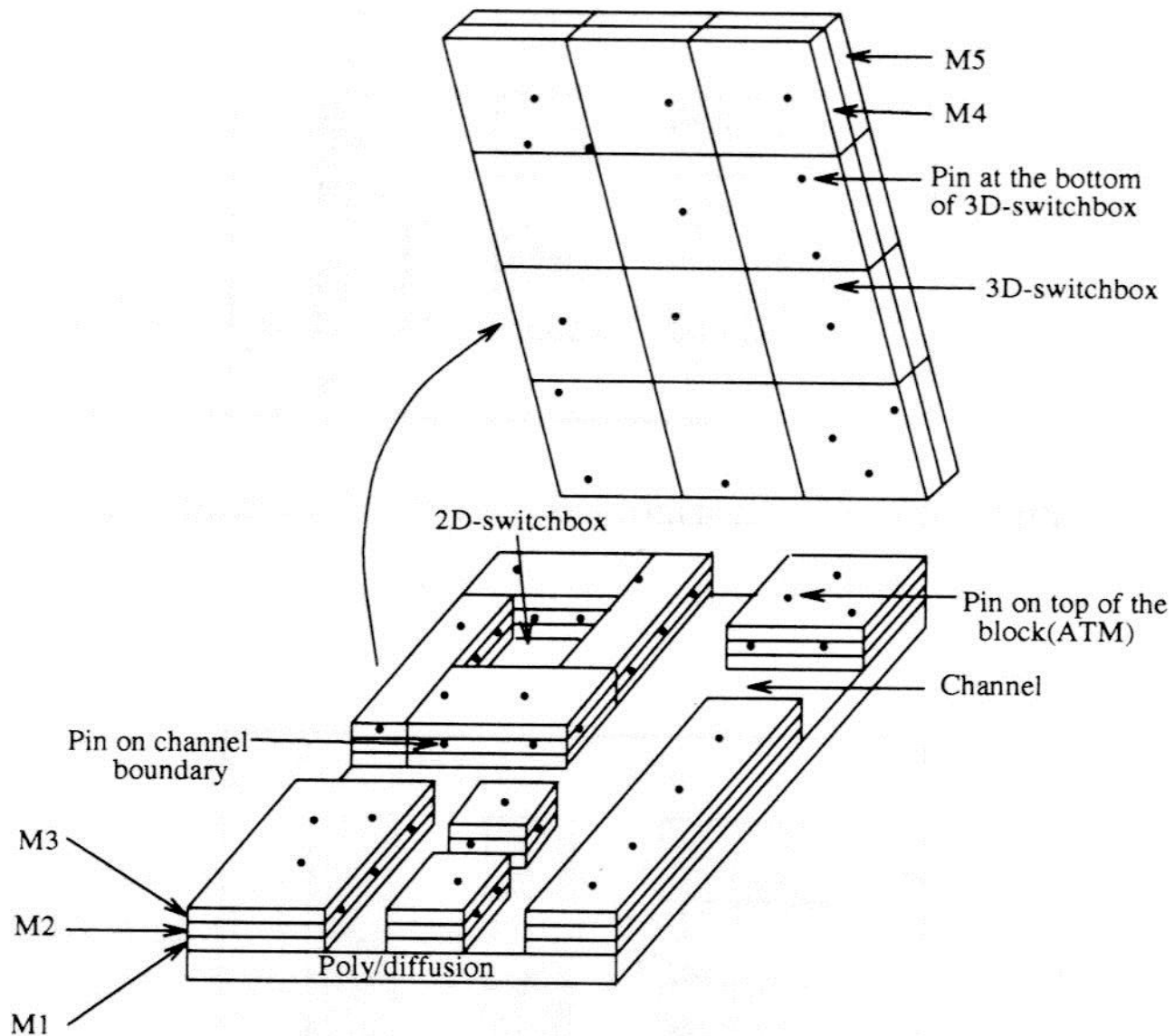
Region Definition

Divide the routing area into routing regions of simple shape (rectangular):



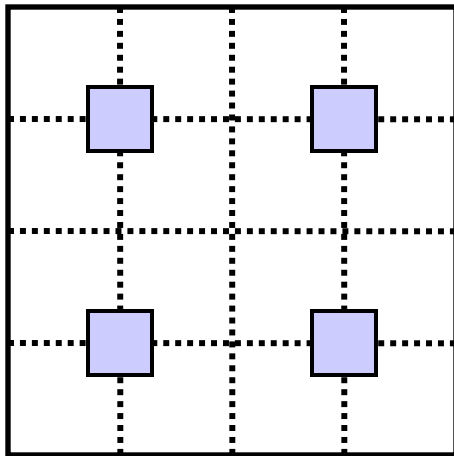
- Channel: Pins on opposite sides.
- 2-D Switchbox: Pins on 4 sides.
- 3-D Switchbox: Pins on all 6 sides.

Routing Regions

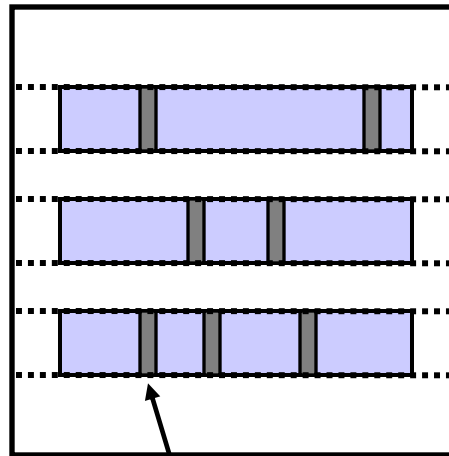


Routing Regions in Different Design Styles

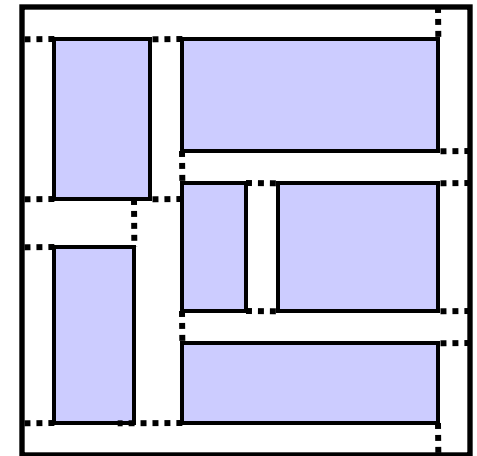
Gate-Array



Standard-Cell



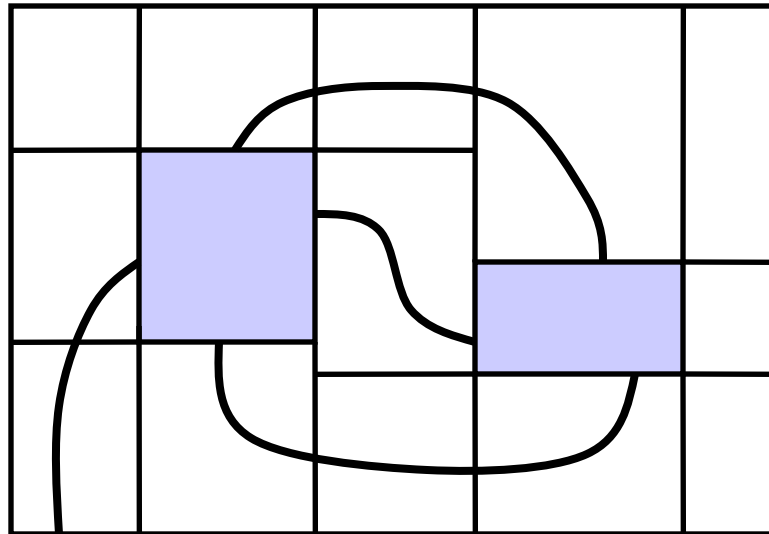
Full-Custom



Feedthrough Cell

Region Assignment

Assign routing regions to each net. Need to consider timing budget of nets and routing congestion of the regions.



Approaches for Global Routing

Sequential Approach:

- Route the nets one at a time.
- Order dependent on factors like criticality, estimated wire length, etc.
- If further routing is impossible because some nets are blocked by nets routed earlier, apply Rip-up and Reroute technique.
- This approach is much more popular.

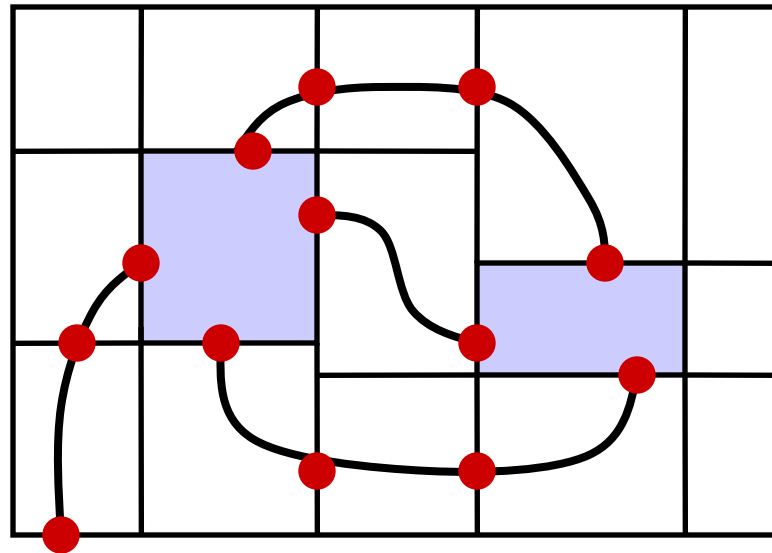
Approaches for Global Routing

Concurrent Approach:

- Consider all nets simultaneously.
- Can be formulated as an integer program.

Pin Assignment

Assign pins on routing region boundaries for each net. (Prepare for the detailed routing stage for each region.)



Detailed Routing

- Three types of detailed routings:
 - Channel Routing
 - 2-D Switchbox Routing
 - 3-D Switchbox Routing
- Channel routing → 2-D switchbox → 3-D switchbox
- If the switchbox or channels are unroutable without a large expansion, global routing needs to be done again.

Extraction and Timing Analysis

- After global routing and detailed routing, information of the nets can be extracted and delays can be analyzed.
- If some nets fail to meet their timing budget, detailed routing and/or global routing needs to be repeated.

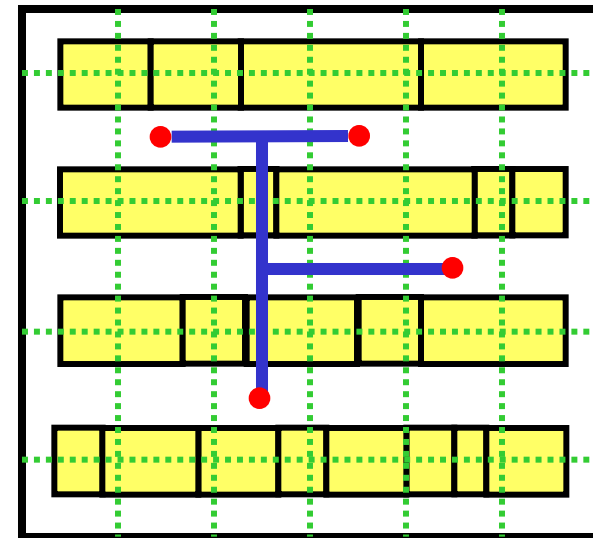
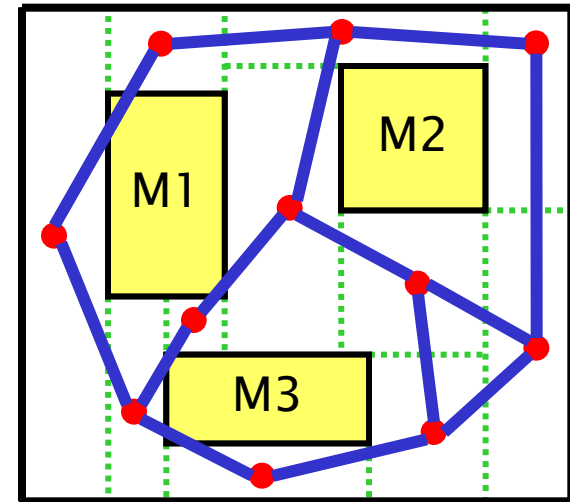
Kinds of Routing

- Global Routing
- Detailed Routing
 - Channel
 - Switchbox
- Others:
 - Maze routing
 - Over the cell routing
 - Clock routing

Global Routing

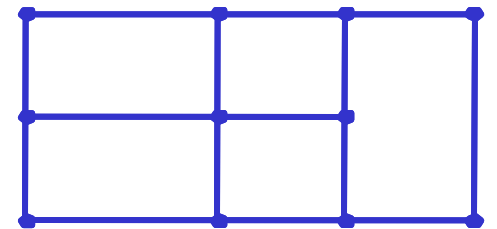
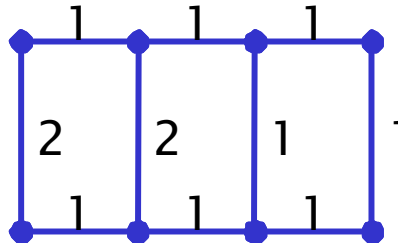
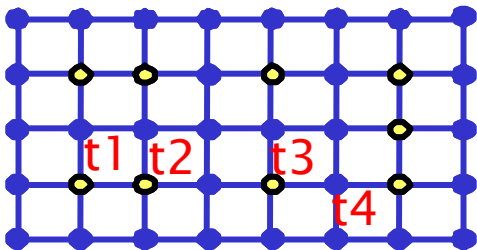
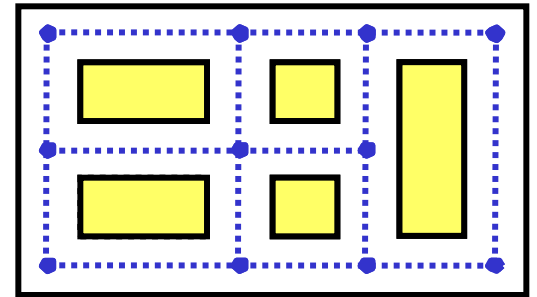
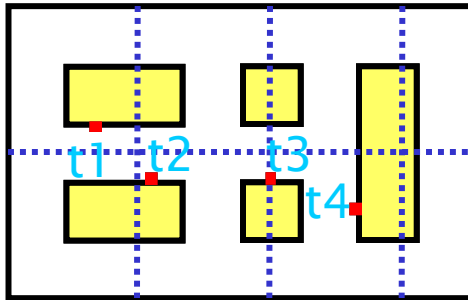
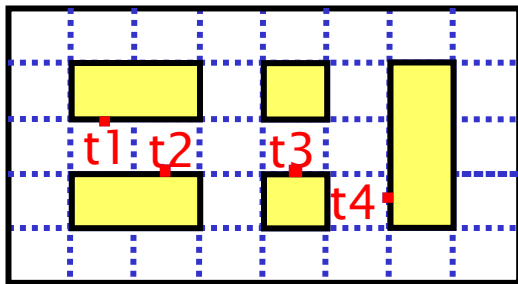
[©Sarrafzadeh]

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



Grid Graph

- Course or fine-grain
- Vertices: routing regions, edges: route exists?
- Weights on edges
 - How costly is to use that edge
 - Could vary during the routing (e.g., for congestion)
 - Horizontal / vertical might have different weights



Global Routing – Graph Search

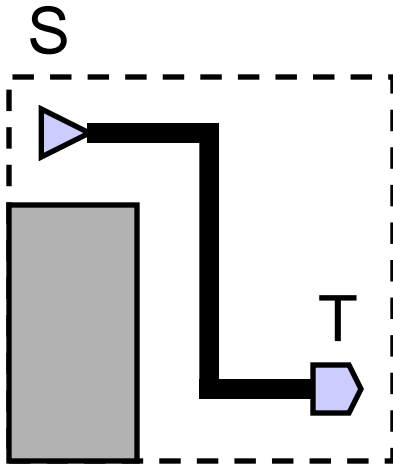
- Good for two-terminal nets
- Build grid graph (Coarse? Fine?)
- Use graph search algorithms, e.g., Dijkstra
- Iterative: route nets one by one
- How to handle:
 - Congestion?
 - Critical nets?
- Order of the nets to route?
 - Net criticality
 - Half-perimeter of the bounding box
 - Number of terminals

Maze Routing

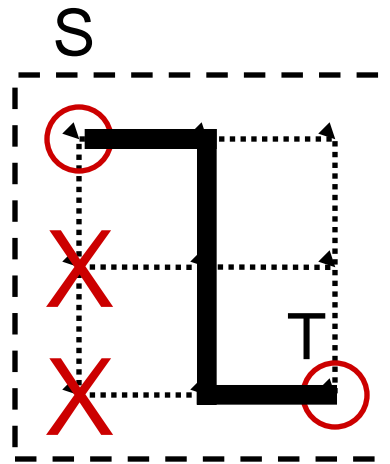
Maze Routing Problem

- **Given:**
 - A planar rectangular grid graph.
 - Two points S and T on the graph.
 - Obstacles modeled as blocked vertices.
- **Objective:**
 - Find the shortest path connecting S and T.
- This technique can be used in global or detailed routing (switchbox) problems.

