# Digital VLSI Design

# Lecture 9: Routing

Semester A, 2018-19

Lecturer: Dr. Adam Teman

January 25, 2019

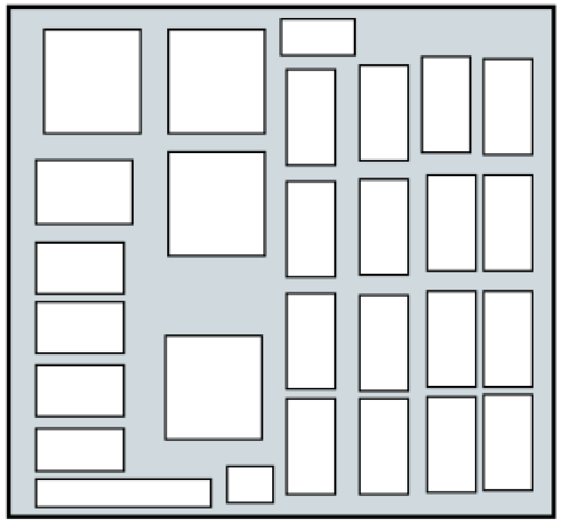
Emerging Nanoscaled Integrated Circuits and Systems Labs

Bar-Ilan University
אוניברסיטת בר-אילן

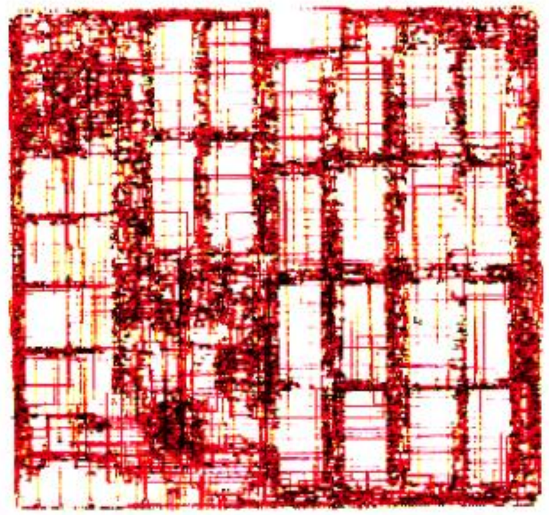
Disclaimer: This course was prepared, in its entirety, by Adam Teman. Many materials were copied from sources freely available on the internet. When possible, these sources have been cited; however, some references may have been cited incorrectly or overlooked. If you feel that a picture, graph, or code example has been copied from you and either needs to be cited or removed, please feel free to email adam.teman@biu.ac.il and I will address this as soon as possible.

# Routing: The Problem

- **Scale**
  - Millions of wires
  - MUST connect them all
- **Geometric Complexity**
  - Basic starting point – grid representation.
  - But at nanoscale – Geometry rules are complex!
  - Also, many routing layers with different "costs".
- **Electrical Complexity**
  - It's not enough *to just connect* all the wires.
  - You also have to:
    - Ensure that the delays through the wires are small.
    - Ensure that wire-to-wire interactions (crosstalk) doesn't mess up behavior.



Thousands of macro blocks
Millions of gates
Millions of wires



Kilometers of wire.

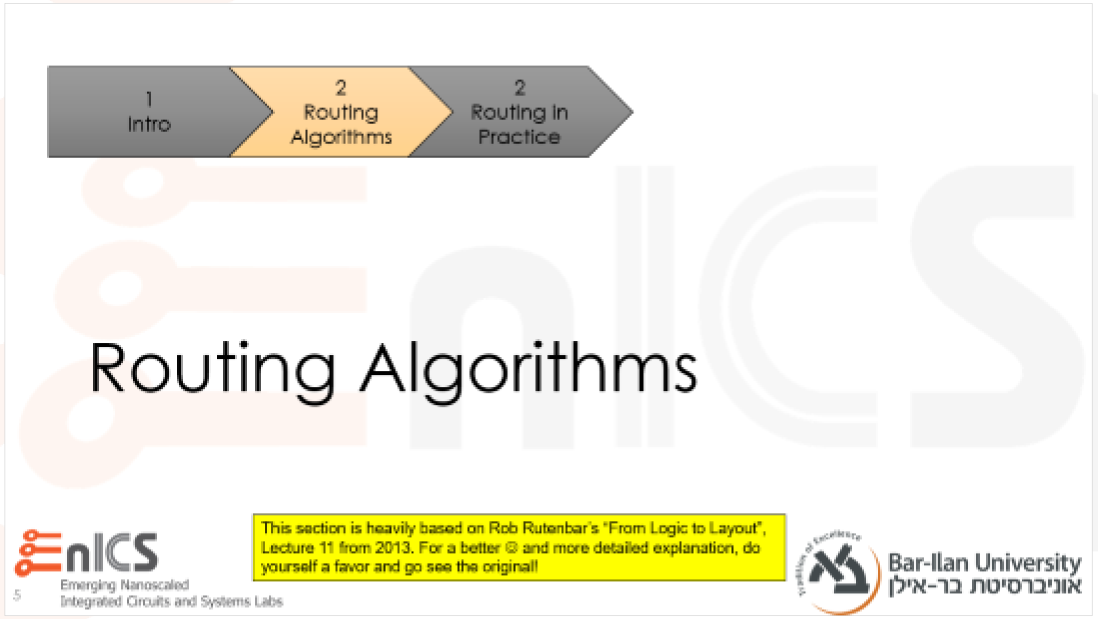
© Adam Teman, 2018

# Problem Definition

- **Problem:**
  - Given a placement, and a fixed number of metal layers, find a valid pattern of horizontal and vertical wires that <u>connect the terminals</u> of the nets.
- **Input:**
  - Cell locations, netlist
- **Output:**
  - Geometric layout of each net connecting various standard cells
- **Two-step process**
  - Global routing
  - Detailed routing
- **Objective**
  - 100% connectivity of a system
  - Minimum area, wirelength
- **Constraints**
  - Number of routing layers
  - Design rules
  - Timing (delay)
  - Crosstalk
  - Process variations

© Adam Teman, 2018

# Lecture Contents



1 Intro | 2 Routing Algorithms | 2 Routing in Practice

## Routing Algorithms

This section is heavily based on Rob Rutenbar's "From Logic to Layout", Lecture 11 from 2013. For a better ☺ and more detailed explanation, do yourself a favor and go see the original!

EnICS Emerging Nanoscaled Integrated Circuits and Systems Labs

5

Bar-Ilan University אוניברסיטת בר-אילן

1 Intro | 2 Routing Algorithms | 2 Routing in Practice

## Routing in practice

EnICS Emerging Nanoscaled Integrated Circuits and Systems Labs

20

Bar-Ilan University אוניברסיטת בר-אילן

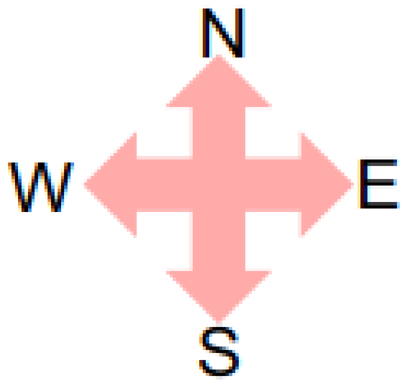
© Adam Teman, 2018

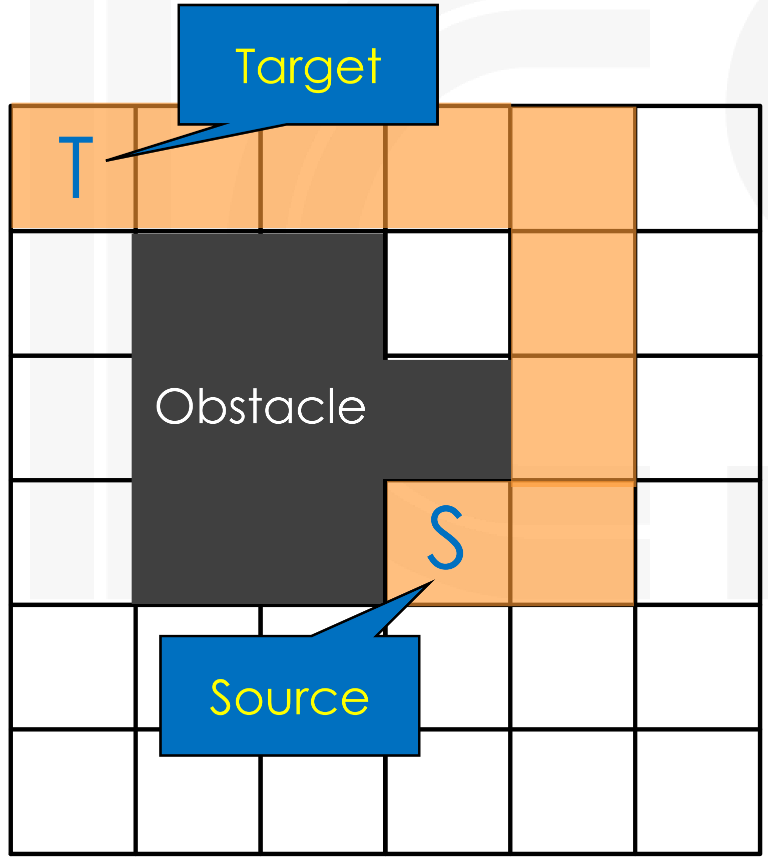
# Routing Algorithms

This section is heavily based on Rob Rutenbar's "From Logic to Layout", Lecture 11 from 2013. For a better ☺ and more detailed explanation, do yourself a favor and go see the original!

# Grid Assumption

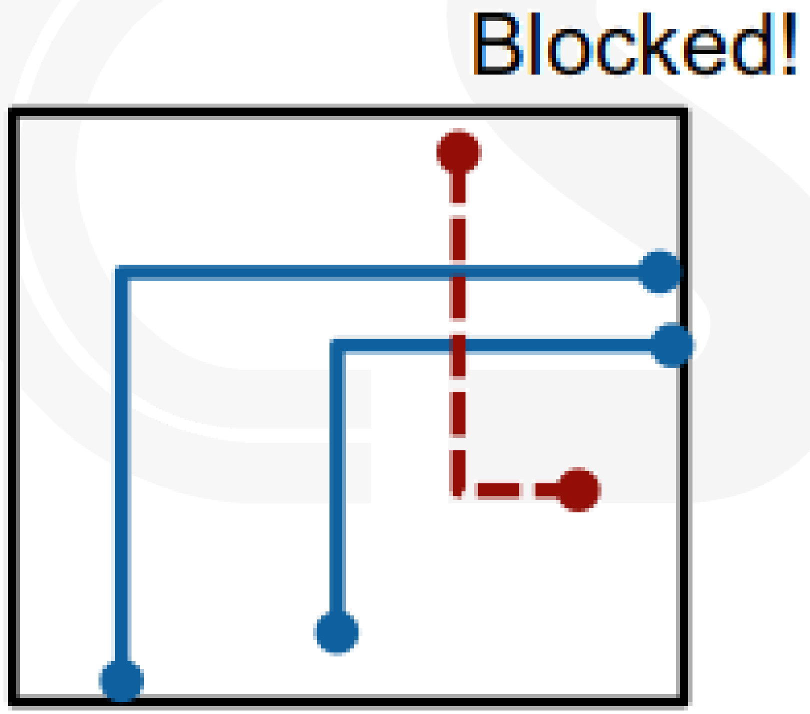
- **Despite the complexity of nanoscaled routing, we will use a grid assumption and add the complexity in later.**
  - Layout is a grid of regular squares
  - A legal wire is a set of connected grid cells through unobstructed cells.
  - Obstacles (or blockages) are marked in the grid.
  - Wires are strictly horizontal and vertical (Manhattan Routing)
  - Paths go
    - North/South
    - East/West.