Grid Graph



Area Routing

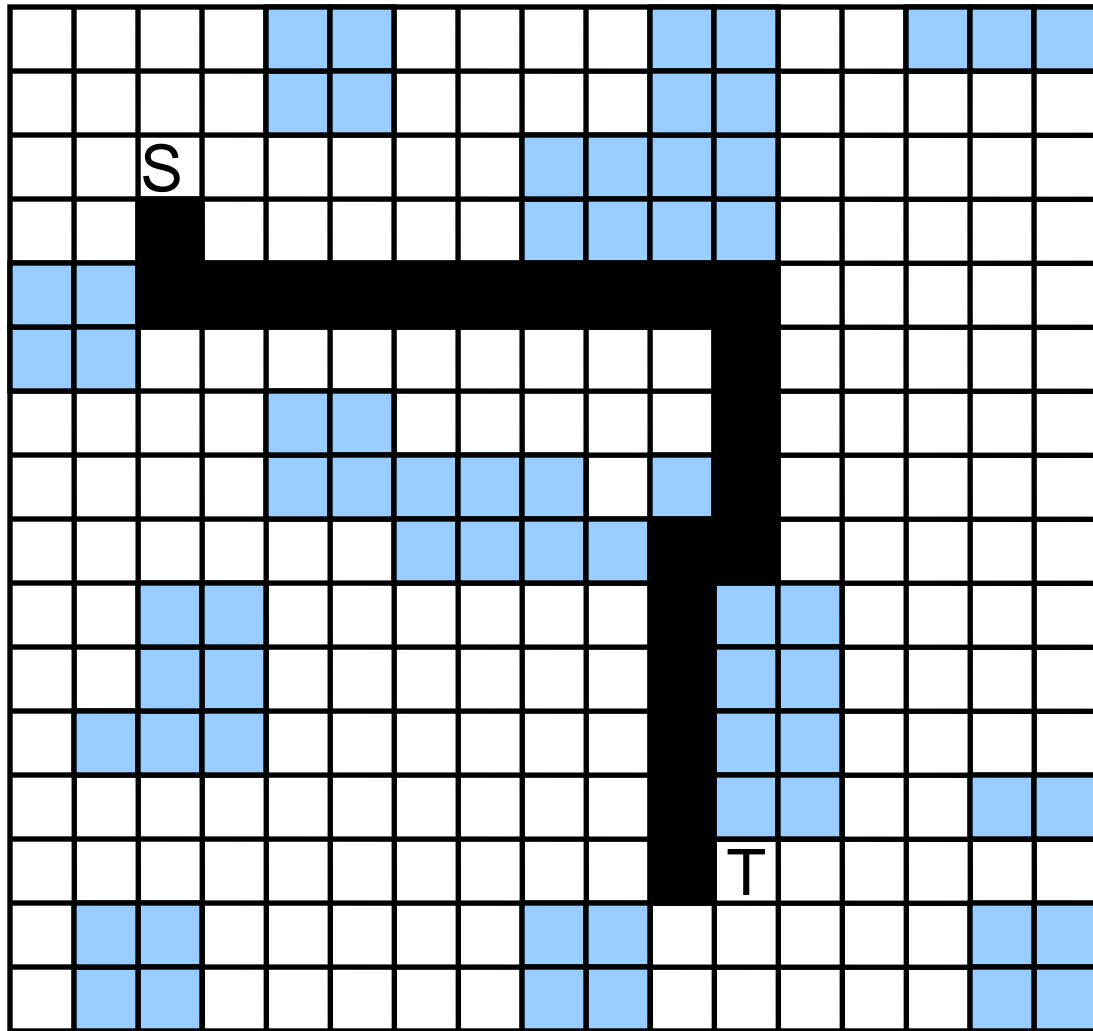


Grid Graph
(Maze)

S	✓	
X	✓	
X	✓	T

Simplified
Representation

Maze Routing



Lee's Algorithm

“An Algorithm for Path Connection and its Application”, C.Y. Lee, IRE Transactions on Electronic Computers, 1961.

Basic Idea

- A Breadth-First Search (BFS) of the grid graph.
- Always find the shortest path possible.
- Consists of two phases:
 - Wave Propagation
 - Retrace

An Illustration

s_0	1	2	3
1	2	3	
	3	4	5
5	4	5	t_6

Wave Propagation

- At step k , all vertices at Manhattan-distance k from S are labeled with k .
- A Propagation List (FIFO) is used to keep track of the vertices to be considered next.

S_0			
			T

After Step 0

S_0	1	2	3
1	2	3	
	3		
			T

After Step 3

S_0	1	2	3
1	2	3	
	3	4	5
5	4	5	T 6

After Step 6

Retrace

- Trace back the actual route.
- Starting from T .
- *At vertex with k , go to any vertex with label $k-1$.*

S 0	1	2	3
1	2	3	
	3	4	5
5	4	5	T 6

Final labeling

— How many grids visited using Lee's algorithm?

13	12	11	10			7	6	7	7			9	10			
12	11	10	9			6	5	6	7			8	9	10	11	12
11	10	9	8	7	6	5	4					7	8	9	10	11
10	9	8	7	6	5	4	3					6	7	8	9	10
		7	6	5	4	3	2	1	2	3	4	5	6	7	8	9
		6	5	4	3	2	1	S	1	2	3	4	5	6	7	8
9	8	7	6			3	2	1	2	3	4	5	6	7	8	9
10	9	8	7						3		5	6	7	8	9	10
11	10	9	8	9	10					7	6	7	8	9	10	11
12	11			10	11	12	11	10	9	8			9	10	11	12
13	12			11	12	13	12	11	10	9			10	11	12	13
				12	13		13	12	11	10			11	12	13	
				13				13	12	11			12	13		
									13	12	T		13			
										13						

Time and Space Complexity

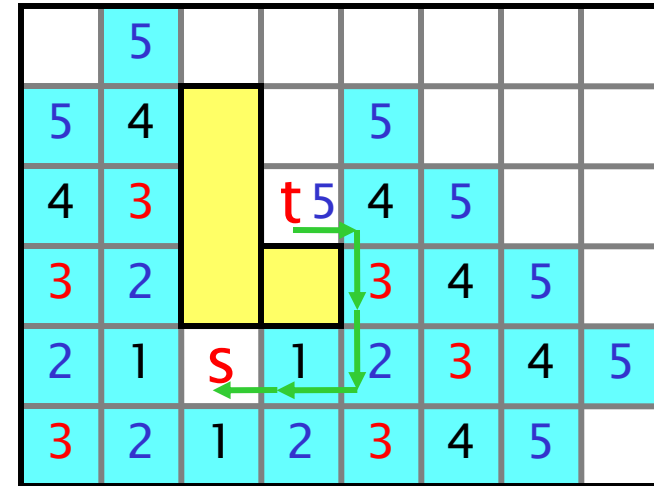
- For a grid structure of size $w \times h$:
 - Time per net = $O(wh)$
 - Space = $O(wh \log wh)$ ($O(\log wh)$ bits are needed to store each label.)
- For a 4000×4000 grid structure:
 - 24 bits per label
 - Total 48 Mbytes of memory!

Improvement to Lee's Algorithm

- Improvement on memory:
 - Aker's Coding Scheme
- Improvement on run time:
 - Starting point selection
 - Double fan-out
 - Framing
 - Hadlock's Algorithm
 - Soukup's Algorithm

Global Routing – Maze Routing

- Similar to breadth-first search
 - Very simple algorithm
 - Works on grid graph
 - Time complexity: grid size ($N \times N$)
- 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



Maze Routing

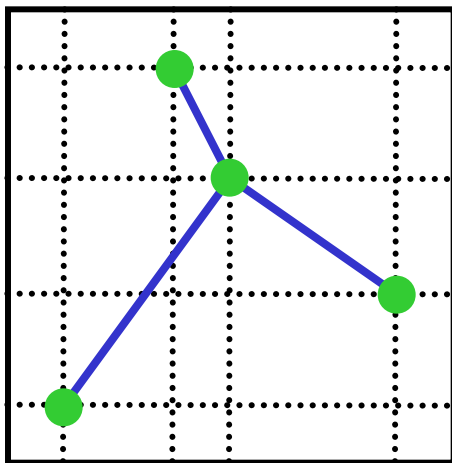
- Key to popularity:
 - Simplicity
 - Guaranteed to find the optimal solution
 - Can realize more complex cost functions too (e.g., number of bends in a path)
- Weakness:
 - Multiple terminals not handled efficiently
 - Dependent on grid, a two dimensional data structure
- Different variations exist
 - 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

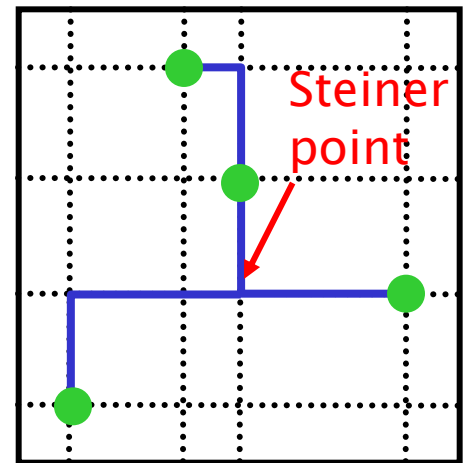
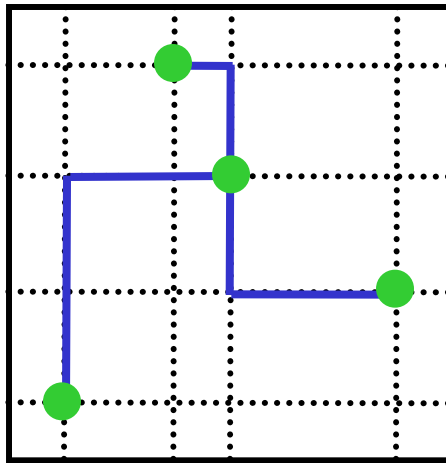
- Steiner tree (aka Rectilinear Steiner Tree – RST):
 - A tree connecting multiple terminals
 - Original points: “demand points” – set D
 - Added points: “Steiner points” – set S
 - Edges horizontal or vertical only
- Steiner Minimum Tree (SMT)
 - Similar to minimum spanning tree (MST)
 - But finding 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



MST (length=11)



Steiner tree (len=13)

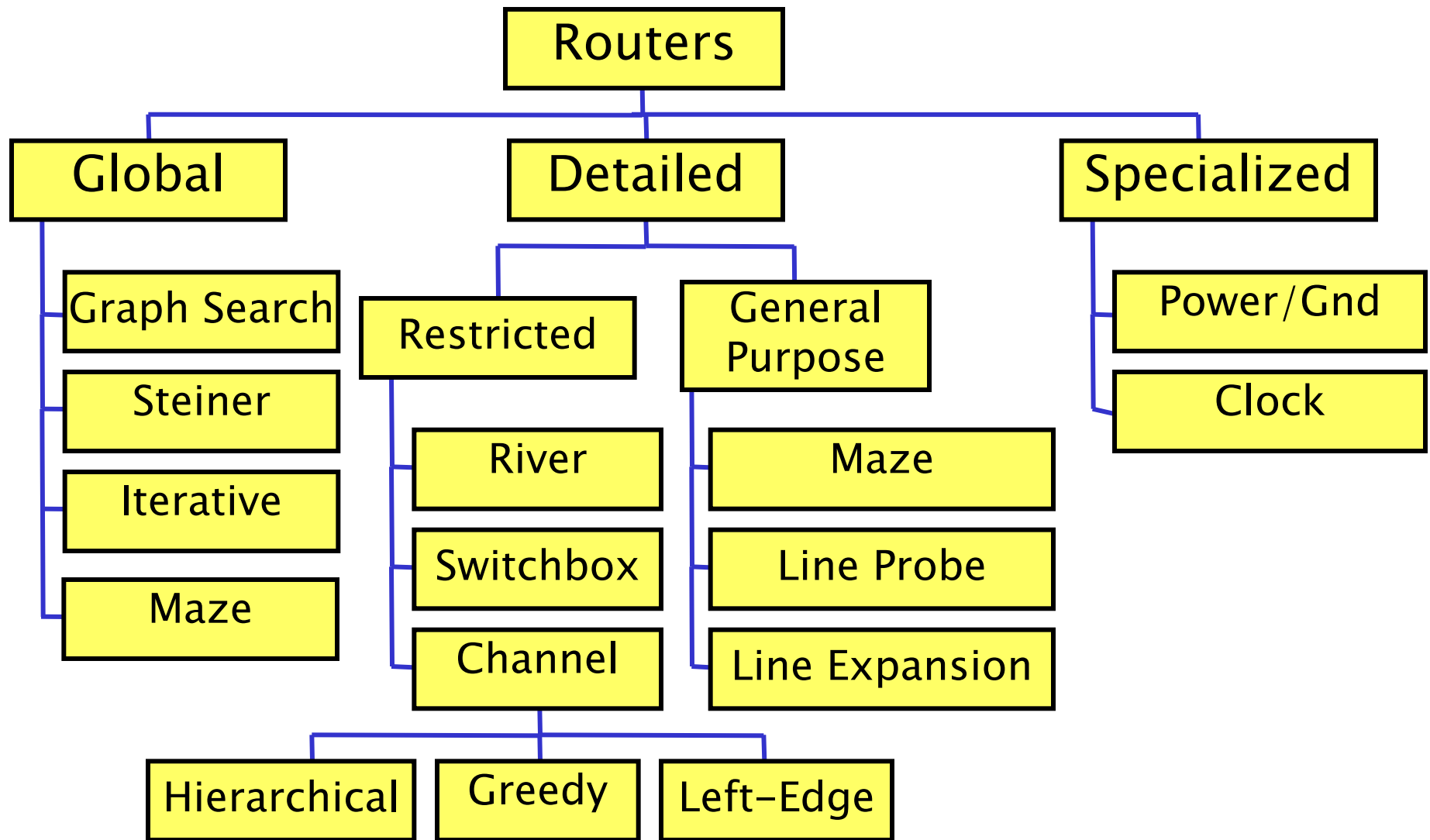
Steiner Tree Routing

- 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

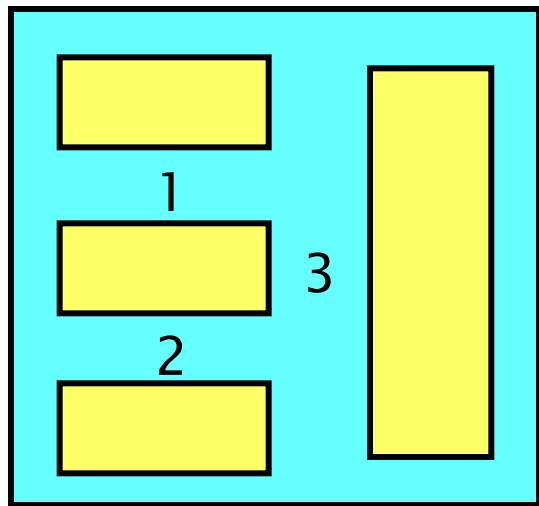
- A combination of different approaches might be used in chip-level routing
 - Route simple nets (2-3 pins in local area) directly (e.g., L-shaped or Z-shaped)
 - Use a “close to optimal” Steiner Tree algorithms to route nets of intermediate length
 - Route remaining “big” nets using a maze router
-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

Taxonomy of VLSI Routers

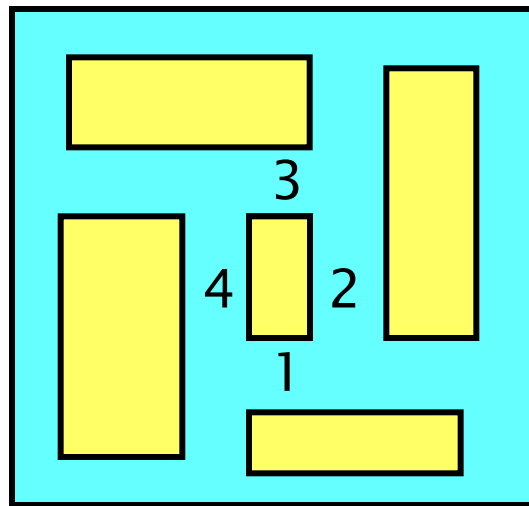


Detailed Routing: Channel vs. Switchbox

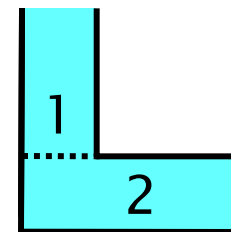
- a) Channels have no conflicts
- b) Conflicting channels
- c) Conflict resolved using L-shaped channels
 - Order matters
- d) Switchbox used to resolve the conflict
 - Order matters
 - Harder problem (compared to channel routing)



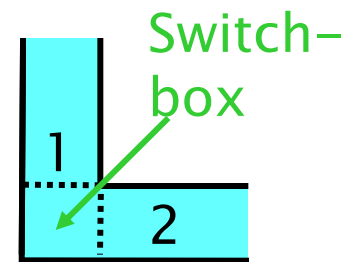
(a)



(b)



(c)



(d)

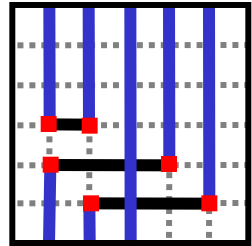
Channel Routing Problem

- Input

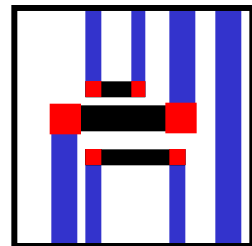
- Fixed terminal locations on the top and bottom
- Possibly floating terminals on the left and right
- Possibly fixed channel capacity constraint (capacity = max # of horizontal wires between the top and bottom boundaries of the channel)
- Either gridless (aka area-based) or grid-based

- In the algorithms we consider

- Constraints:
 - Grid structure
 - Two routing layers (one for H, another for V)
Still relevant?
- Minimize
 - # tracks (channel height)
 - Total wire length
 - # vias



Grid-based



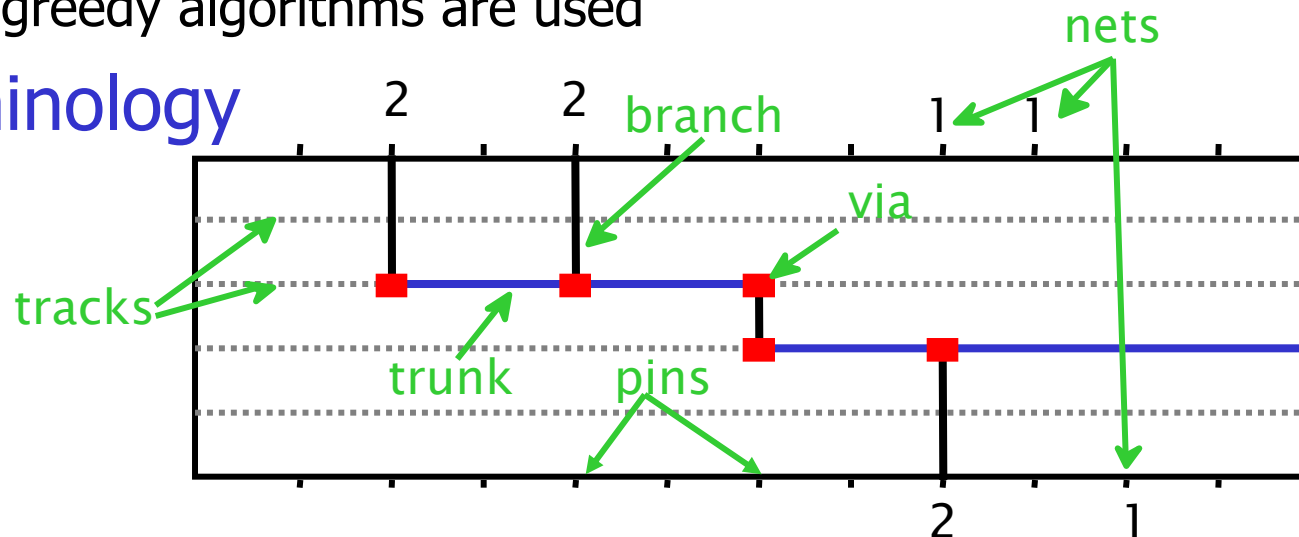
Gridless

[©Sherwani]
[©Keutzer]

Channel Routing Algorithms and Terminology

- General case is NP-Complete
- Algorithms
 - Simple case: left-edge algorithm (P)
 - General case: NP → heuristics
 - Problem defined using horizontal constraints graph and vertical constraints graph
 - Either graph-based algorithms or greedy algorithms are used

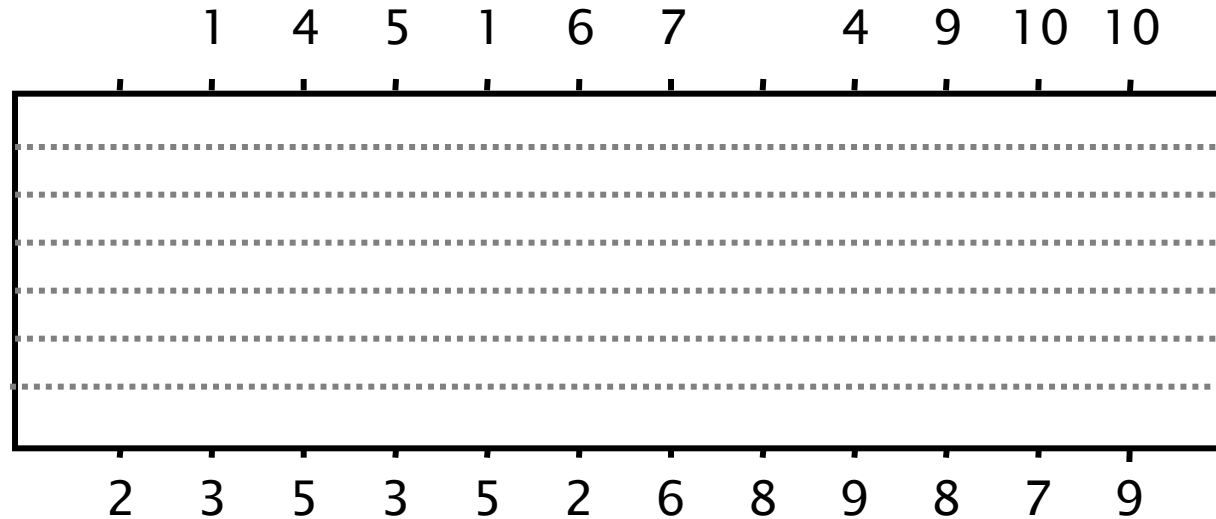
- Terminology



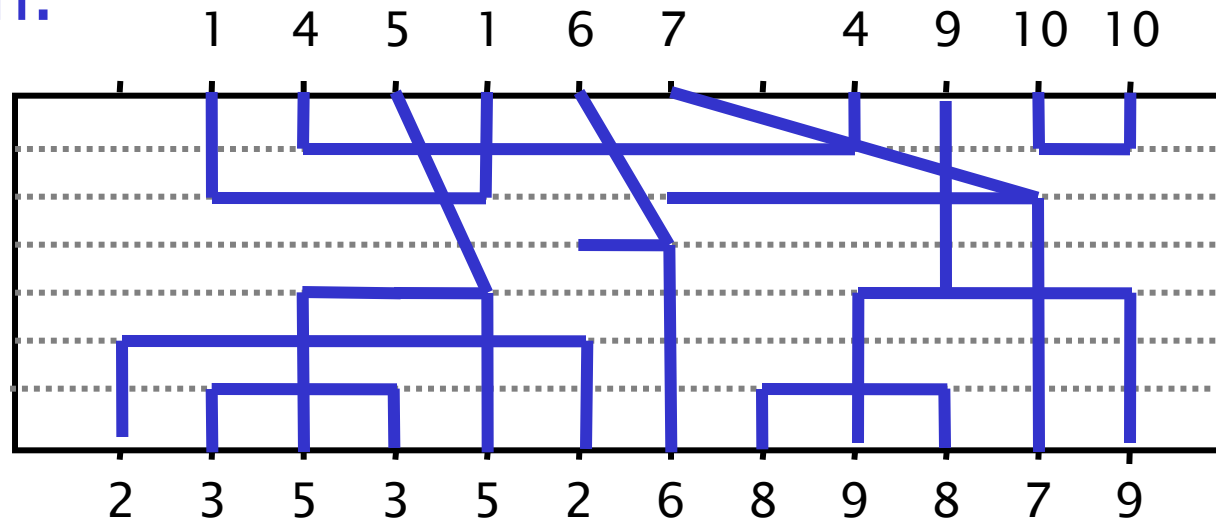
[©Keutzer]

Channel Routing Example

- Problem instance:



- Solution:

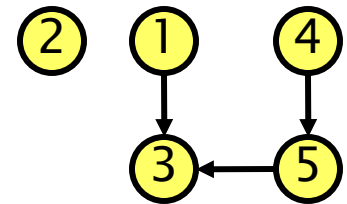
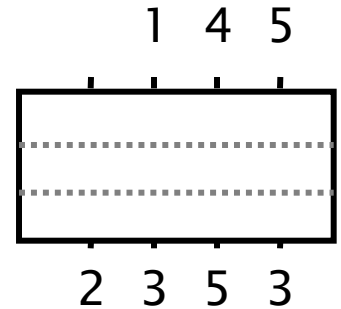


[©Keutzer]

Vertical Constraint Graph

- Represents the relative vertical positions of different net trunks (tracks)

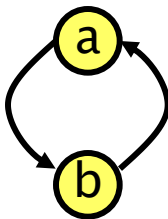
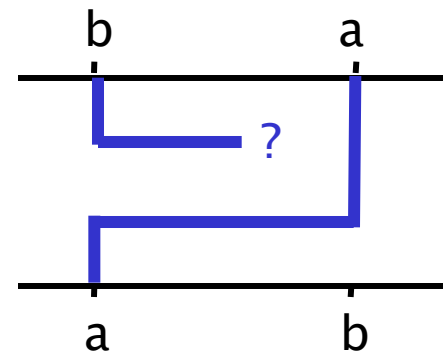
- Node: represents a net
- Edge (x, y) : if at the same column, x has a terminal on the upper edge and y has a terminal on the lower edge
- (a, b) means that net "a" has to be above "b"



- Lower bound:

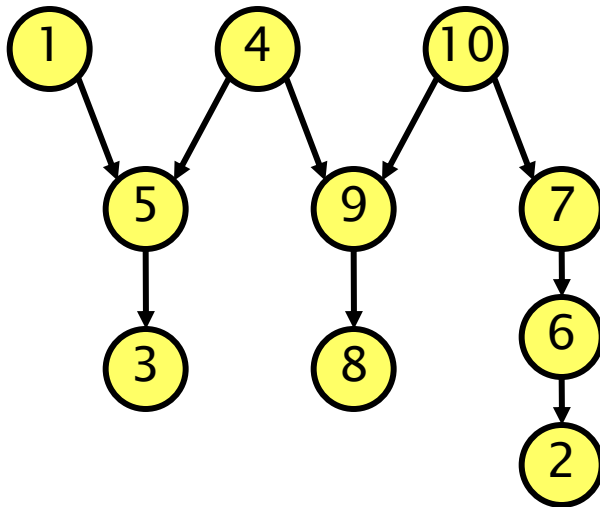
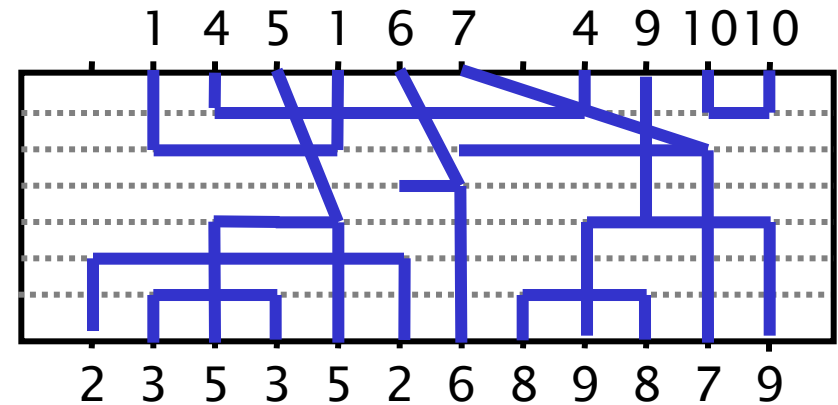
- $\# \text{ tracks} \geq \text{longest path in VCG}$

- VCG may have a cycle!

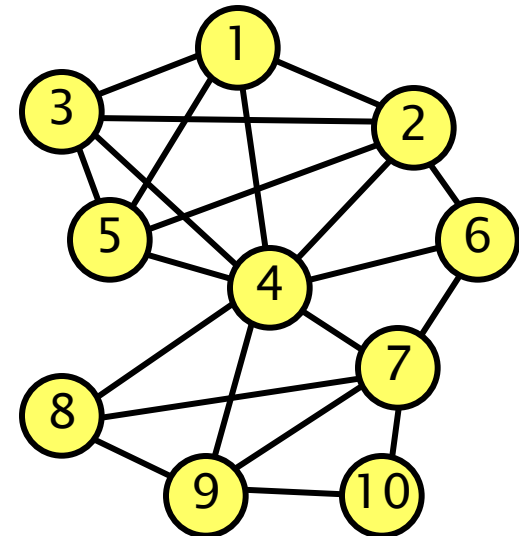


Horizontal / Vertical Constraint Graphs

- Channel routing problem can be completely characterized by VCG and HCG



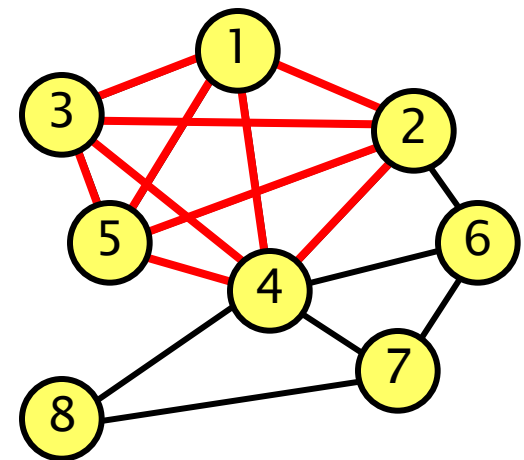
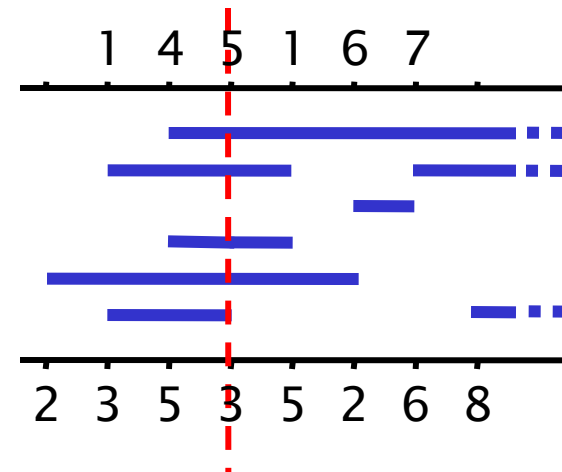
Vertical constraint graph (VCG)



Horizontal constraint graph (HCG)

Channel Density

- A net extends from its leftmost terminal to its rightmost one
- Local density at column C
 - $ld(C) = \#$ nets split by column C
- Channel density
 - $d = \max ld(c)$ over all C
- Relationship to HCG?
 - Local density \Leftrightarrow clique in HCG
 - $d \Leftrightarrow$ size of maximum clique in HCG
- Lower bound:
 - $\#$ tracks $\geq d$

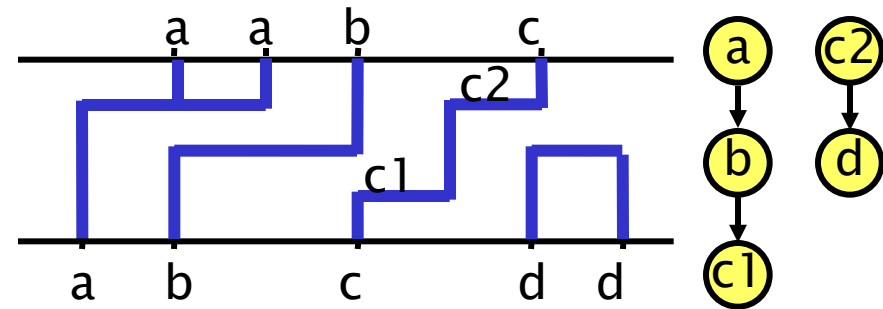
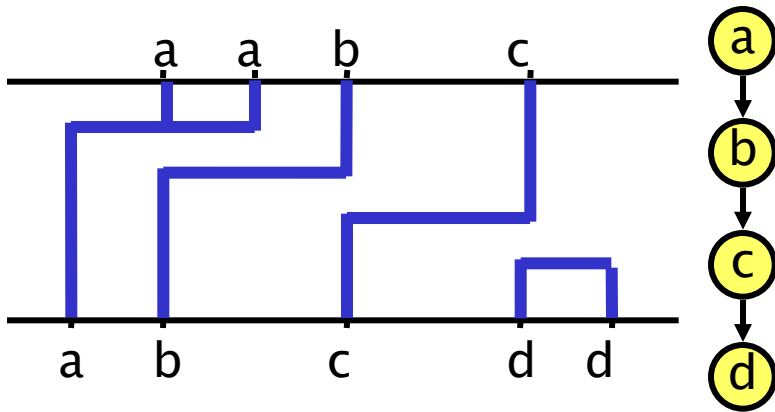