# Maze Routers

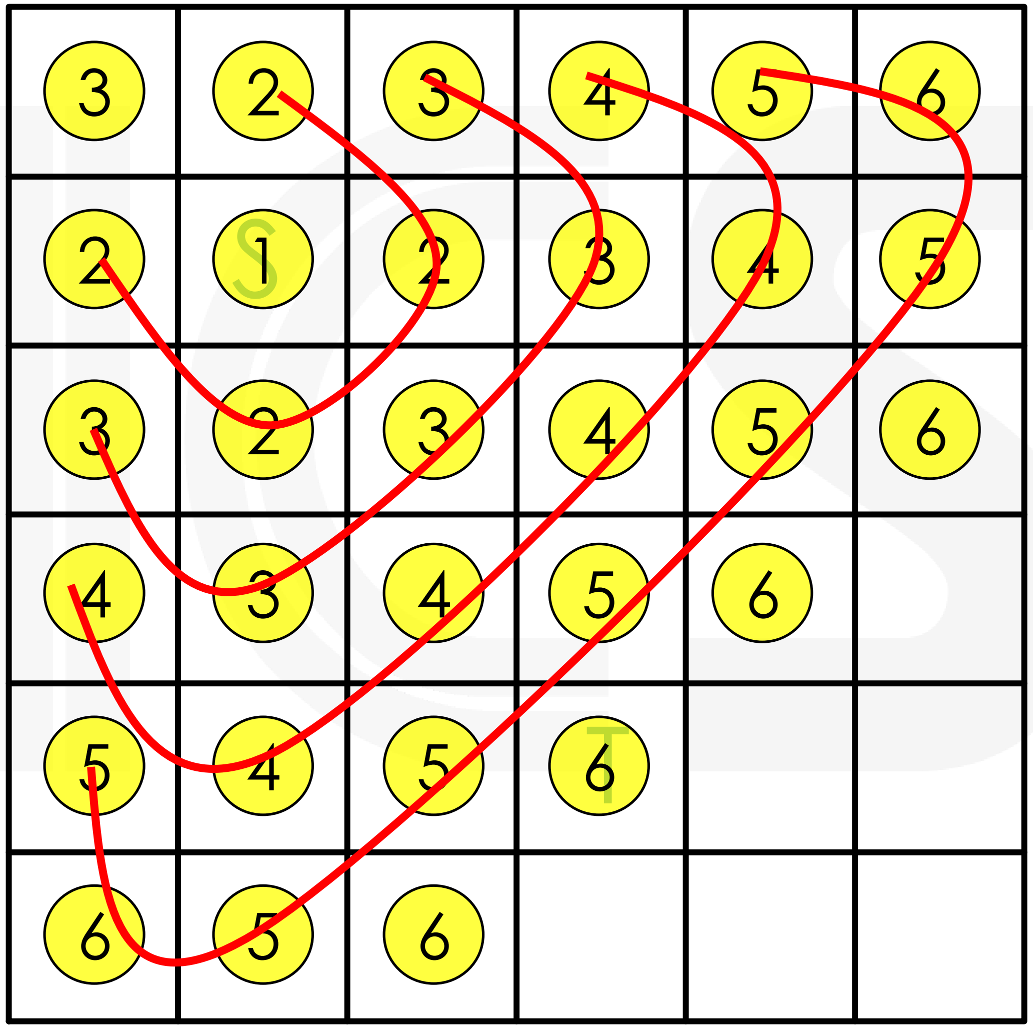
- **Also known as "Lee Routers"**
  - C. Y. Lee, "An algorithm for path connections and its applications" 1961
- **Strategy:**
  - Route one net at a time.
  - Find the best wiring path for the current net.
- **Problems:**
  - Early wired nets may block path of later nets.
  - Optimal choice for one net may block others.
- **Basic Idea:**
  - Expand → Backtrace → Cleanup



© Adam Teman, 2018

# Maze Routing: Expansion

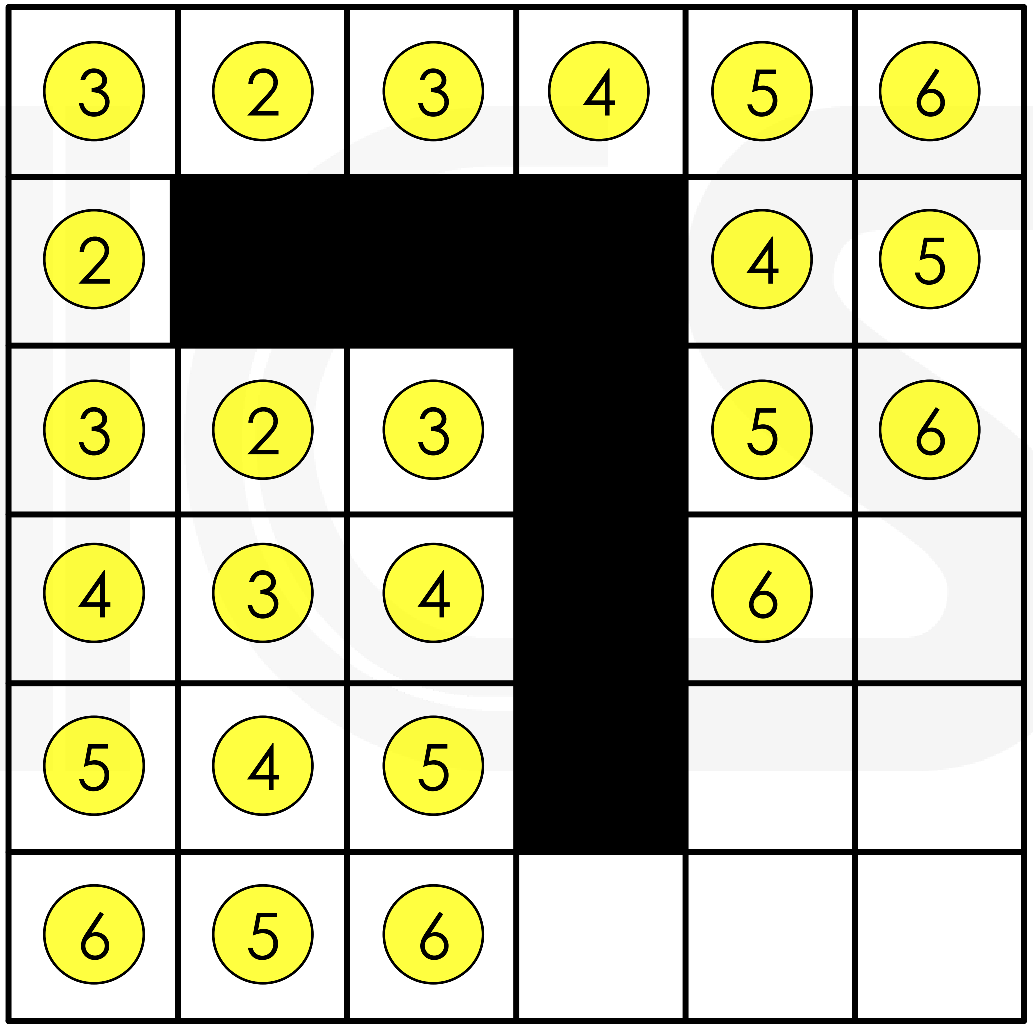
- **Start at the source.**
- **Find all paths 1 cell away.**
- **Continue until reaching the target.**
  - We approach the target with a "***wavefront***"
  - We found that the shortest path to the target is 6 unit steps.



© Adam Teman, 2018

# Maze Routing: Backtrace & Cleanup

- **Backtrace:**
  - Follow the path lengths backwards in descending order.
  - This will mark a shortest-path to the target.
  - However, there may be *many* shortest paths, so optimization can be used to select the *best* one.
- **Cleanup**
  - We have now routed the first net.
  - To ensure that future nets do not try to use the same resources, mark the net path from S to T as an obstacle.



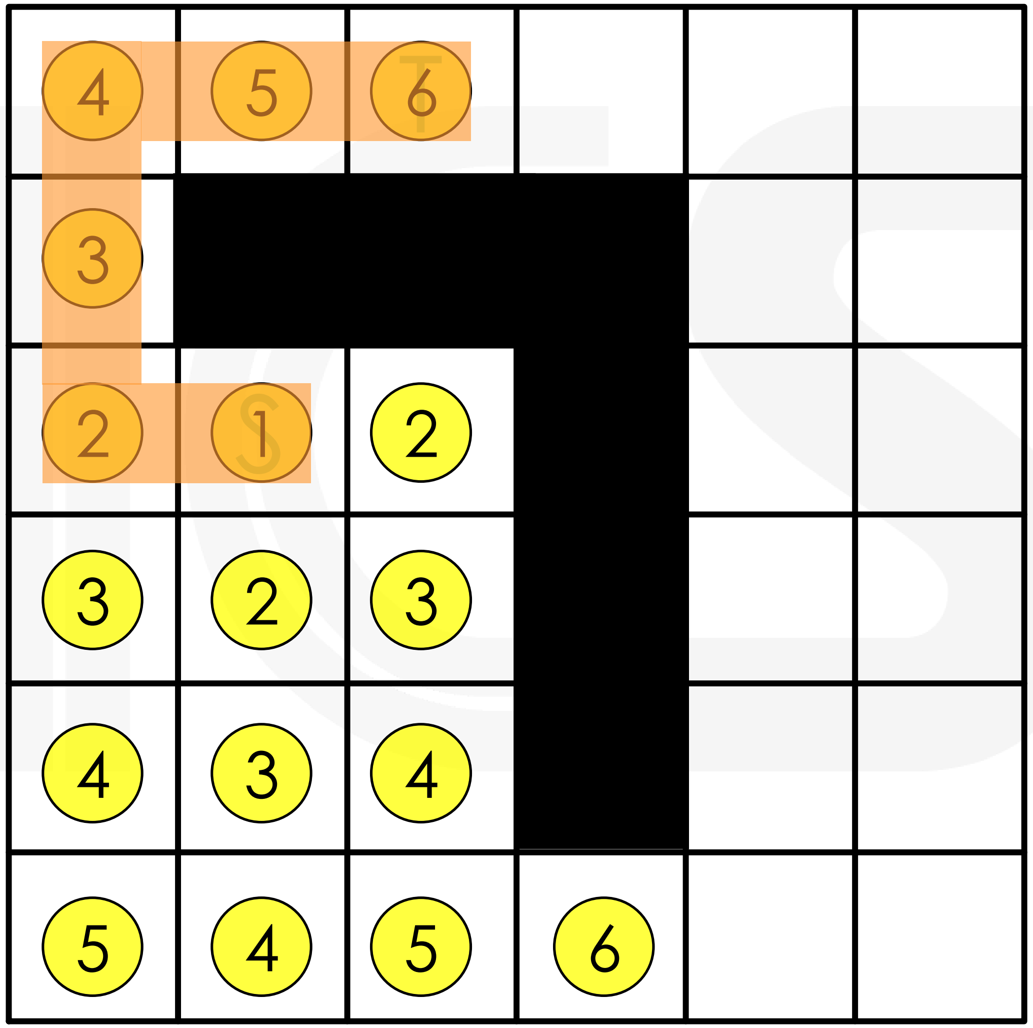
© Adam Teman, 2018

# Maze Routing: Blockages

- **How do we deal with blockages?**
  - Easy. Just “go around” them!

**To summarize:**

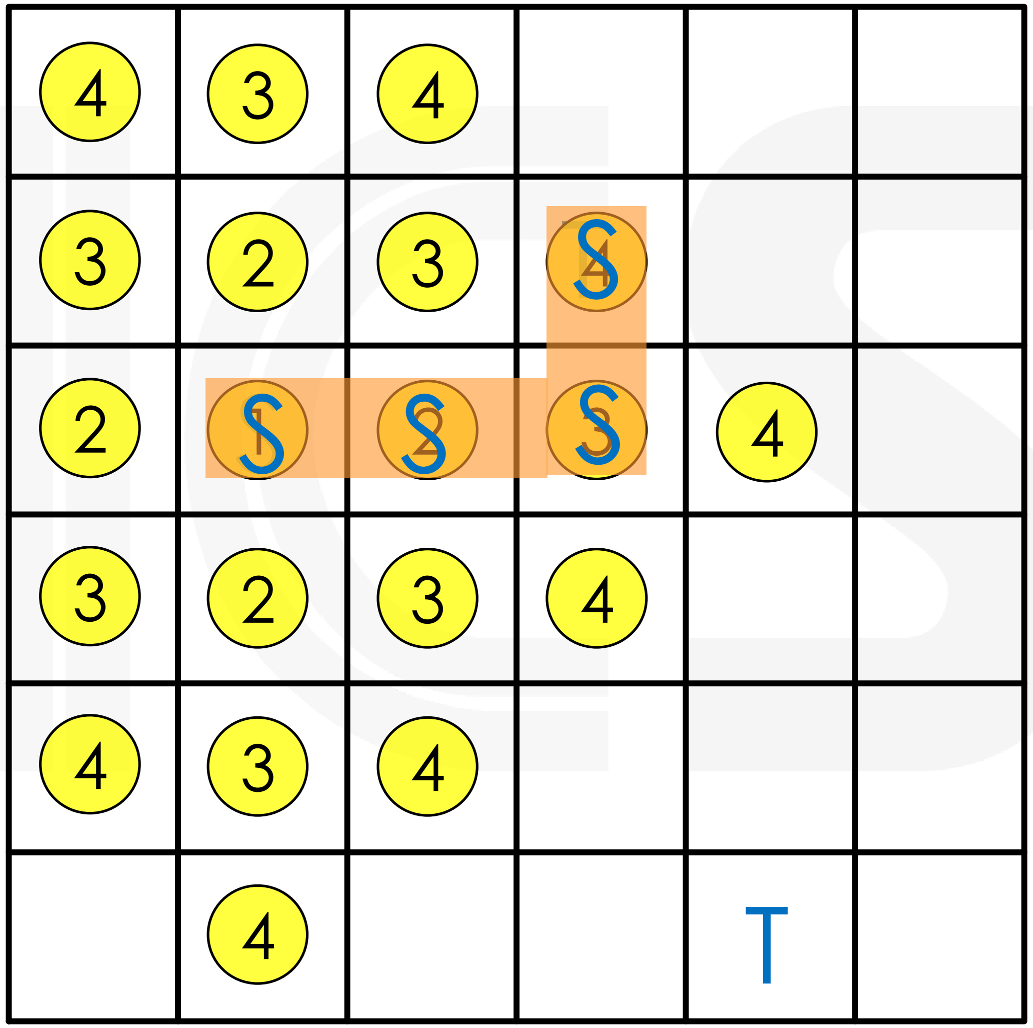
- **Expand:**
  - Breadth-first-search (BFS) to find all paths from S to T in path-length order.
- **Backtrace:**
  - Walk shortest path back to source.
- **Cleanup:**
  - Mark net as obstacle and erase distance markers.



© Adam Teman, 2018

# Multi-Point Nets

- **How do we go about routing a net with multiple targets?**
  - Actually, pretty straightforward.
  - Start with our regular maze routing algorithm to find the path to the nearest target.
  - Then re-label *all cells* on found path as *sources*, and re-run maze router using all sources simultaneously.

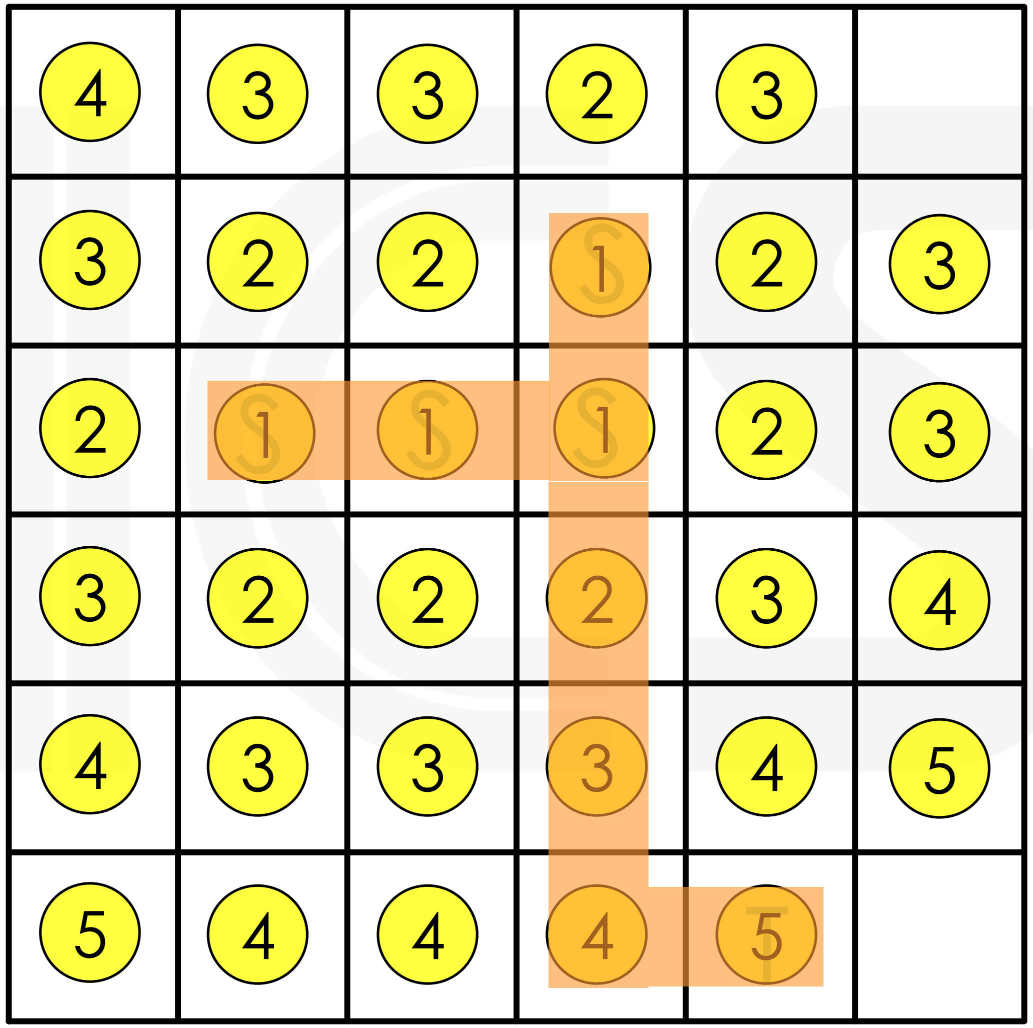


© Adam Teman, 2018

# Multi-Point Nets

- **How do we go about routing a net with multiple targets?**
  - Actually, pretty straightforward.
  - Start with our regular maze routing algorithm to find the path to the nearest target.
  - Then re-label *all cells* on found path as *sources*, and re-run maze router using all sources simultaneously.
  - Repeat until reaching all target points.

Note that this does not *guarantee* the shortest path (="Steiner Tree")