Left-edge Channel Routing Algorithm

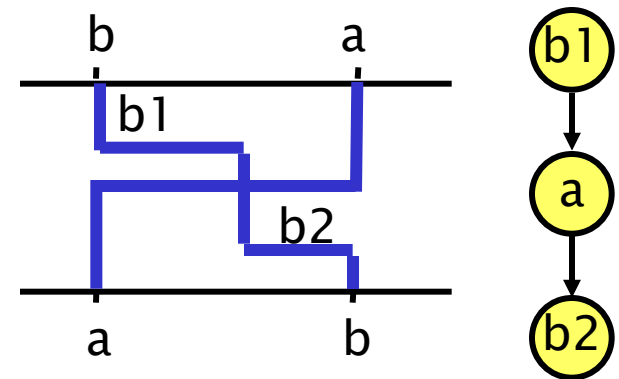
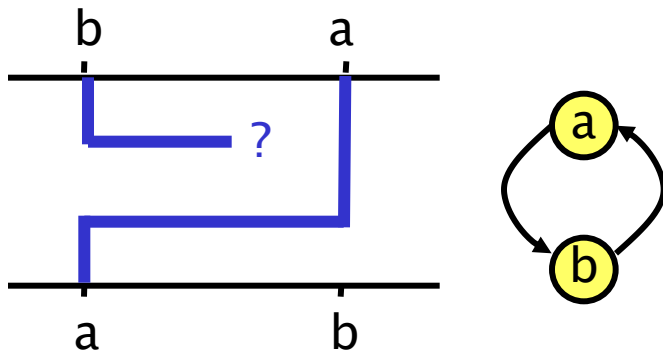
- Used when no VCG edges
- Finds the optimal solution (# tracks = d)
- Nets are sorted according to their left endpoints
- Algorithm:
 - Create an initial track t
 - For all nets n_i in the order of their left endpoints
 - if feasible to place the net on an existing track t_j ,
assign net n_i to track t_j .
 - else create a new track t_{new} and assign n_i to it.
- Time complexity: $O(n \log n)$

Doglegs

- Doglegs can reduce the longest path in VCG



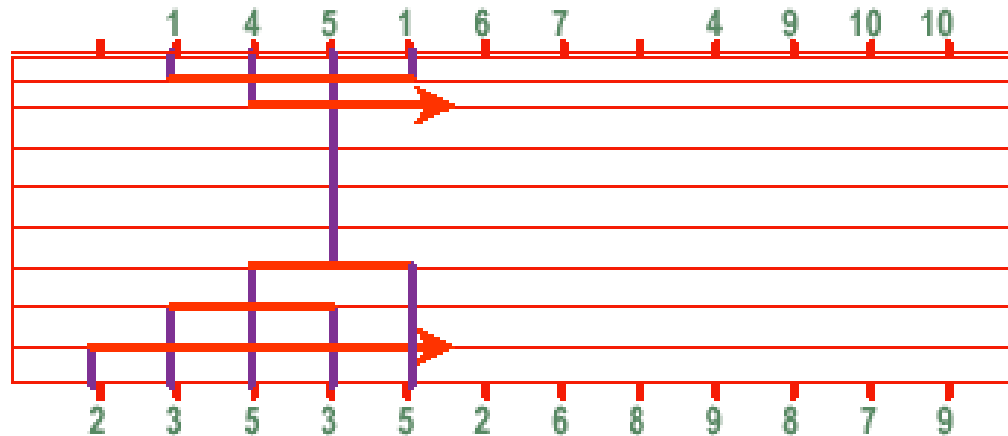
- Doglegs break cycles in VCG



Greedy Channel Router

- Many greedy algorithms for channel routing exist
- Example: Rivest and Fiduccia DAC'82
 - Simple, linear algorithm
 - Guarantees routing of all nets
 - Uses doglegs (both restricted and unrestricted)
 - BUT may extend to right hand side of the channel
- Other techniques
 - Hierarchical: divide the channel into two smaller channels, route each small channel, merge
 - VCG reduction: nets that can be placed on the same track merged into one VCG node to reduce VCG size

Greedy Router: Rivest and Fiduccia

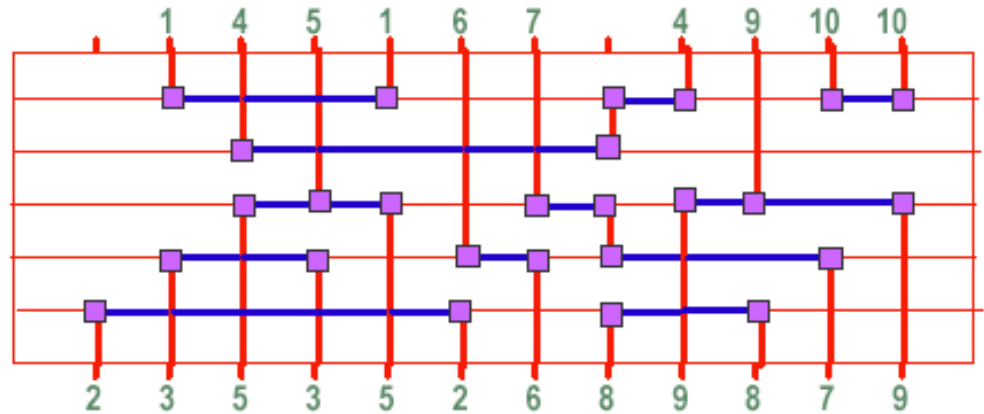


- Proceed column by column (left to right)
- Make connections to all pins in that column
- Free up tracks by collapsing as many tracks as possible to collapse nets
- Shrink range of rows occupied by a net by using doglegs
- If a pin cannot enter a channel, add a track
- $O(\text{pins})$ time

[©Keutzer]

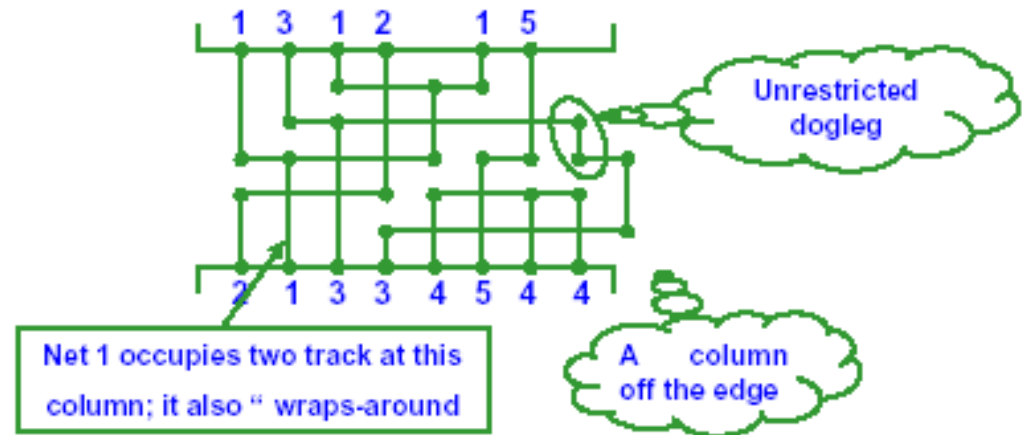
Rivest and Fiduccia: Example

- Example output:



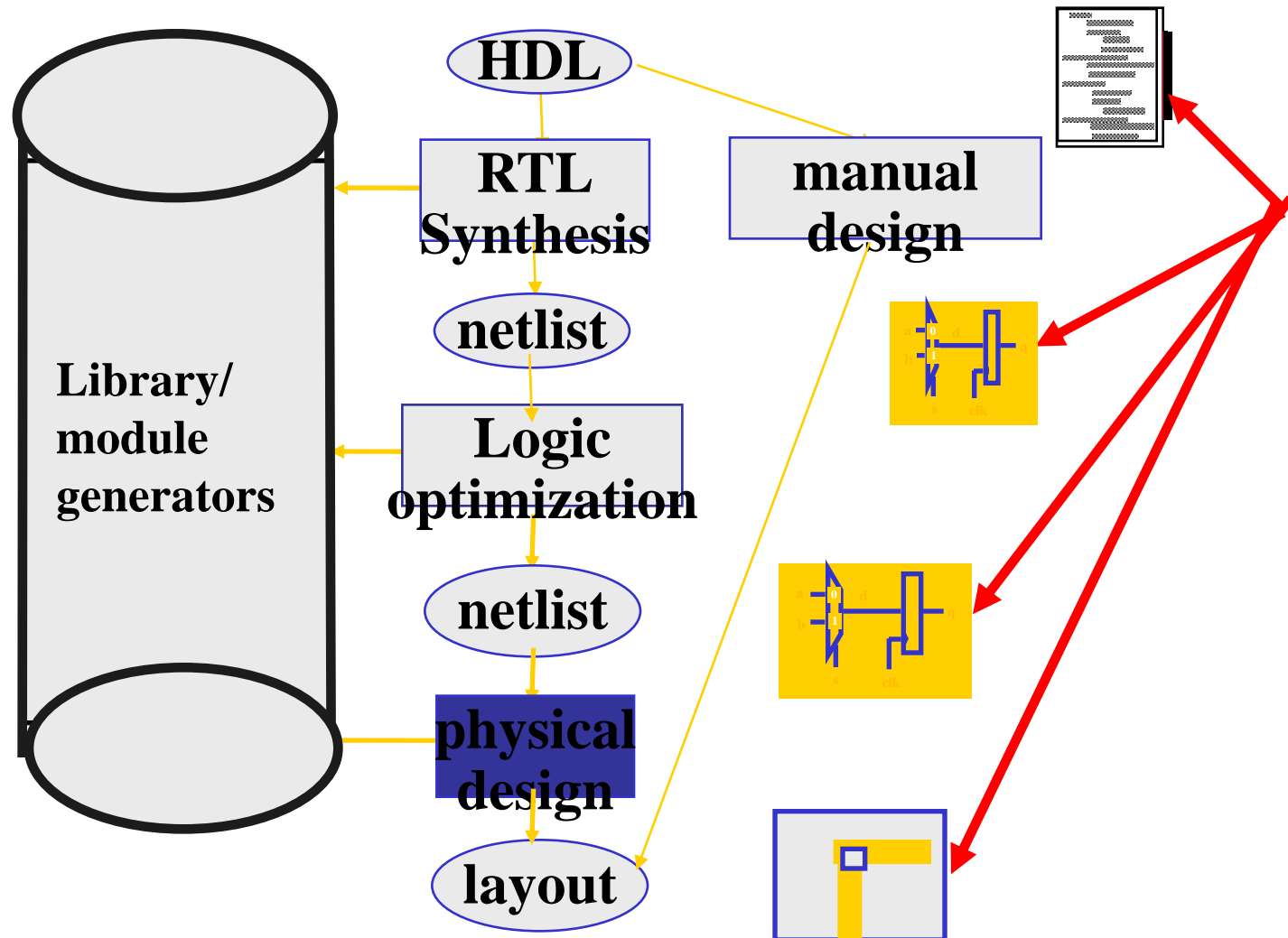
- Observations:

- Always succeeds (even if cyclic conflict is present)
- Allows unrestricted doglegs
- Allows a net to occupy more than 1 track at a given column
- May use a few columns off the edge



[©Keutzer]

RTL design flow



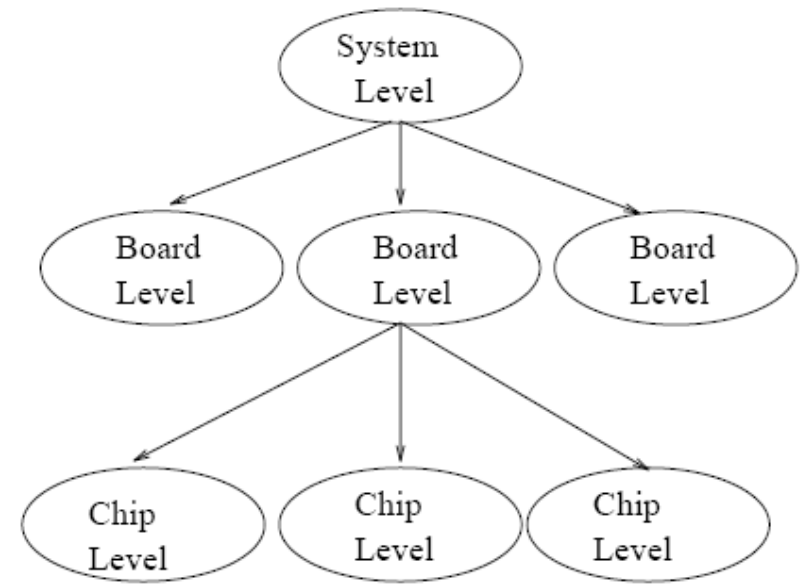
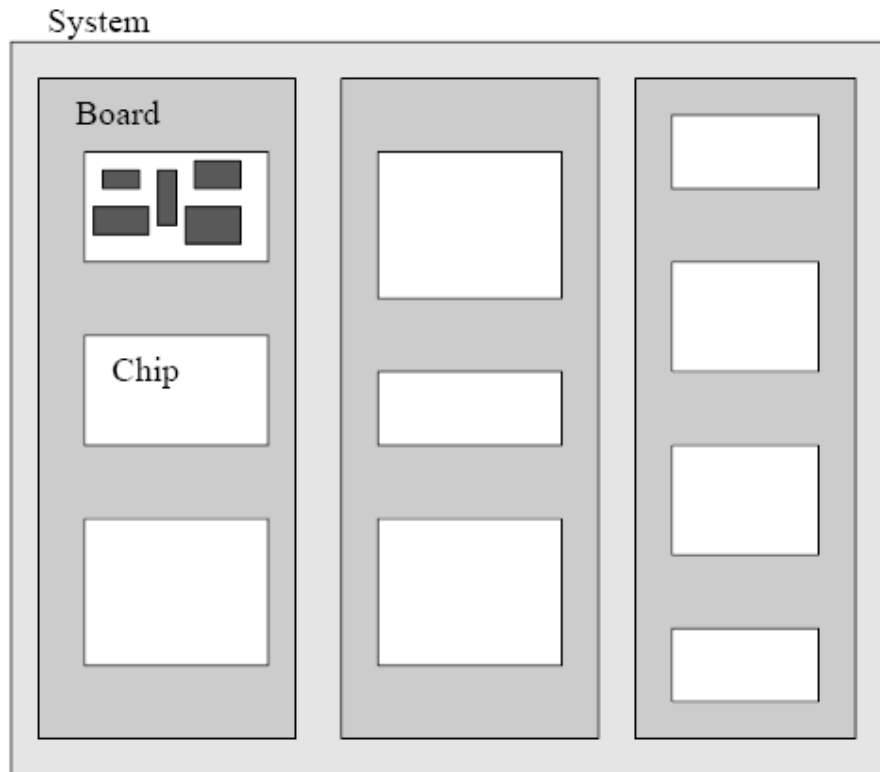
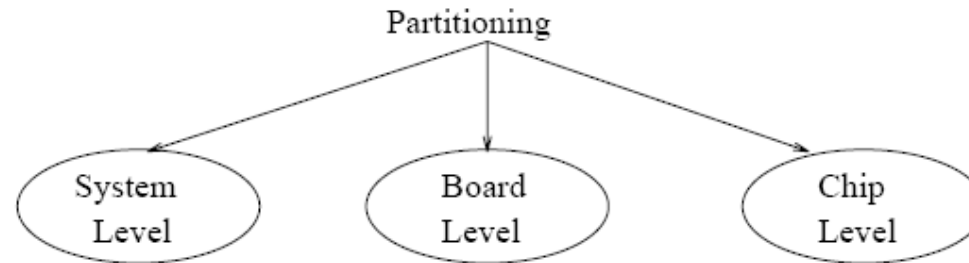
Physical design – overall flow



Partitioning

- ❑ Decompose a large complex system into smaller subsystems
- ❑ Decompose hierarchically until each subsystem is of manageable size
- ❑ Design each subsystem separately to speed up the process
- ❑ Minimize connection between two subsystems to reduce interdependency

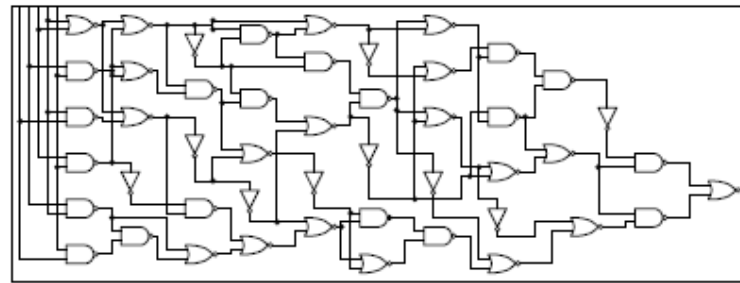
Partitioning at different levels*



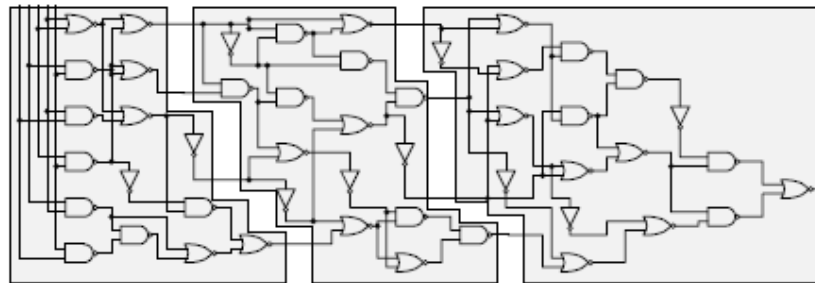
* © Sherwani

Partitioning example*

Input size = 48



(a)