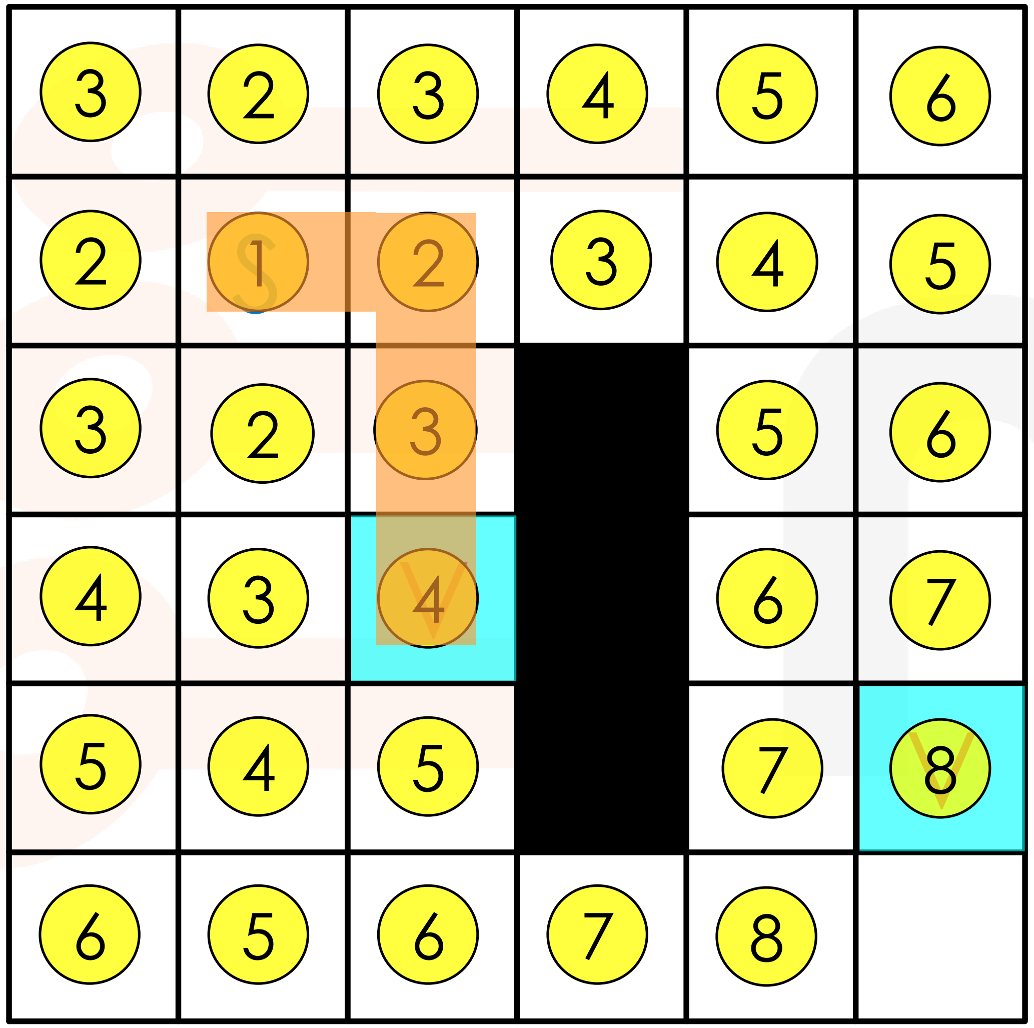
© Adam Teman, 2018

## Multi-Layer Routing

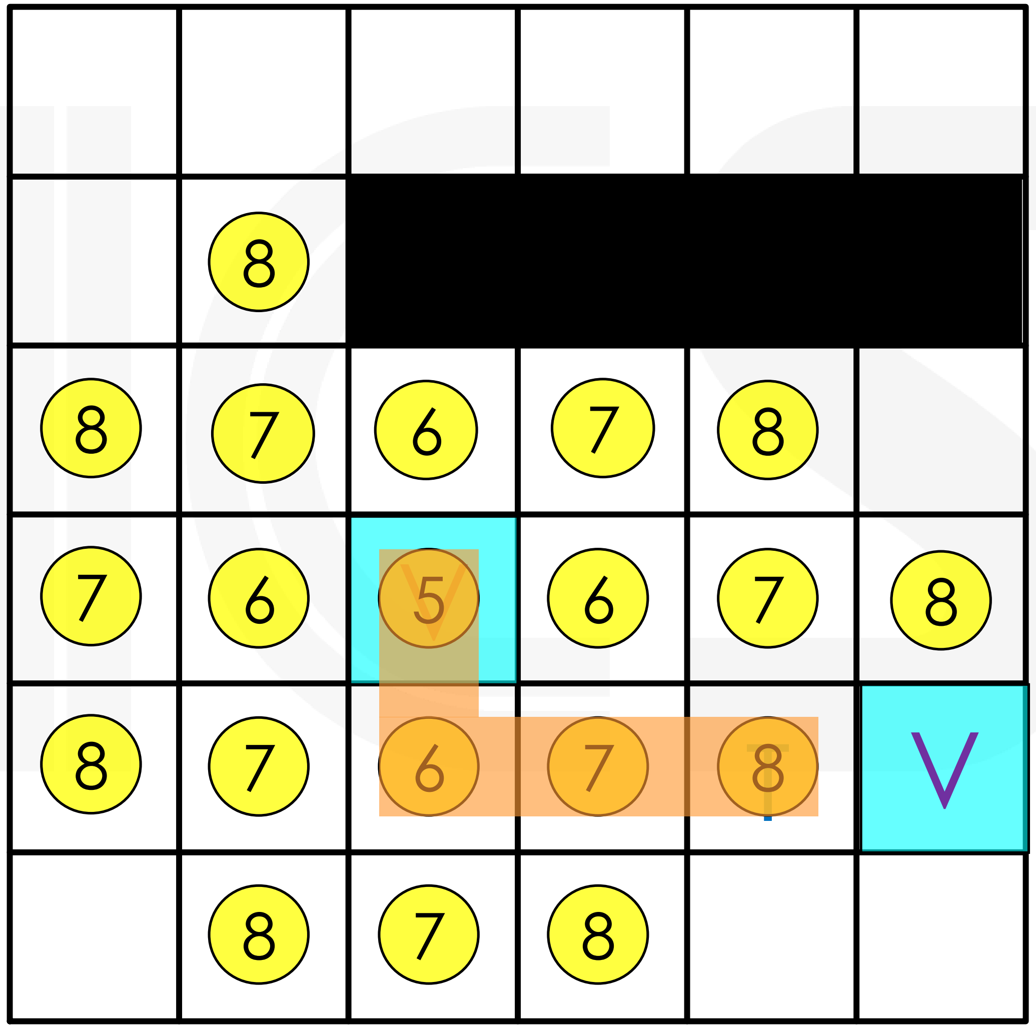
- **Okay, so what about dealing with several routing layers?**
  - Same basic idea of grid – one grid for each layer.
  - Each grid box can contain one via.
  - New expansion direction – up/down.

© Adam Teman, 2018

# Multi-Layer Routing



Metal 1



Metal 2

© Adam Teman, 2018

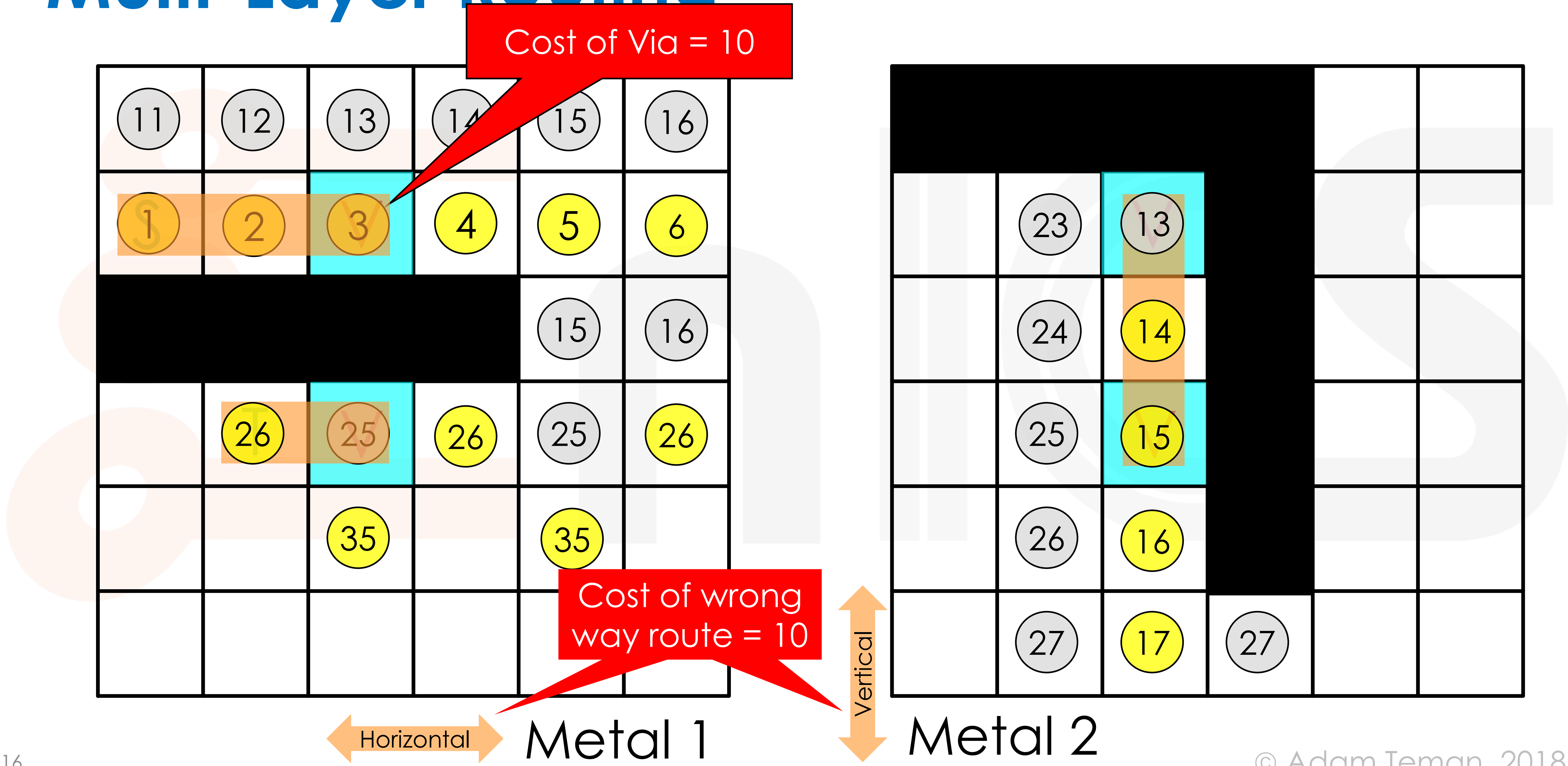
## Non-Uniform Grid Costs

- **But we know that vias have (relatively) high resistance.**
  - Shouldn’t we prefer to stay on the same metal layer?
- **We also prefer Manhattan Routing**
  - Each layer is only routed in one direction.
  - A “turn” requires going through a via or a “jog” should be penalized.
- **Is there a way to prefer routing in a certain layer/direction?**

- **Yes.**
  - Let’s introduce *non-uniform grid costs*.

© Adam Teman, 2018

# Multi-Layer Routing



© Adam Teman, 2018

# How do we implement this?

- **Grids are *huge*.**
  - Assume 1cm X 1cm chip.
  - Assume 100 nm track
  - Assume 10 routing layers
  - That is $10^{10}$ (100 billion) grid cells!
- **We need a low cost representation**
  - Only store the *wavefront*.
  - Remember which cells have been *reached*, at what *cost*, and from *which direction*.
  - Use *Dijkstra's algorithm* to find the cheapest cell first.
  - Store data in a *heap*.

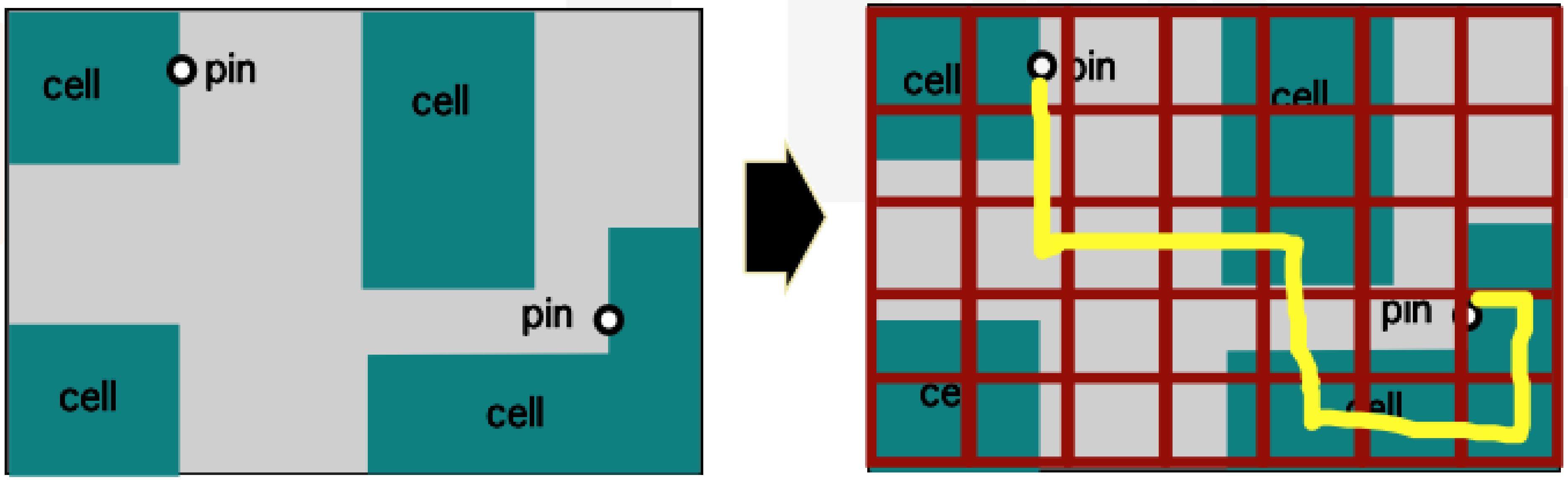
All of this is *hard*!

**Use many different heuristics**:
- Which net to route first
- Bias towards the right direction
- How to go about fixing problems
- etc., etc., etc.

© Adam Teman, 2018

# Divide and Conquer: Global Routing

- **To deal with a big chip, we make our problem smaller**
  - Divide the chip into big, course regions
  - e.g., 200 X 200 tracks each.
  - These are called GBOXes.
- **Now Maze Route through the GBOXes**

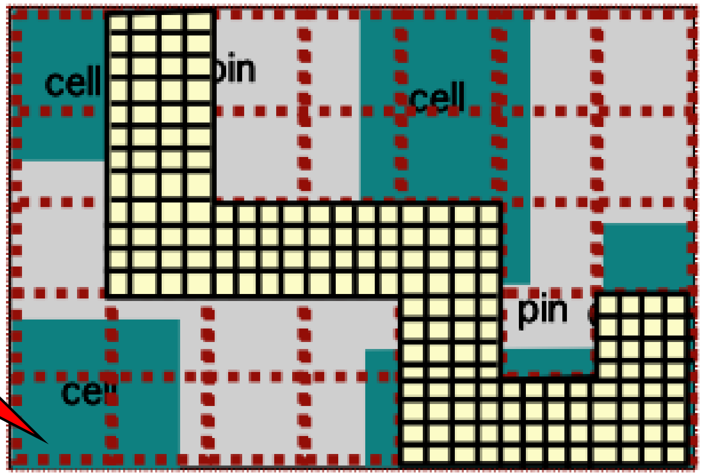


© Adam Teman, 2018

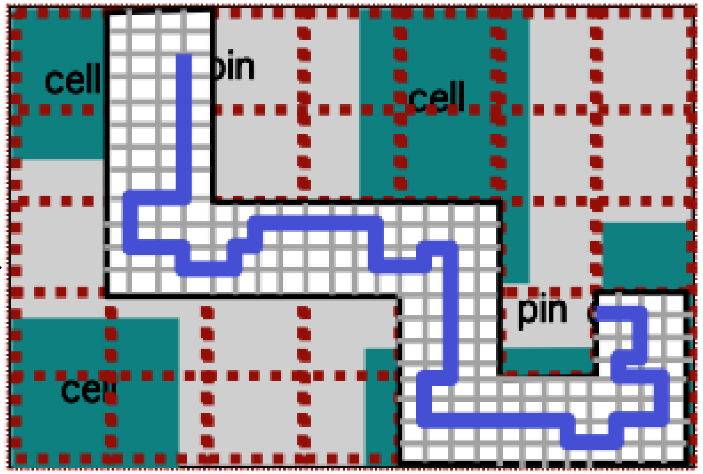
# Divide and Conquer: Global Routing

- **Global routing takes care of basic congestion.**
  - Balances *supply vs. demand* of routing resources.
  - Generates *regions of confinement* for the wires.
- **Detailed routing <u>decides on the exact path</u>.**

GBOXes have a *dynamic cost* according to how *congested* they are!



**Global router** tells us to search for detailed paths only here



**Detailed router** tells us exact final path in this region

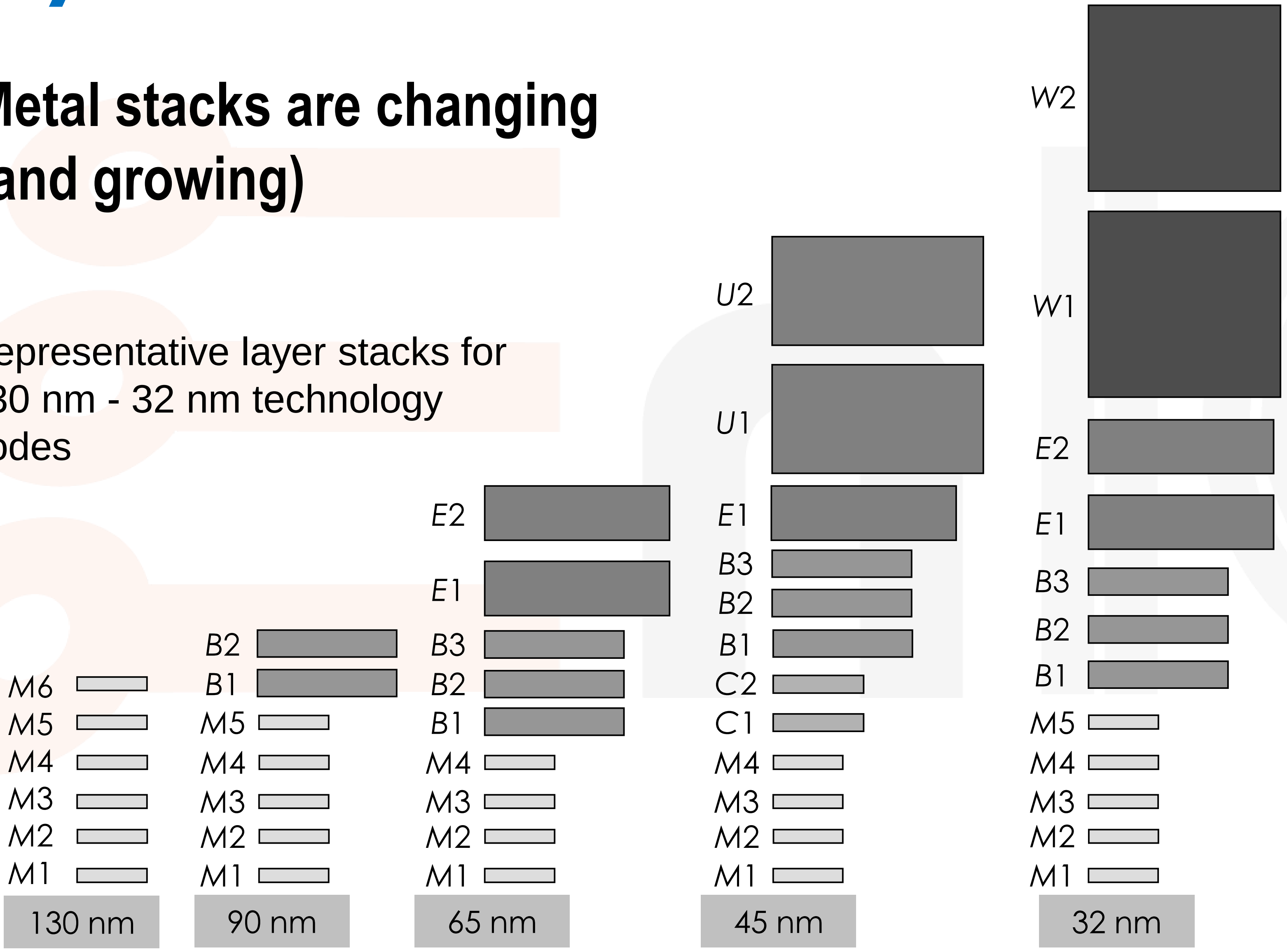
© Adam Teman, 2018

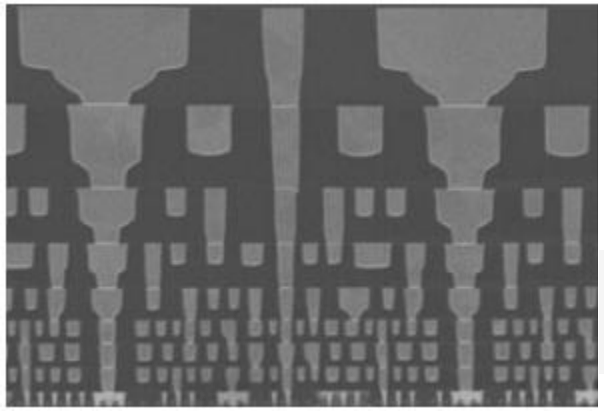
# Routing in practice

# Layer Stacks

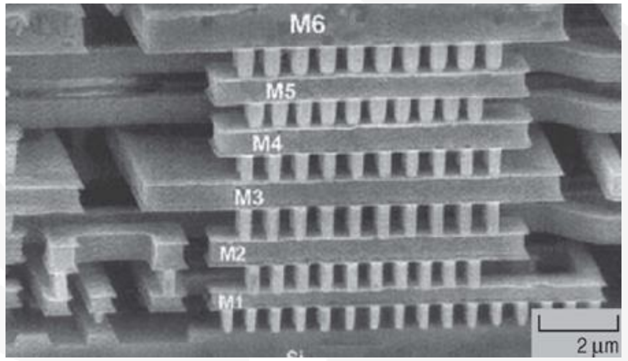
- **Metal stacks are changing (and growing)**