(b)

Cut 1 = 4

Cut 2 = 4

Size 1 = 15

Size 2 = 16

Size 3 = 17

* © Sherwani

Partitioning problem

❑ Objective:

- Minimize interconnections between partitions
- Minimize delay due to partitioning

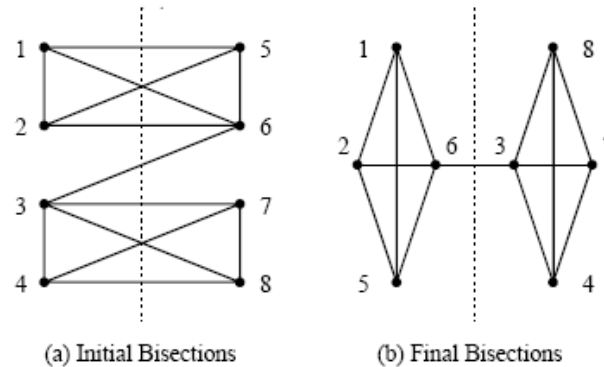
❑ Constraints

- Number of terminals in each subsystem ($\text{Count}(V_i) < T_i$)
- Area of each partition ($A_i^{\min} < \text{Area}(V_i) < A_i^{\max}$)
- Number of partitions ($K_{\min} < k < K_{\max}$)
- Critical path should not cut boundaries

Kernighan-Lin algorithm

- ❑ Input: Graph representation of the circuit
- ❑ Output: Two subsets of equal sizes
- ❑ Bisecting algorithm :
 - Initial bisection
 - Vertex pairs which gives the largest decrease in cutsize are exchanged
 - Exchanged vertices are locked
 - If no improvement is possible and some vertices are still unlocked then vertices which give smallest increase are exchanged

K-L algorithm example*

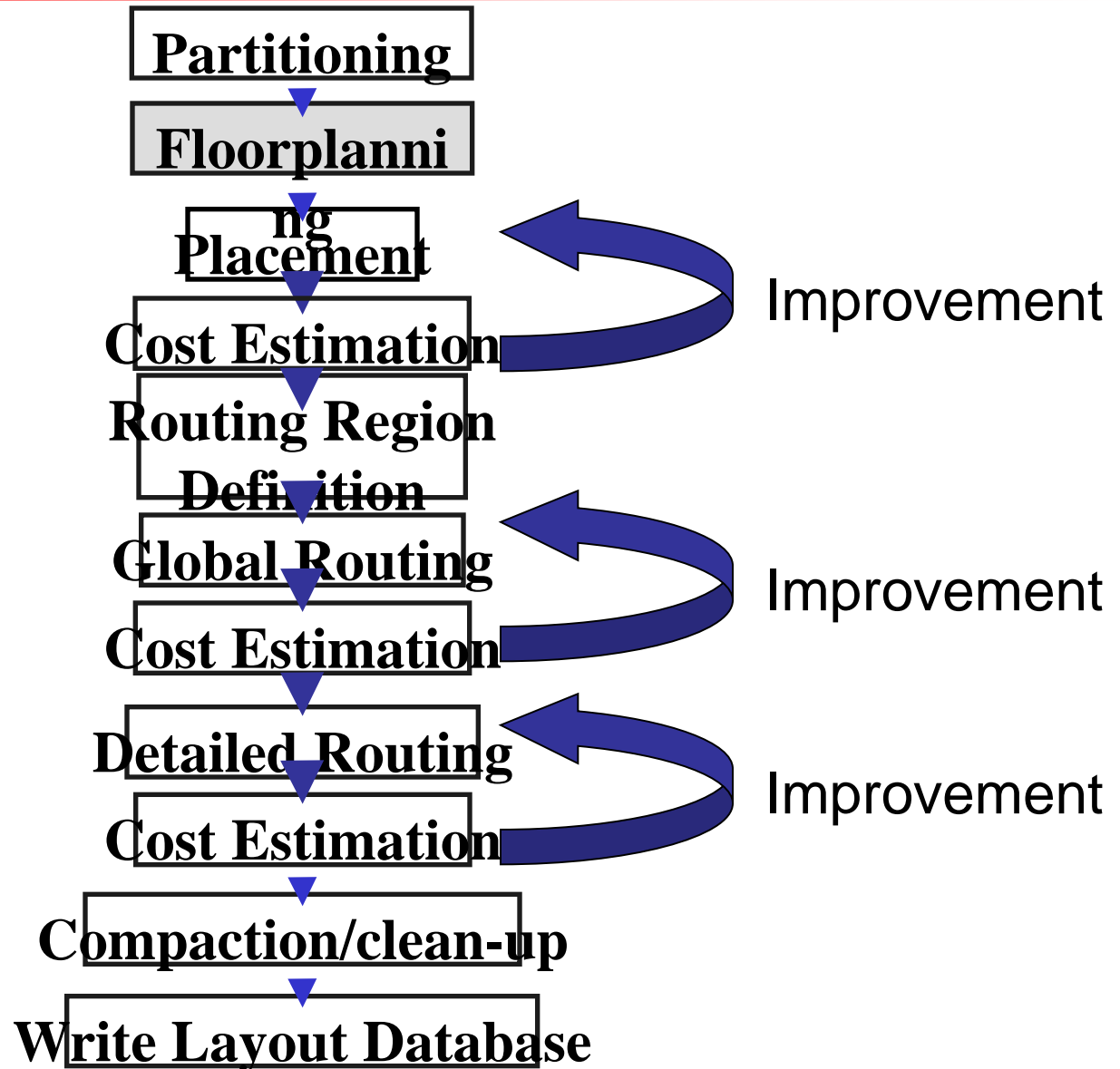


i	Vertex Pair	$g(i)$	$\sum_{j=1}^i g(i)$	Cutsizes
0	-	-	-	9
1	(3,5)	3	3	6
2	(4,6)	5	8	1
3	(1,7)	-6	2	7
4	(2,8)	-2	0	9

Partitioning methods

- ❑ Top-down partitioning
 - Iterative improvement
 - Spectral based
 - Clustering methods
 - Network flow based
 - Analytical based
 - Multi-level
- ❑ Bottom-up clustering
 - Unit delay model
 - General delay model
 - Sequential circuits with retiming

Physical design – overall flow



Floorplanning

- ❑ Output from partitioning used for floorplanning
- ❑ Inputs:
 - Blocks with well-defined shapes and area
 - Blocks with approximated area and no particular shape
 - Netlist specifying block connections
- ❑ Outputs:
 - Locations for all blocks

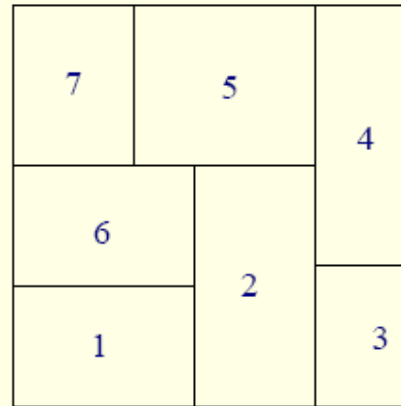
Floorplanning problem*

□ Objectives

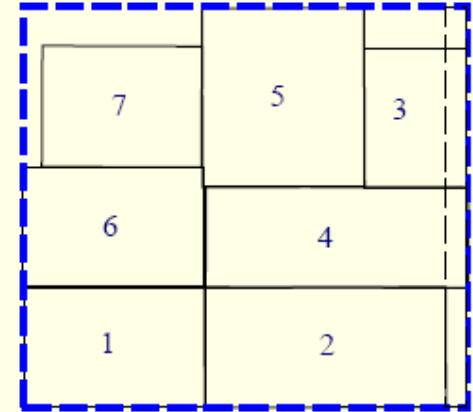
- Minimize area
- Reduce wirelength
- Maximize routability
- Determine shapes of flexible blocks

□ Constraints

- Shape of each block
- Area of each block
- Pin locations for each block
- Aspect ratio



An optimal floorplan,
in terms of area



A non-optimal floorplan

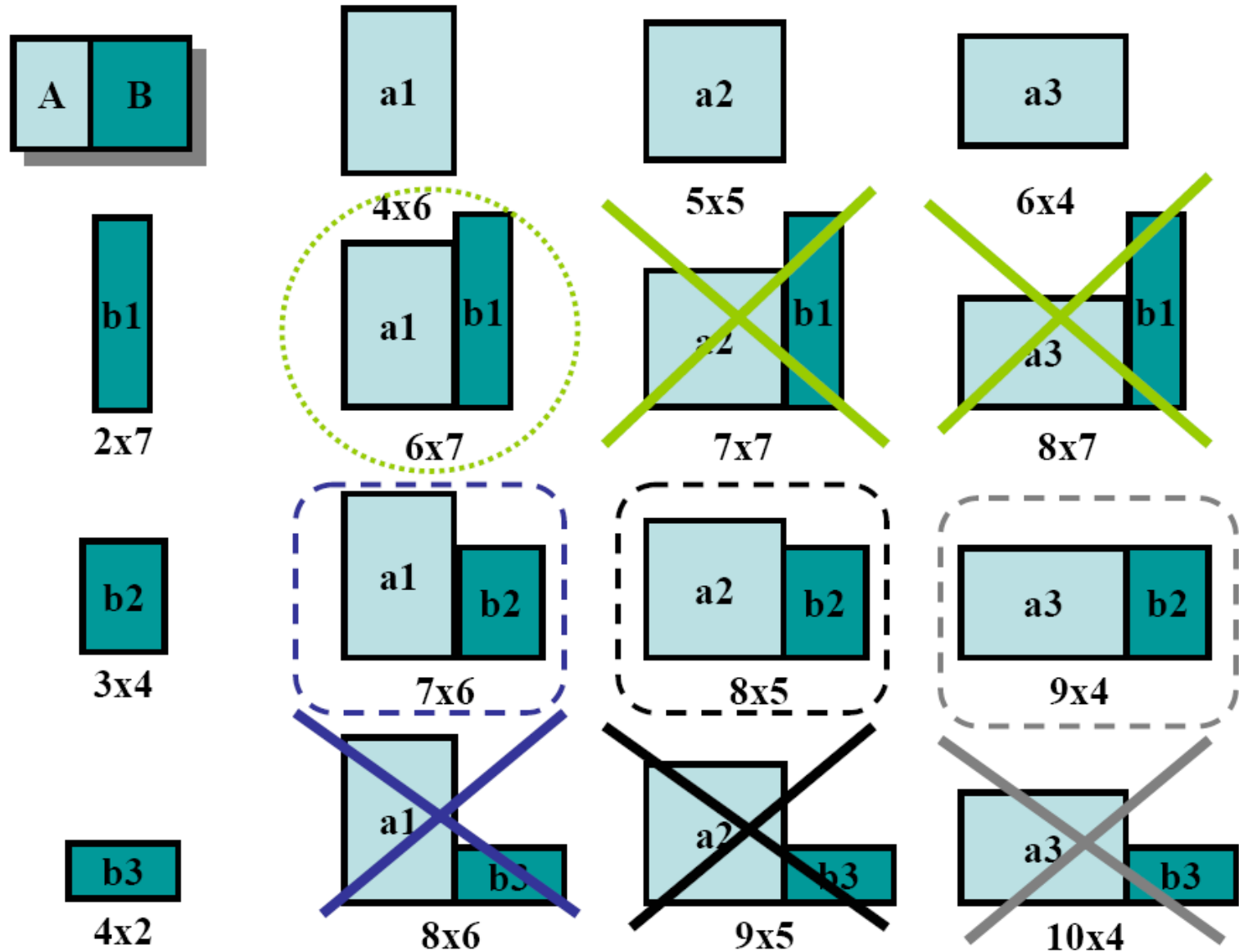
* Sung Kyu Li

Slicing floorplan sizing*

- ❑ General case: all modules are soft macros
- ❑ Phase 1: bottom-up
 - Input – floorplan tree, modules shapes
 - Start with a sorted shapes list of modules
 - Perform vertical_node_sizing and horizontal_node_sizing
 - On reaching the root node, we have a list of shapes, select the one that is best in terms of area
- ❑ Phase 2: top-down
 - Traverse the floorplan tree and set module locations

* Sung Kyu Li

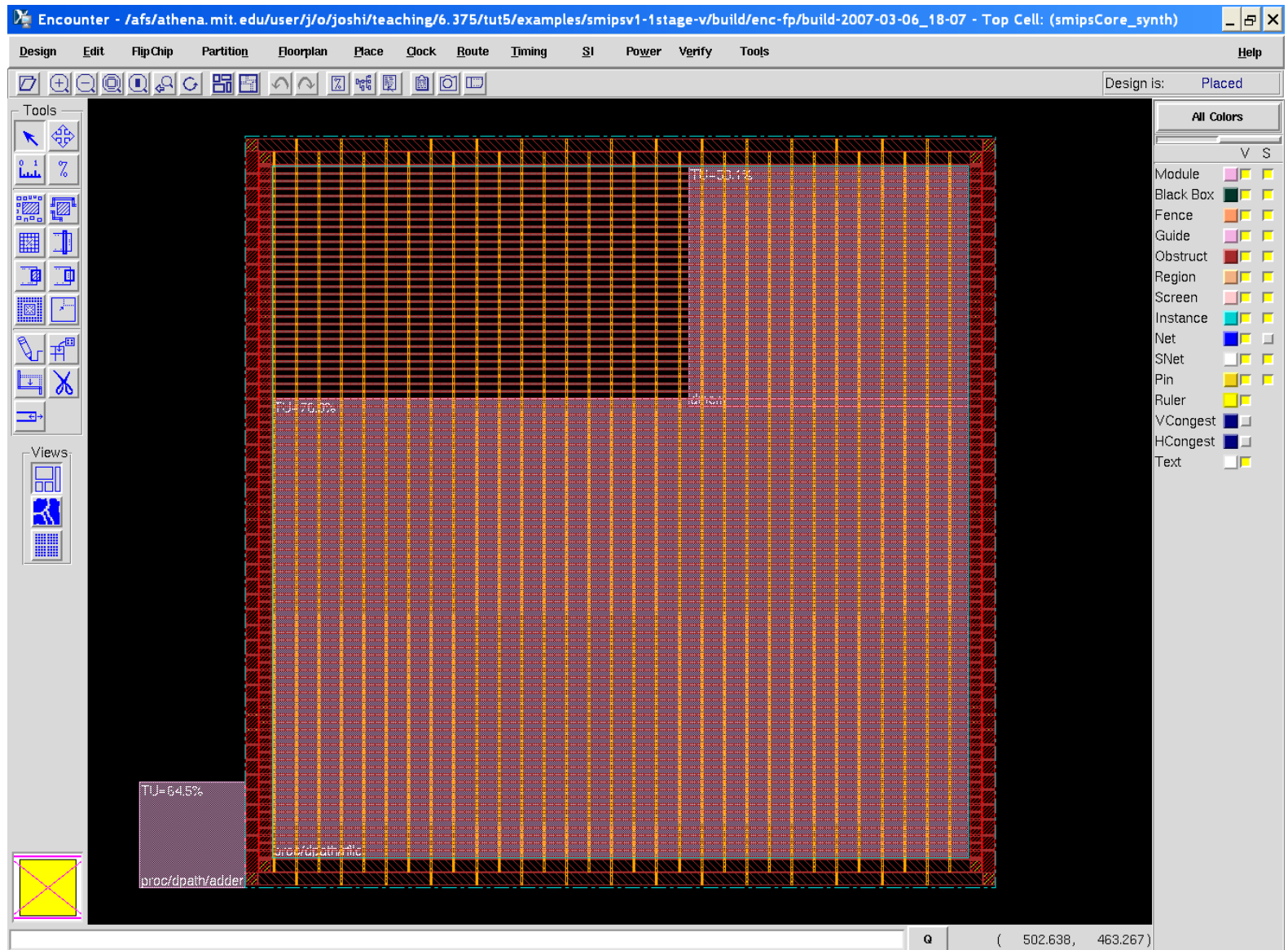
Sizing example*



Floorplanning Algorithms

- ❑ Stockmeyer algorithm
- ❑ Simulated annealing
- ❑ Linear programming
- ❑ Sequence-pair based floorplanning