Representative layer stacks for 130 nm - 32 nm technology nodes





Intel 45nm 8 metal stack

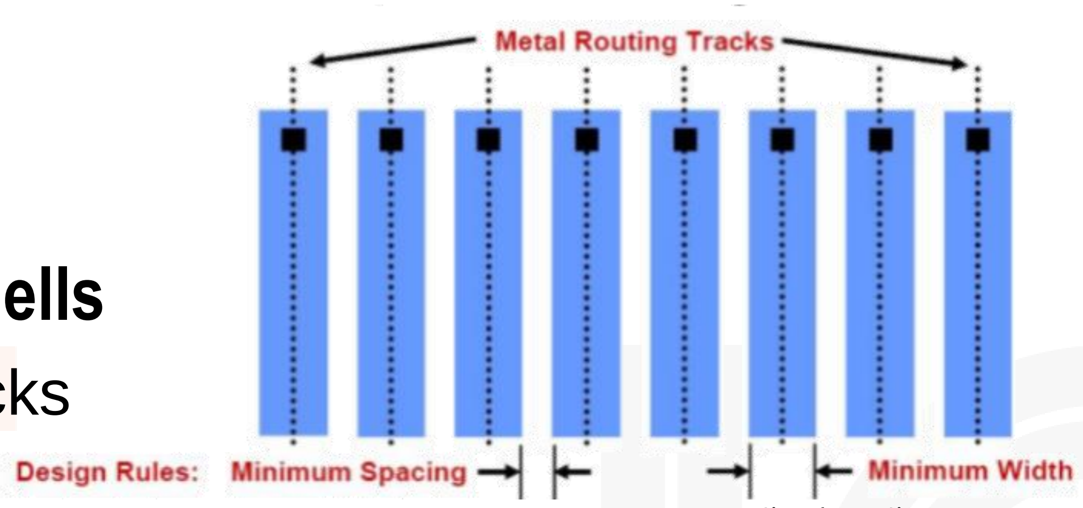


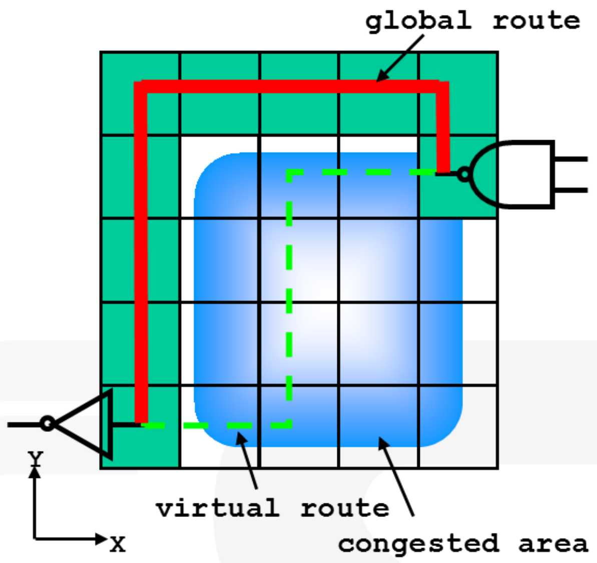
UMC 6 metal stack

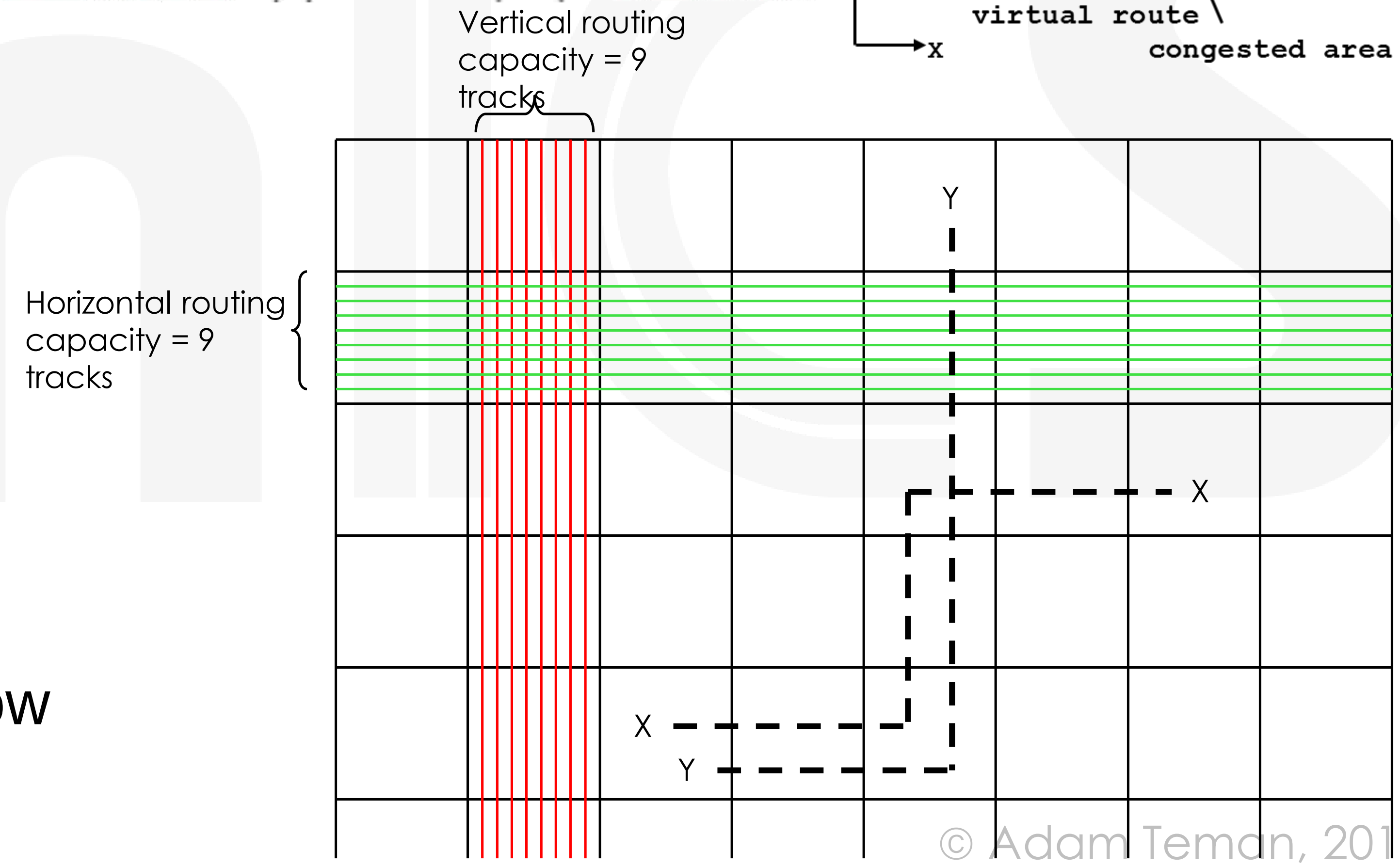
© Adam Teman, 2018

# Global Route

- **Divide floorplan into GCells**
  - Approximately 10 tracks per layer each.
- **Perform fast grid routing:**
  - Minimize wire-length
  - Balance Congestion
  - Timing-driven
  - Noise/SI-driven
  - Keep buses together
- **Also used for trial route**
  - During earlier stages of the flow





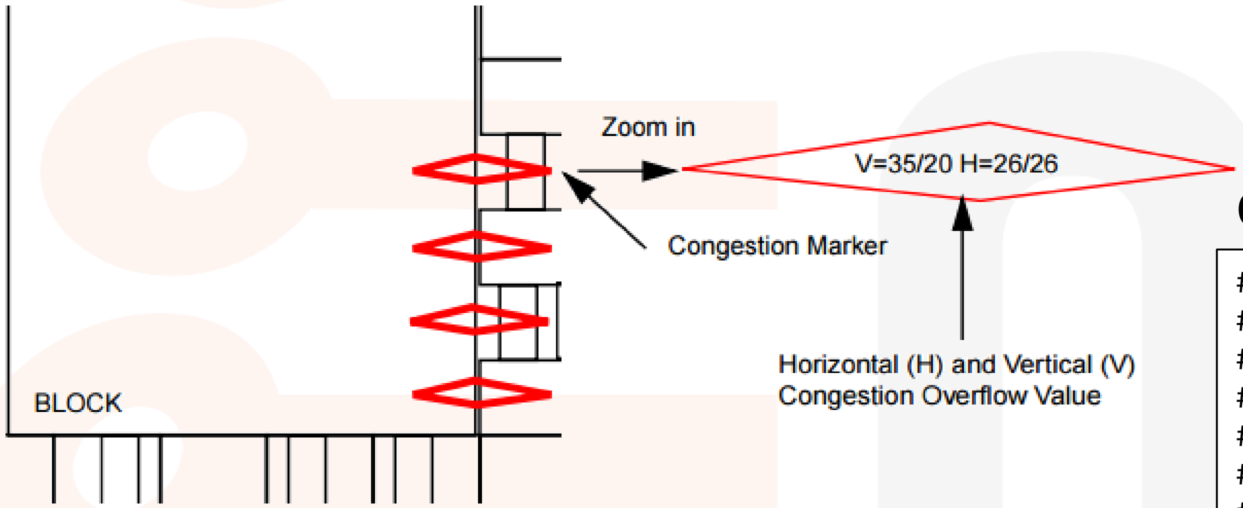


Vertical routing capacity = 9 tracks
Horizontal routing capacity = 9 tracks
Y
X
X
Y

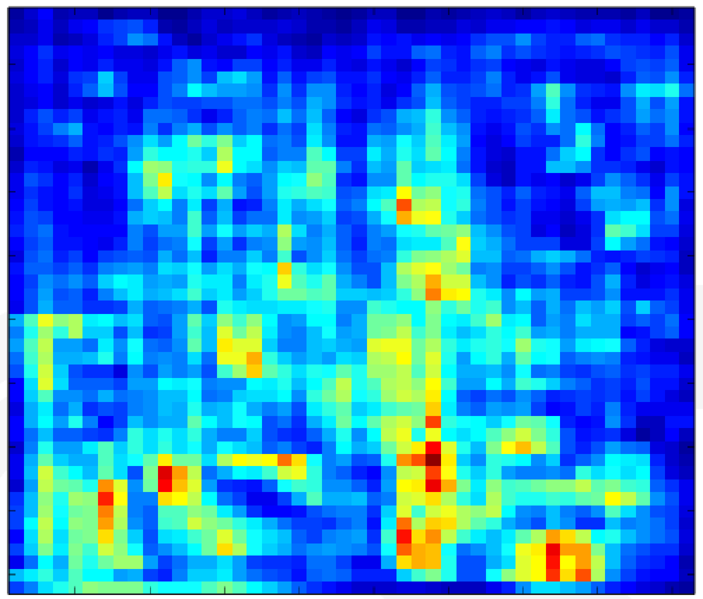
© Adam Teman, 2018

# Congestion Map

- **Use congestion map and report to examine design routability**



Congestion map



Congestion Report

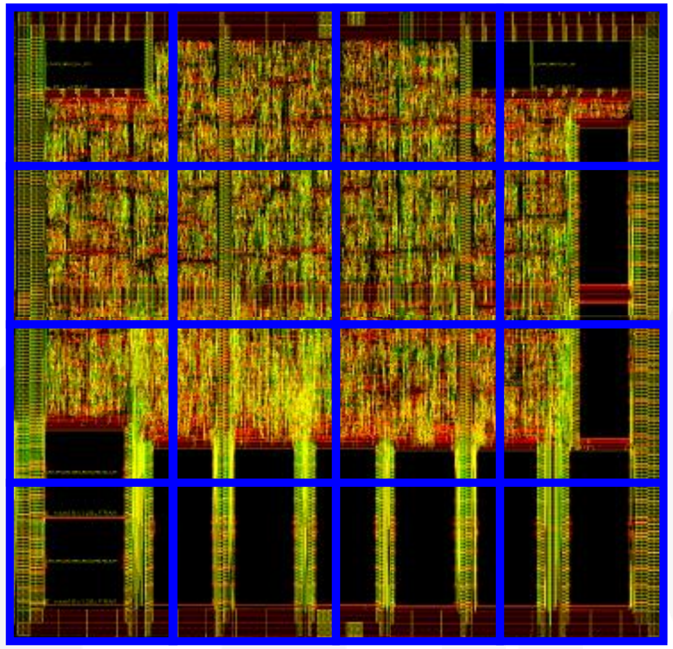
```
#                     Routing   #Avail       #Track       #Total     %Gcell
#  Layer             Direction   Track      Blocked        Gcell    Blocked
#  --------------------------------------------------------------------------
#  Metal 1               H        7607         9692      1336335     62.57%
#  Metal 2               H        7507         9792      1336335     55.84%
#  Metal 3               V        7636         9663      1336335     59.51%
#  Metal 4               H        8609         8691      1336335     52.02%
#  Metal 5               V        5747        11551      1336335     56.39%
#  Metal 6               H        5400        11899      1336335     55.09%
#  Metal 7               V        1831         2486      1336335     55.30%
#  Metal 8               H        2415         1903      1336335     43.85%
#  --------------------------------------------------------------------------
#  Total                         46753       56.99%     10690680     55.07%
#
#  589 nets (0.47%) with 1 preferred extra spacing.
```

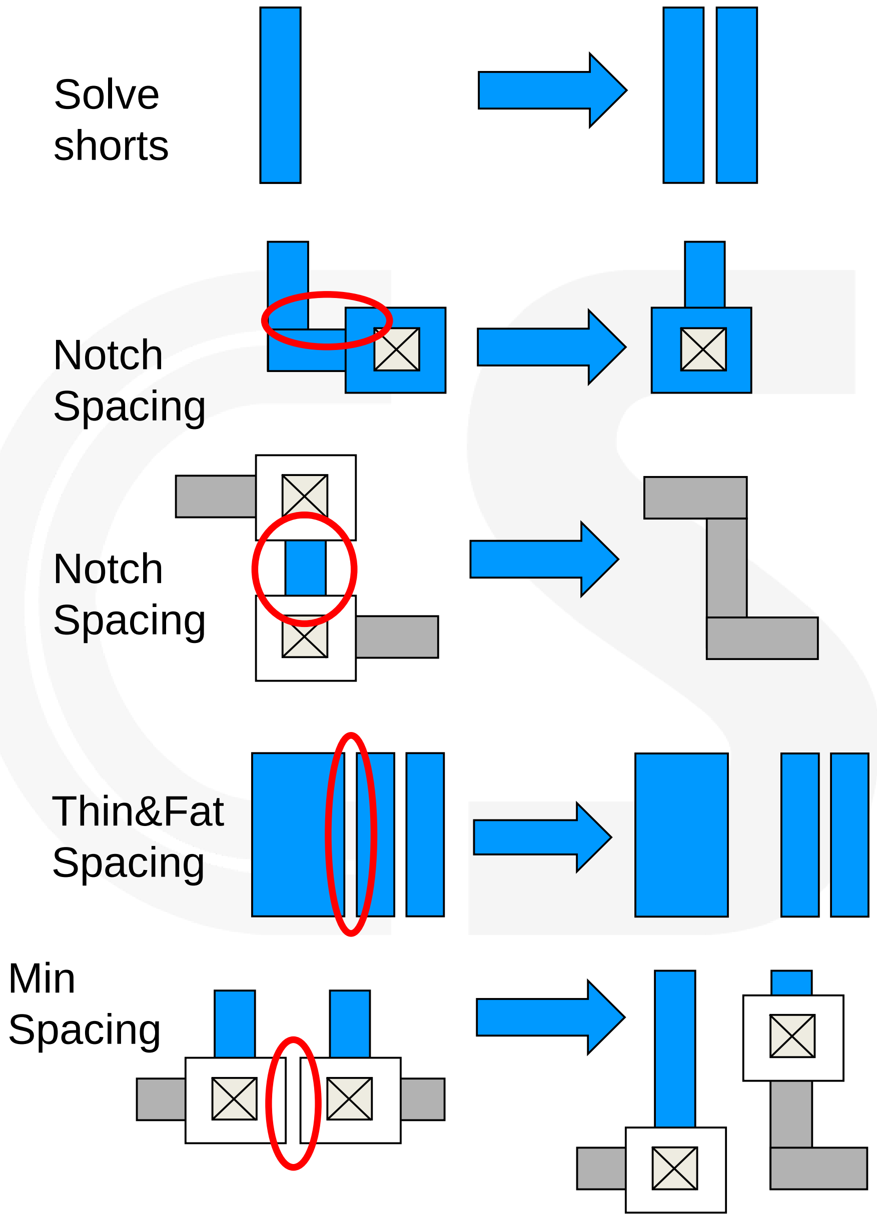
© Adam Teman, 2018

# Detailed Route

- **Using global route plan, within each global route cell**
  - Assign nets to tracks
  - Lay down wires
  - Connect pins to nets
  - Solve DRC violations
  - Reduce cross couple cap
  - Apply special routing rules
- **Flow:**
  - Track Assignment (TA)
  - DRC fixing inside a Global Routing Cell (GRC)
  - Iterate to achieve a solution (default ~20 iterations)

Detailed Route Boxes





© Adam Teman, 2018

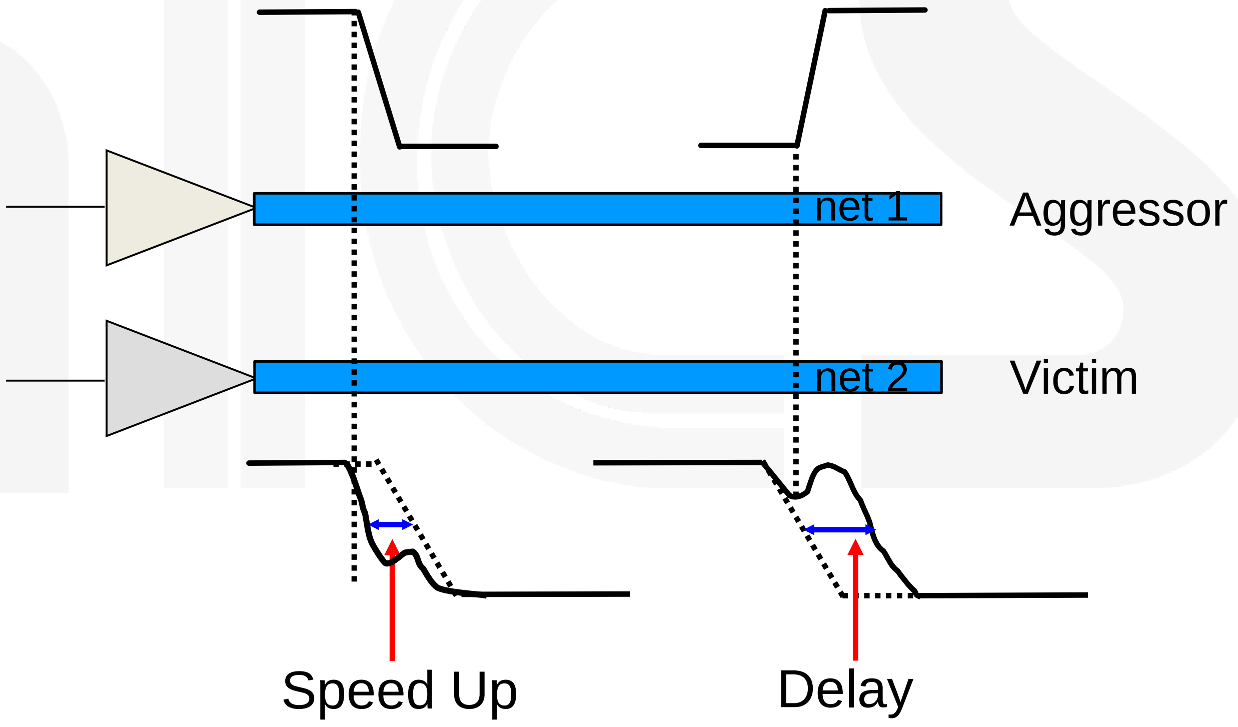
# Timing-Driven Routing

- **Optimize critical paths**
- **Route some nets first**
  - Most routing freedom at start
  - Use shortest paths possible
- **Net weights**
  - Order of routing (priorities: e.g., Default : Clocks 50, others 2)
- **Wire widening**
  - Reduce resistance
- **If you have a congested design you may need to set the timing driven effort to "low"**
- **Beware when changing default options**

© Adam Teman, 2018

# Signal Integrity (SI)

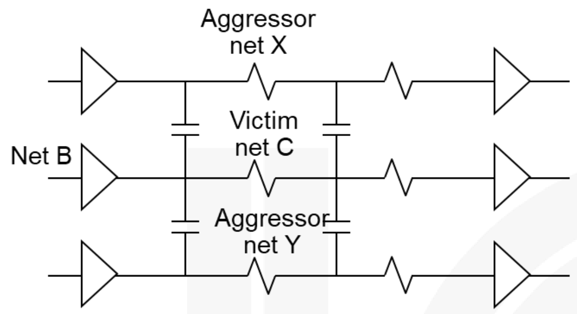
- **Signal Integrity during routing is synonymous with *Crosstalk*.**
  - A switching signal may affect a neighboring net.
  - The switching net is called the *Aggressor*.
  - The affected net is called the *Victim*.
- **Two major effects:**
  - Signal slow down
    - When the aggressor and victim switch in ***opposite*** directions.
  - Signal speed up
    - When the aggressor and victim switch in the ***same*** direction.