Floorplanning - Encounter



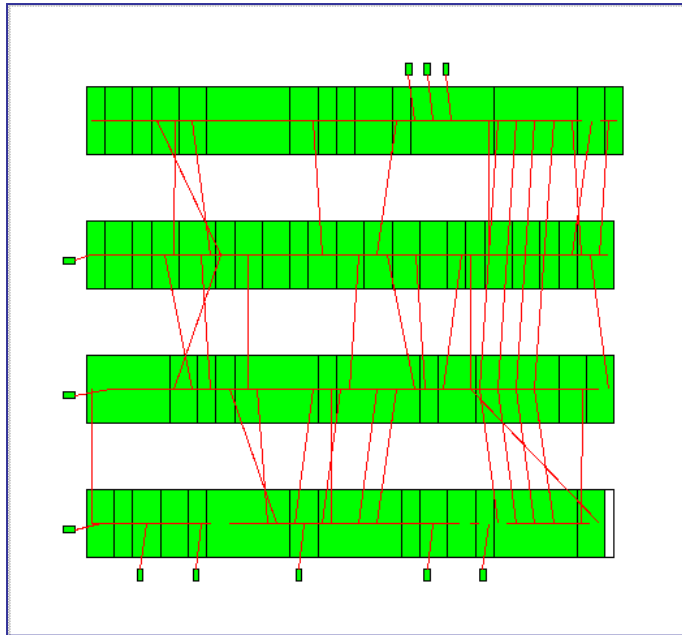
Physical design – overall flow



Placement

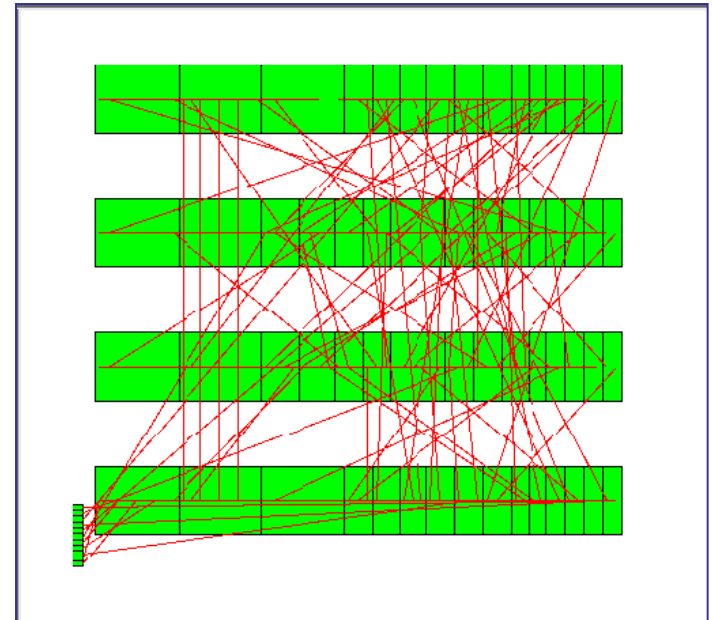
- ❑ The process of arranging circuit components on a layout surface
- ❑ Inputs : Set of fixed modules, netlist
- ❑ Output : Best position for each module based on various cost functions
- ❑ Cost functions include wirelength, wire routability, hotspots, performance, I/O pads

Good placement vs Bad placement*



□ Good placement

- No congestion
- Shorter wires
- Less metal levels
- Smaller delay
- Lower power dissipation



□ Bad placement

- Congestion
- Longer wire lengths
- More metal levels
- Longer delay
- Higher power dissipation

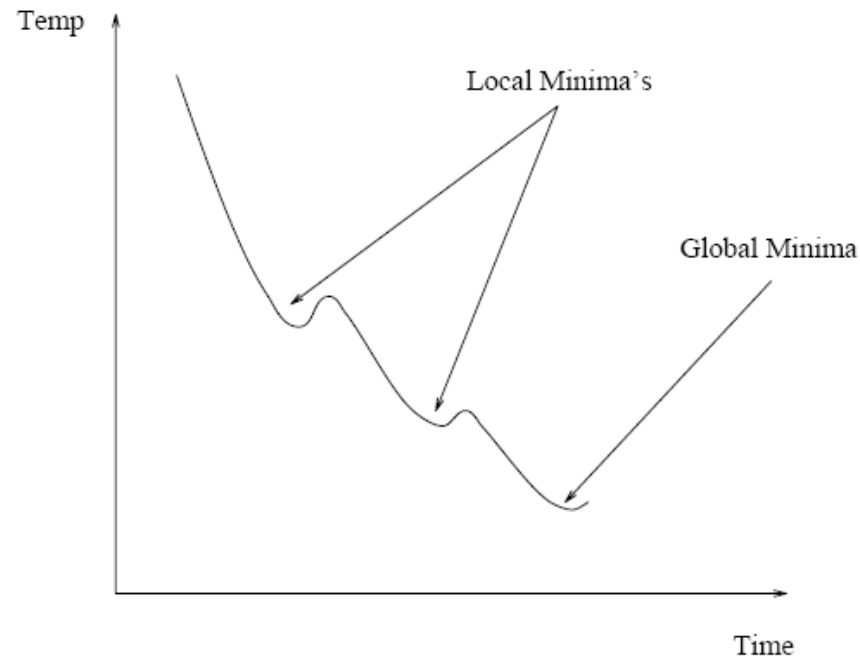
* S. Devadas

Simulated annealing algorithm

- ❑ Global optimization technique
- ❑ Cooling schedule is adopted
- ❑ An action performed at each new temperature
- ❑ Estimate the cost associated with an action
- ❑ If new cost $<$ old cost accept the action
- ❑ If new cost $>$ old cost then accept the action with probability p
- ❑ Probability p depends on a temperature schedule – Higher p at higher temperature

Annealing curve

The Annealing curve



Placement using simulated annealing

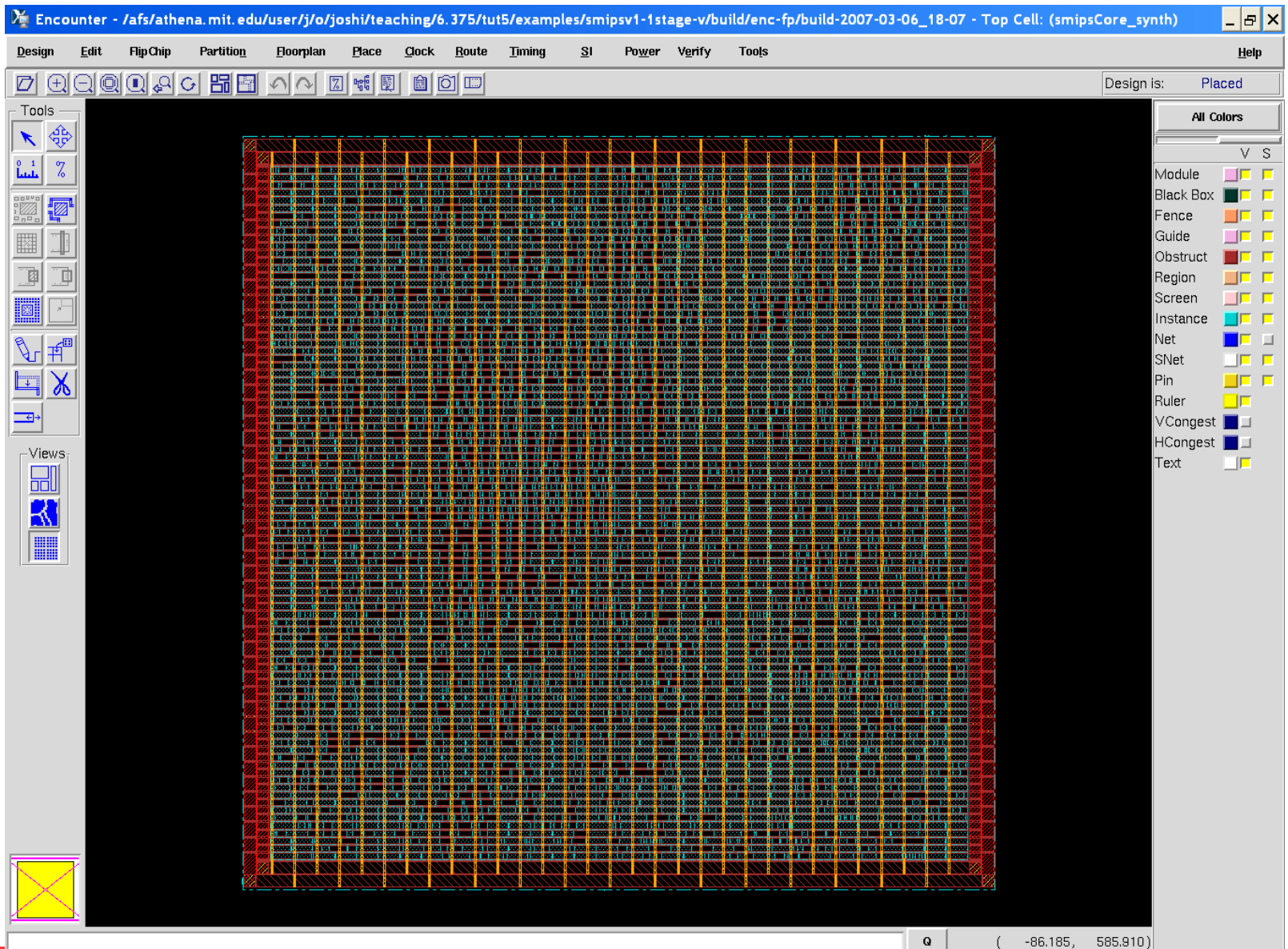
- ❑ Use initial placement results – e.g. random placement
- ❑ Two stage process*
 - Stage 1
 - Modules moved between different rows and same row
 - Module overlaps allowed
 - Stage two begins when temperature falls below a certain value
 - Stage 2
 - Module overlaps removed
 - Annealing continued, but interchange adjacent modules in the same row

* Sechen DAC

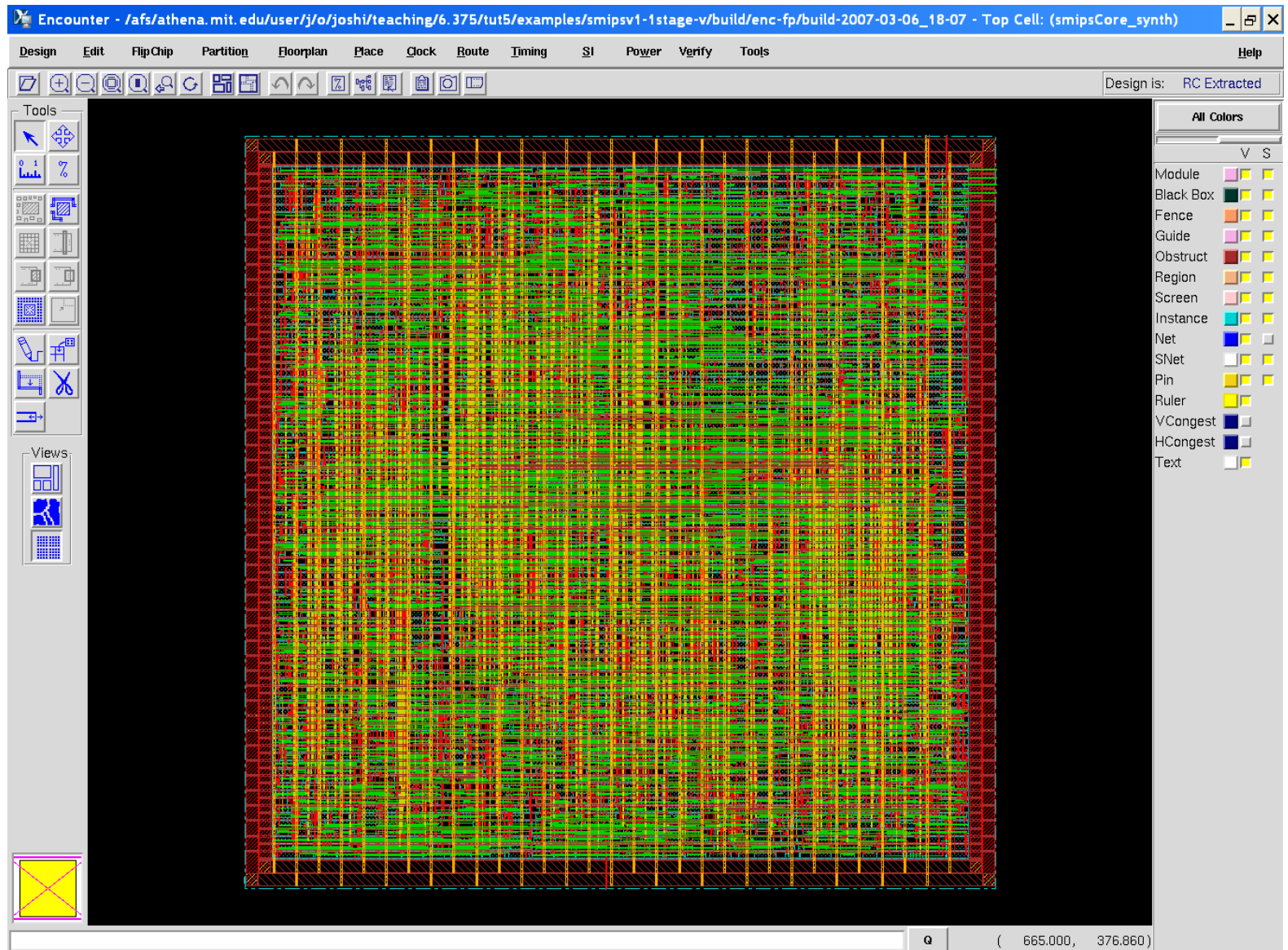
Placement methods

- ❑ Constructive methods
 - Cluster growth algorithm
 - Force-directed method
 - Algorithm by Goto
 - Min-cut based algorithm
- ❑ Iterative improvement methods
 - Pairwise exchange
 - Simulated annealing – Timberwolf
 - Genetic algorithm
- ❑ Analytical methods
 - Gordian, Gordian-L

Placement - Encounter



Optimized placement - Encounter



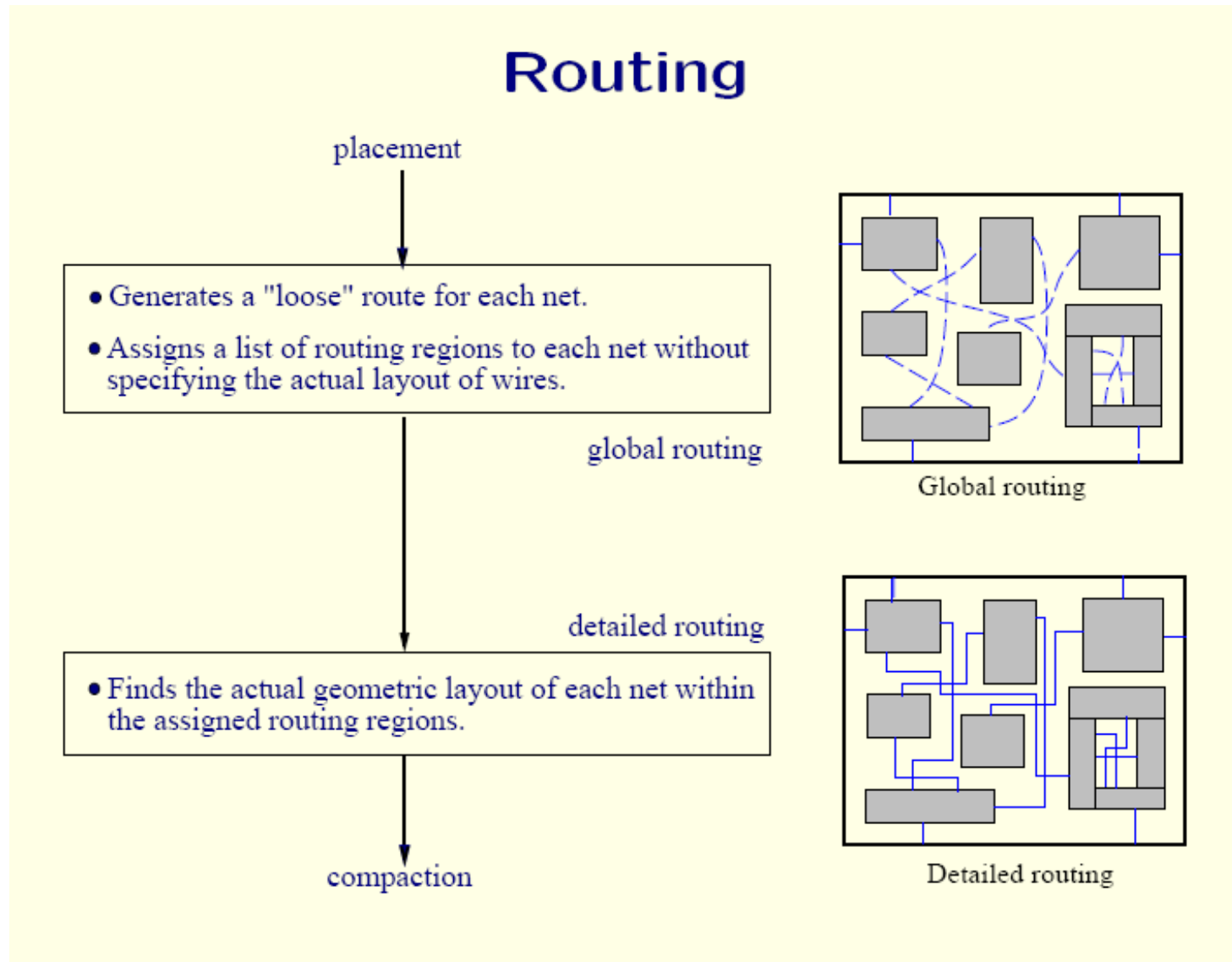
Physical design – overall flow



Routing

- ❑ Connect the various standard cells using wires
- ❑ Input:
 - Cell locations, netlist
- ❑ Output:
 - Geometric layout of each net connecting various standard cells
- ❑ Two-step process
 - Global routing
 - Detailed routing

Global routing vs detailed routing*



* Sung Kyu Li

Routing problem formulation

❑ Objective

- 100% connectivity of a system
- Minimize area
- Minimize wirelength

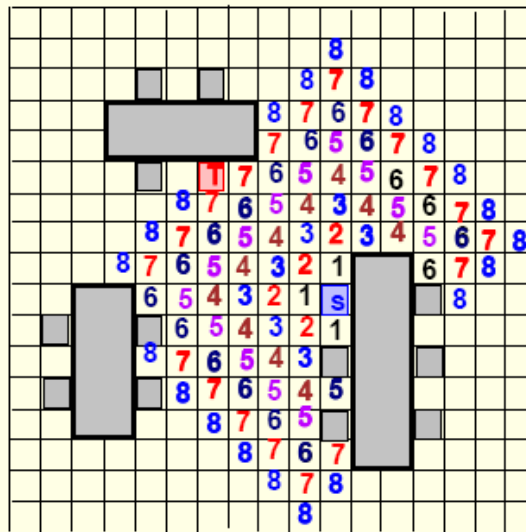
❑ Constraints

- Number of routing layers
- Design rules
- Timing (delay)
- Crosstalk
- Process variations

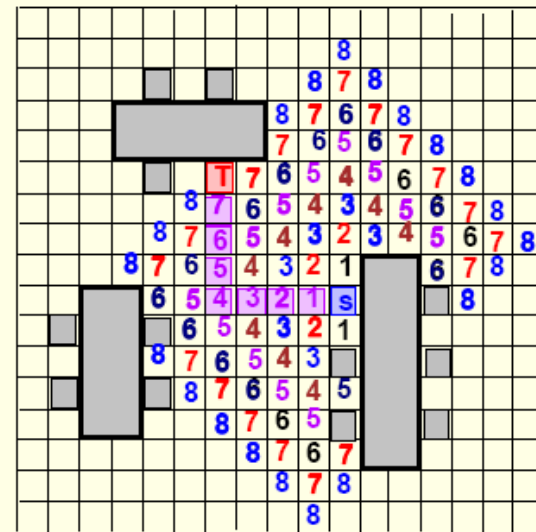
Maze routing - example

Lee Algorithm

- Find a path from S to T by “wave propagation”.



Filing



Retrace

- Time & space complexity for an $M \times N$ grid: $O(MN)$ (**huge!**)

Maze routing

- ❑ Mainly for single-layer routing
- ❑ Strengths
 - Finds a connection between two terminals if it exists
- ❑ Weakness
 - Large memory required as dense layout
 - Slow
- ❑ Application – global routing, detailed routing

Routing algorithms

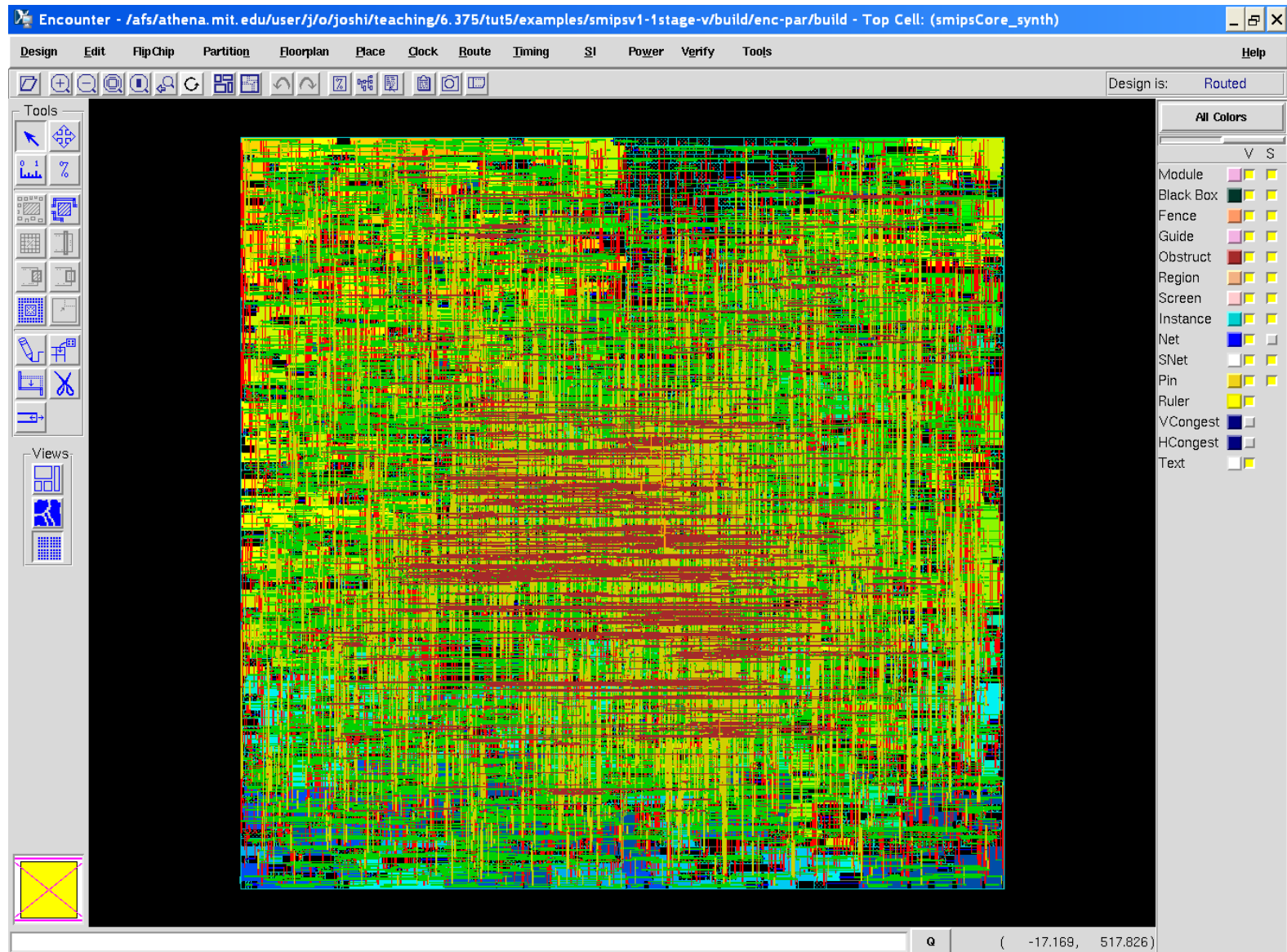
❑ Global routing

- Maze routing
- Cong/Prea's algorithm
- Spanning tree algorithm
- Steiner tree algorithm

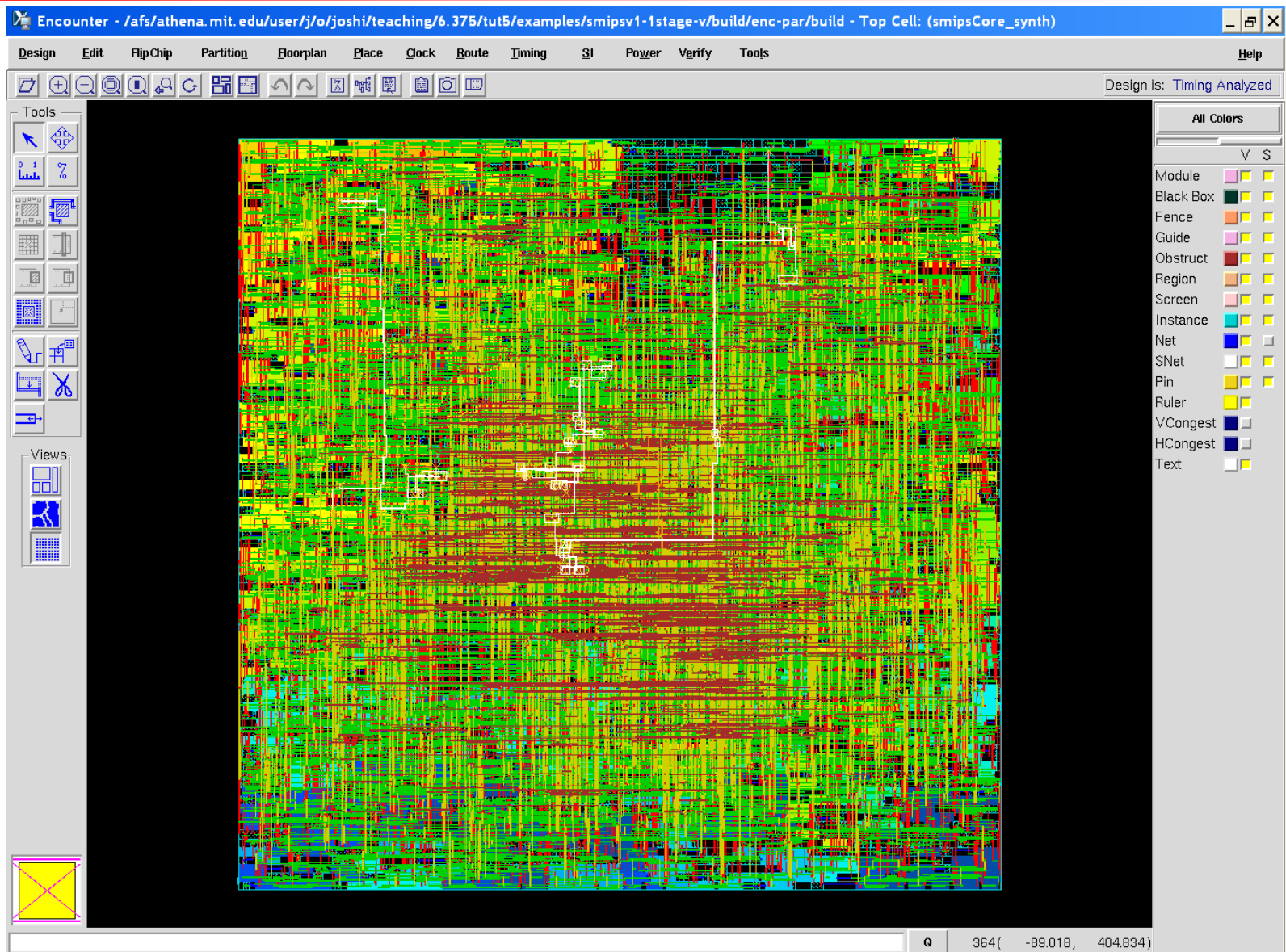
❑ Detailed routing

- 2-L Channel routing: Basic left-edge algorithm
- Y-K algorithm

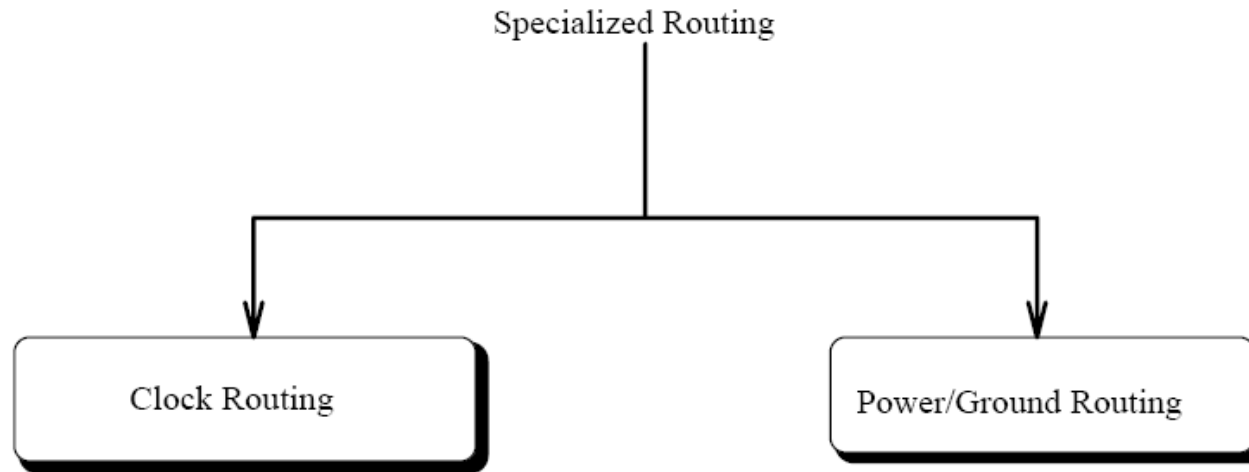
Detailed routing - Encounter



Critical path - Encounter



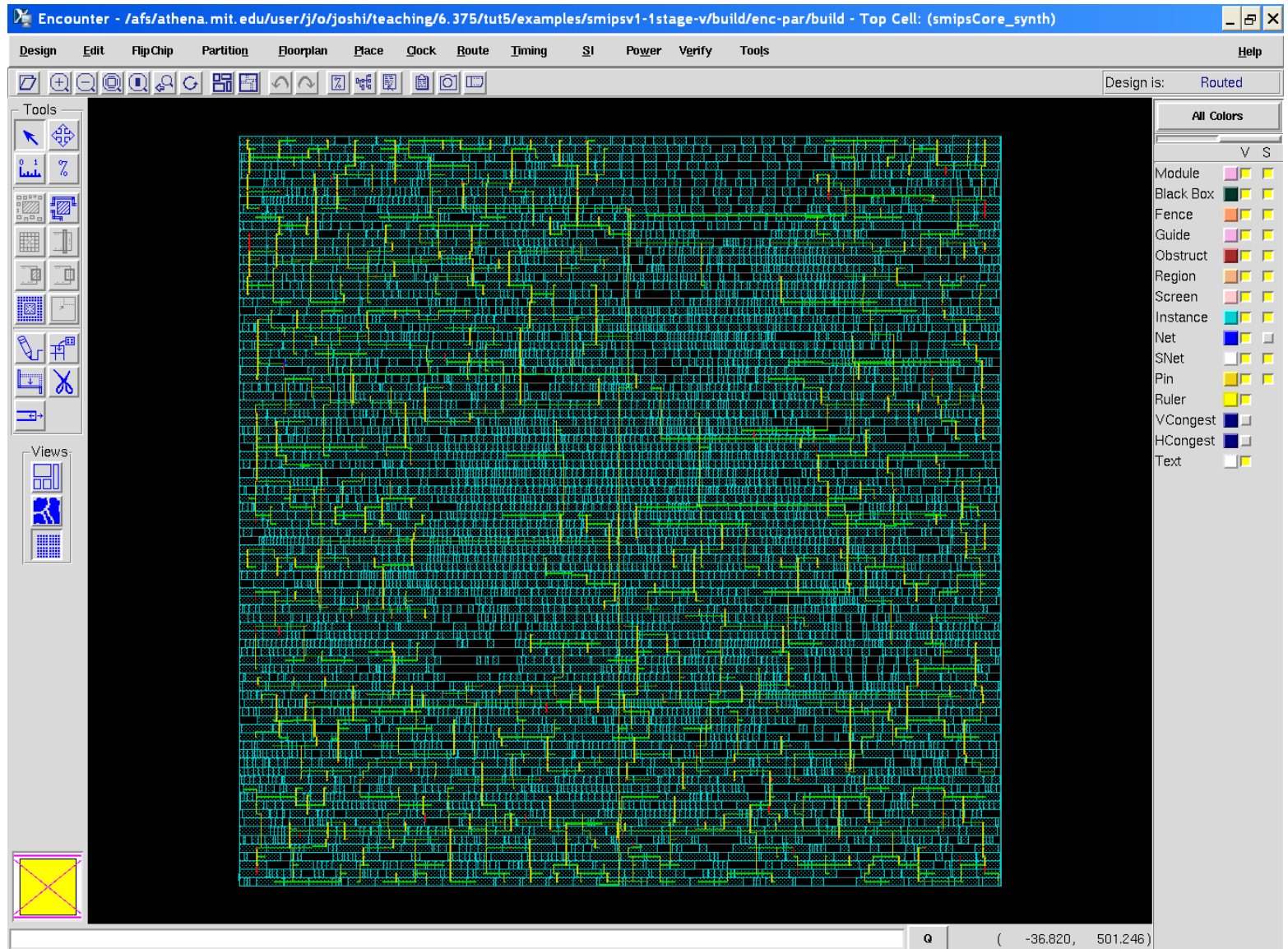
Specialized routing



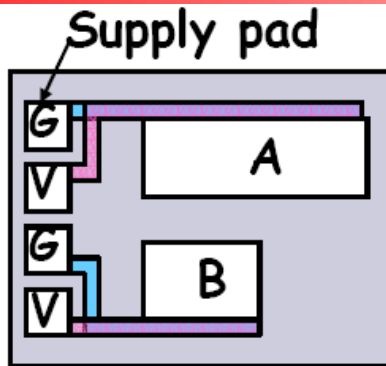
- Routing clock nets such that
 - clock arrives simultaneously
 - clock delay is minimum

- Routing of power/ground net on
 - Low resistance metal lines

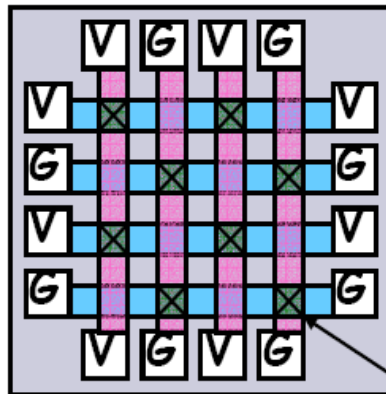
Clock distribution - Encounter



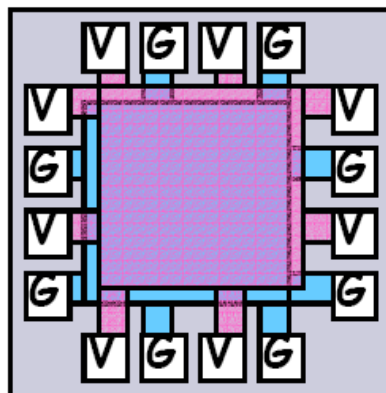
Power routing*



Routed power distribution on two stacked layers of metal (one for VDD, one for GND). OK for low-cost, low-power designs with few layers of metal.



Power Grid. Interconnected vertical and horizontal power bars. Common on most high-performance designs. Often well over half of total metal on upper thicker layers used for VDD/GND.

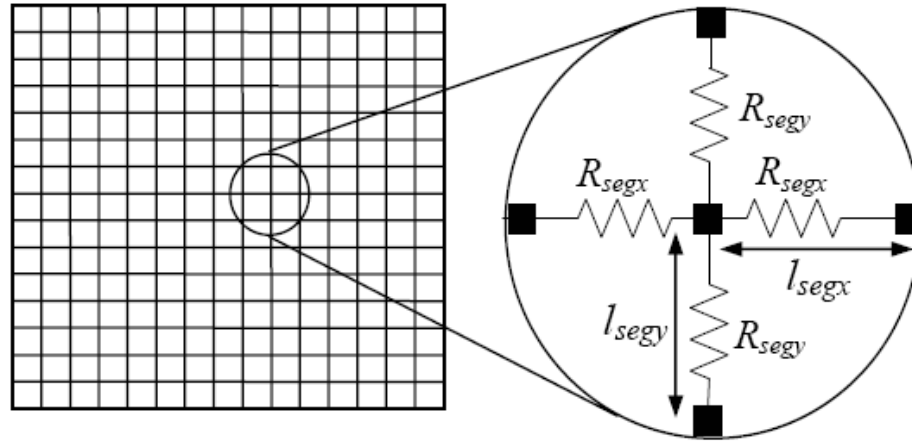


Via

Dedicated VDD/GND planes. Very expensive. Only used on Alpha 21264. Simplified circuit analysis. Dropped on subsequent Alphas.

*6.884 Spring

Power distribution issues



Kaveh Shakeri

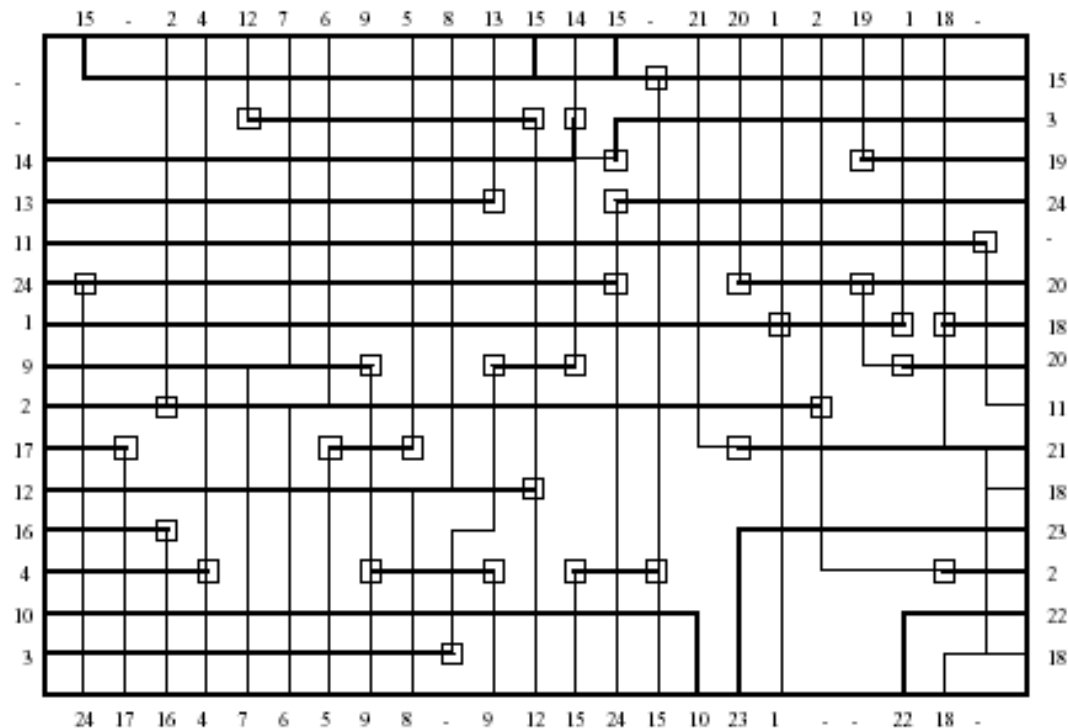
- ❑ Noise in power supply
 - IR drop (static)
 - L di/dt (transient)
- ❑ Electromigration
- ❑ Solution: decoupling capacitance, wire material

Summary

- ❑ Looked at the physical design flow
- ❑ Involved several steps
 - Partitioning
 - Floorplanning
 - Placement
 - Routing
- ❑ Each step can be formulated as an optimization problem
- ❑ Need to go through 2 or more iterations in each step to generate an optimized solution

Switchbox Routing




- Much harder problem than channel routing
 - Two-dimensional problem
(channel routing was in a way one dimensional)
 - Need to solve in a hierarchical flow
(split a channel into two, route one first, and route the second as a switchbox)
- A number of complex heuristic algorithms exist

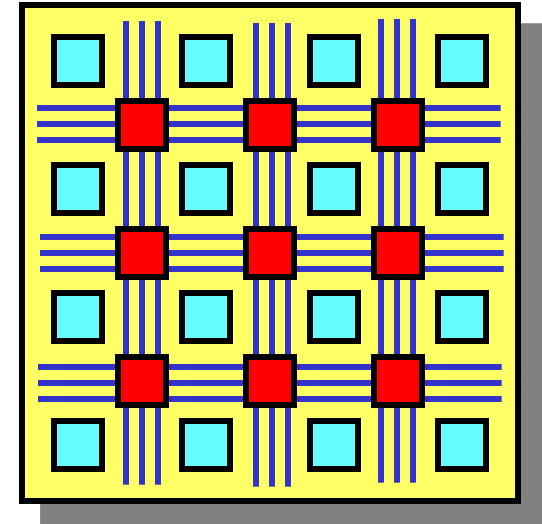


[©Sarrafzadeh]

FPGA Architecture - Layout

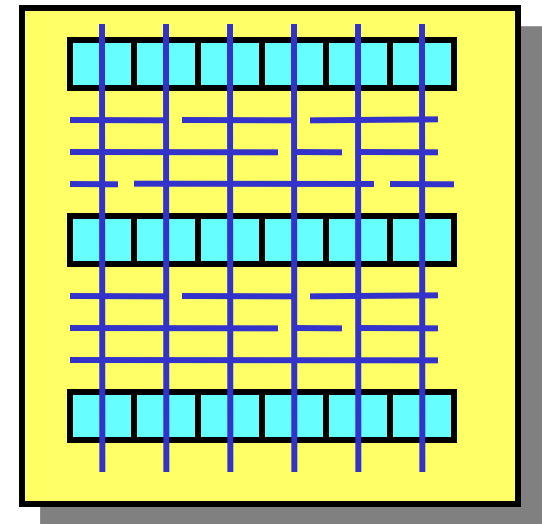
- Island FPGAs

- Array of functional units 
- Horizontal and vertical routing channels connecting the functional units 
- Versatile switch boxes 
- Example: Xilinx, Altera



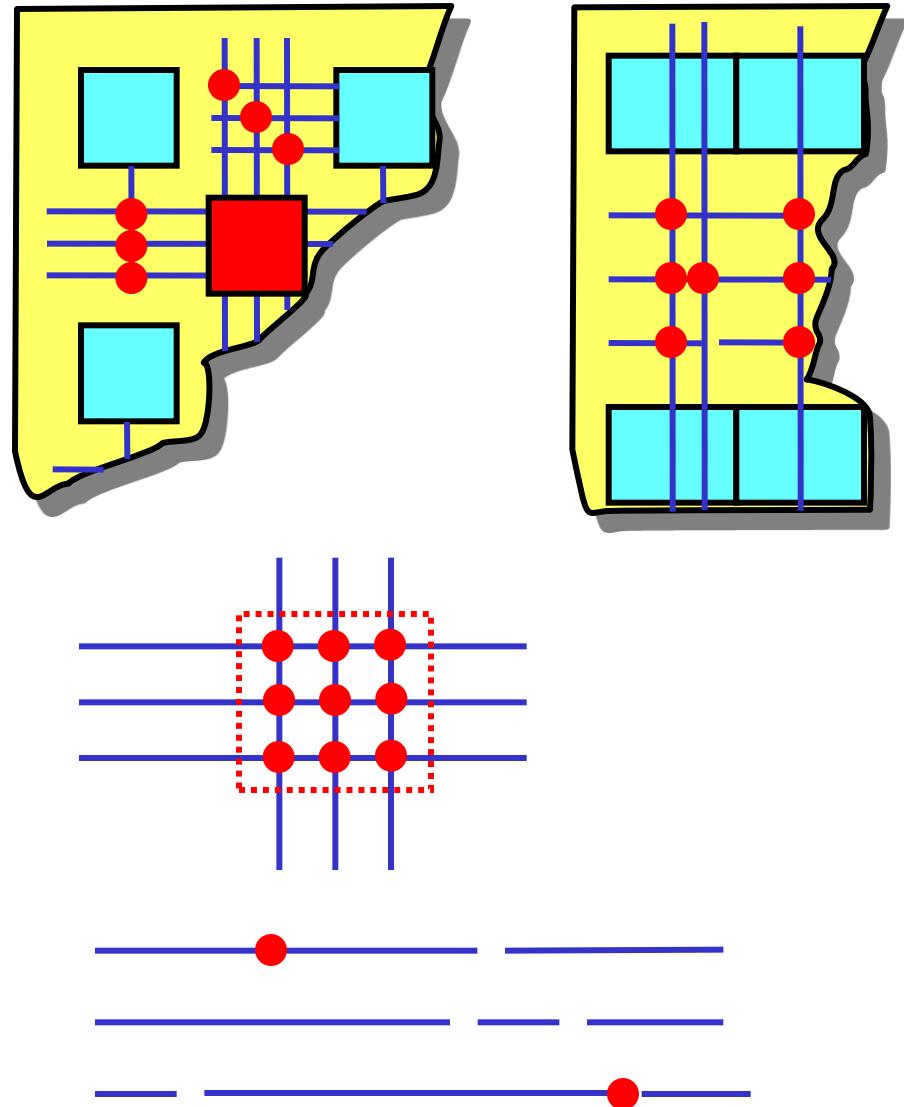
- Row-based FPGAs

- Like standard cell design
- Rows of logic blocks
- Routing channels (fixed width) between rows of logic
- Example: Actel FPGAs



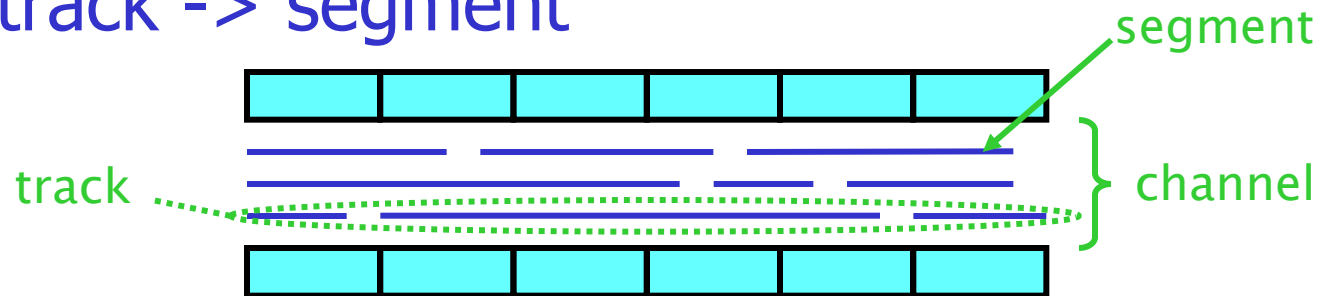
FPGA Programmable Switch Elements

- Used in connecting:
 - The I/O of functional units to the wires
 - A horizontal wire to a vertical wire
 - Two wire segments to form a longer wire segment

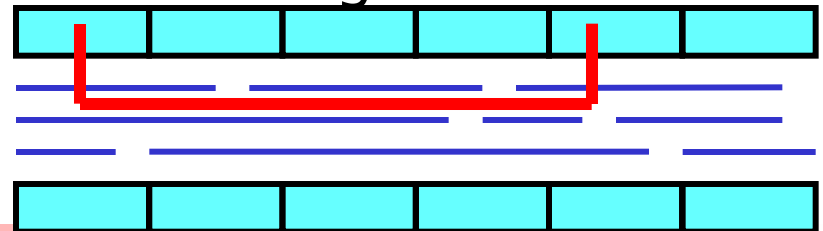


FPGA Routing Channels Architecture

- Note: fixed channel widths (tracks)
- Should “predict” all possible connectivity requirements when designing the FPGA chip
- Channel -> track -> segment

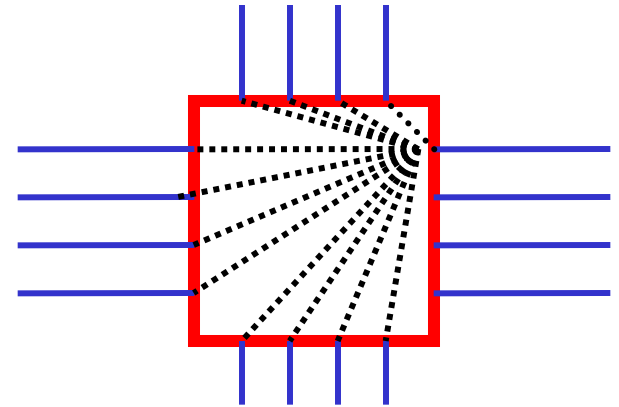


- Segment length?
 - Long: carry the signal longer, less “concatenation” switches, but might waste track
 - Short: local connections, slow for longer connections



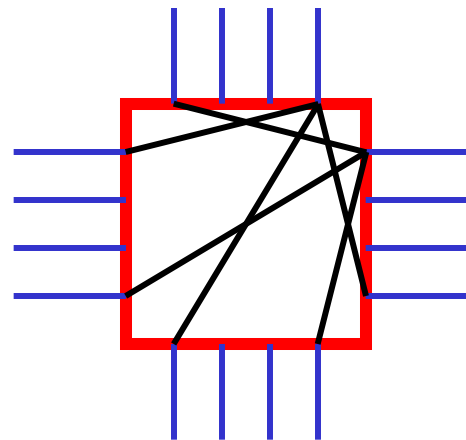
FPGA Switch Boxes

- Ideally, provide switches for all possible connections

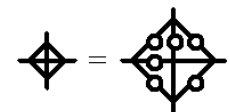
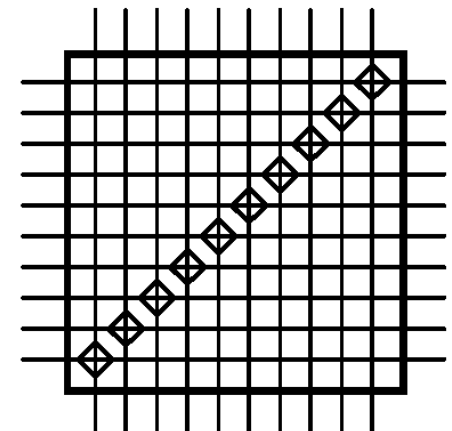


- Trade-off:

- Too many switches:
 - Large area
 - Complex to program
- Too few switches:
 - Cannot route signals



One possible
solution



Xilinx 4000

FPGA Routing

- Routing resources pre-fabricated
 - 100% routability using existing channels
 - If fail to route all nets, redo placement
- FPGA architectural issues
 - Careful balance between number of logic blocks and routing resources (100% logic area utilization?)
 - Designing flexible switchboxes and channels (conflicts with high clock speeds)
- FPGA routing algorithms
 - Graph search algorithms
 - Convert the wire segments to graph nodes, and switch elements to edges
 - Bin packing heuristics (nets as objects, tracks as bins)
 - Combination of maze routing and graph search algorithms