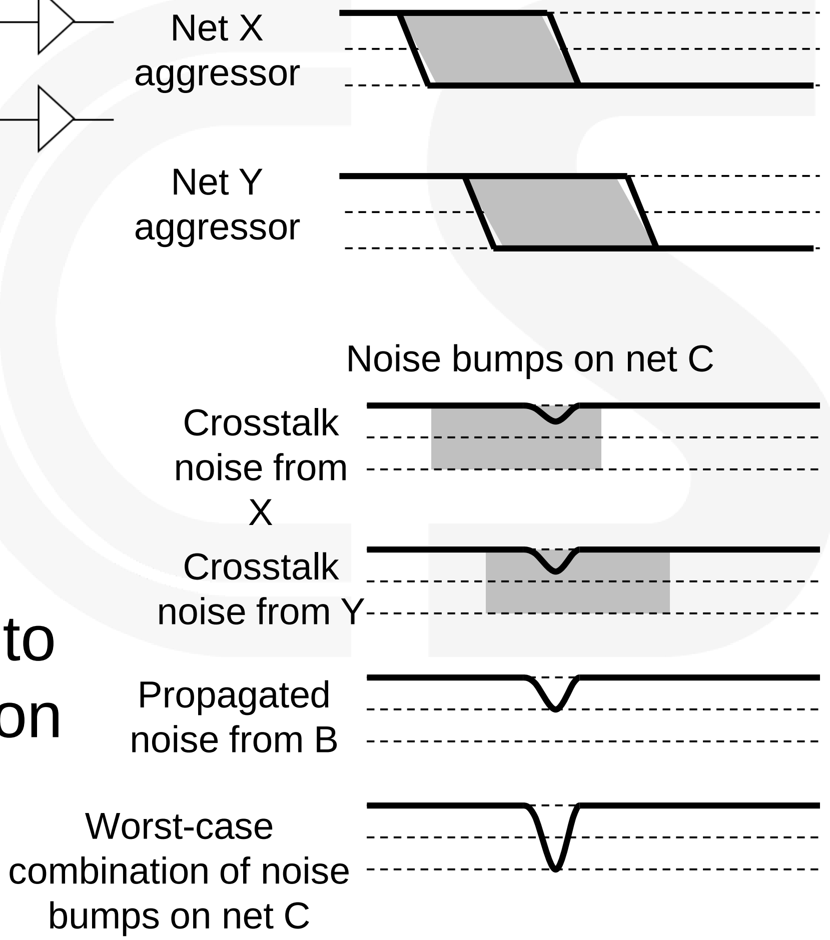


© Adam Teman, 2018

# SI Multi-Aggressor Timing Analysis

- **Infinite Window Analysis**
  - An *infinite noise window* applies the maximum delay due to crosstalk during timing analysis.
  - This model was sufficient for older (pre-90nm) technologies, but became too severe with the growing sidewall capacitances at scaled nodes.
- **Propagated Noise Analysis**
  - Min/Max vectors are propagated through the design to create a transition window for all aggressors in relation to a certain victim.
  - Noise is only applied at the overlap of the two windows to determine the worst case noise bump.

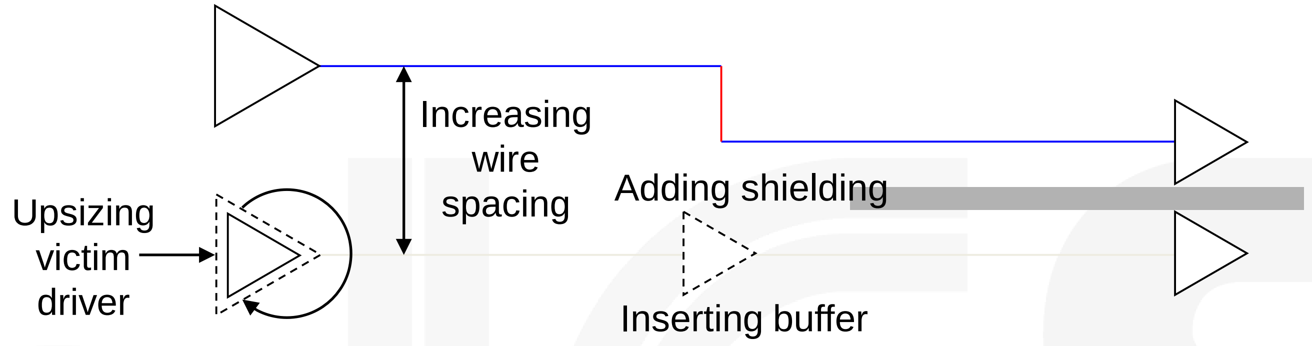


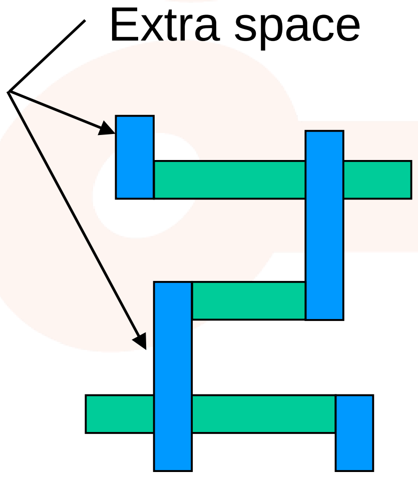


© Adam Teman, 2018

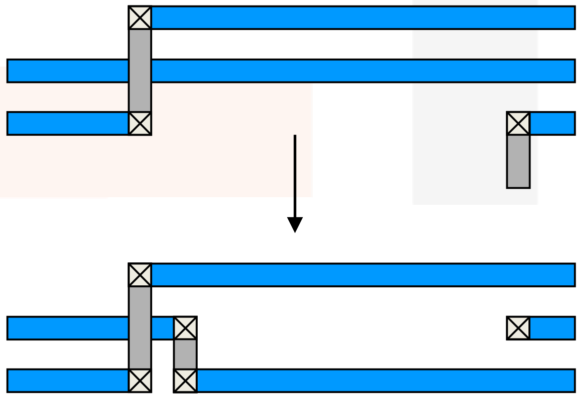
# Signal Integrity - Solutions

- **Crosstalk Prevention**
  - Limit length of parallel nets
  - Wire spreading
  - Shield special nets
  - Upsize driver or buffer

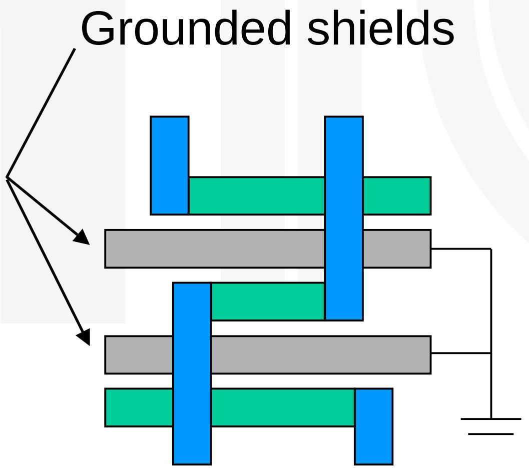




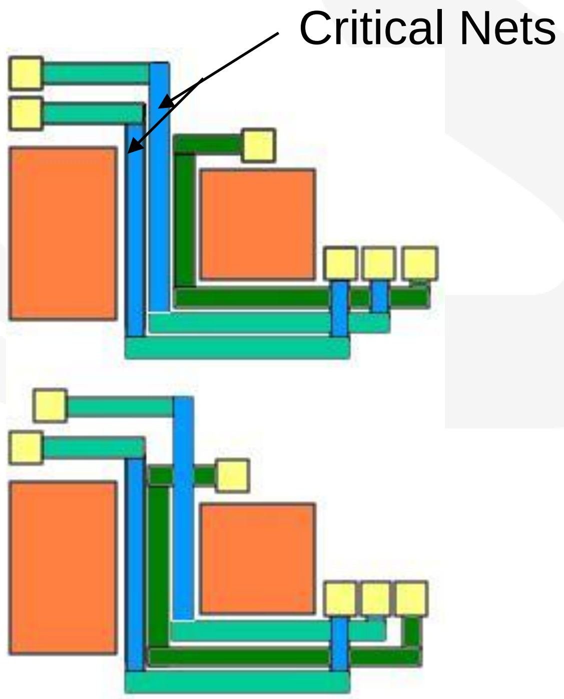
Spacing



Wire Spreading



Shielding
Same layer (H)
Adjacent layers (V)

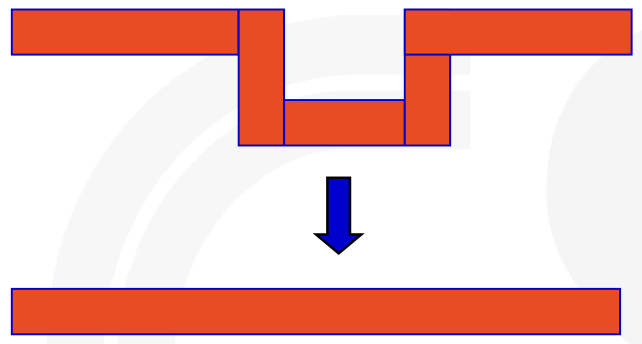


Net Ordering

© Adam Teman, 2018

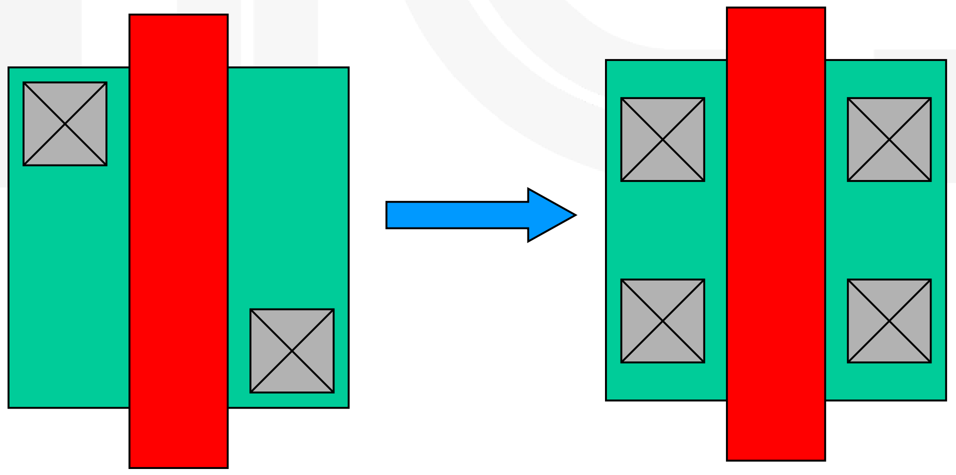
# Design For Manufacturing

- **During route, apply additional design for manufacturing (DFM) and/or design for yield (DFY) rules:**
  - Via reduction
  - Redundant via insertion
  - Wire straightening
  - Wire spreading



Wire straightening (reduce jogs)

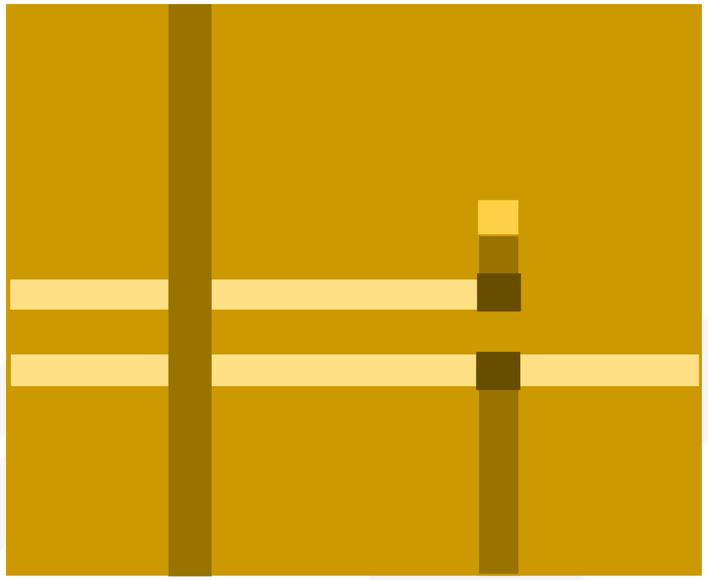
Avoid asymmetrical contacts

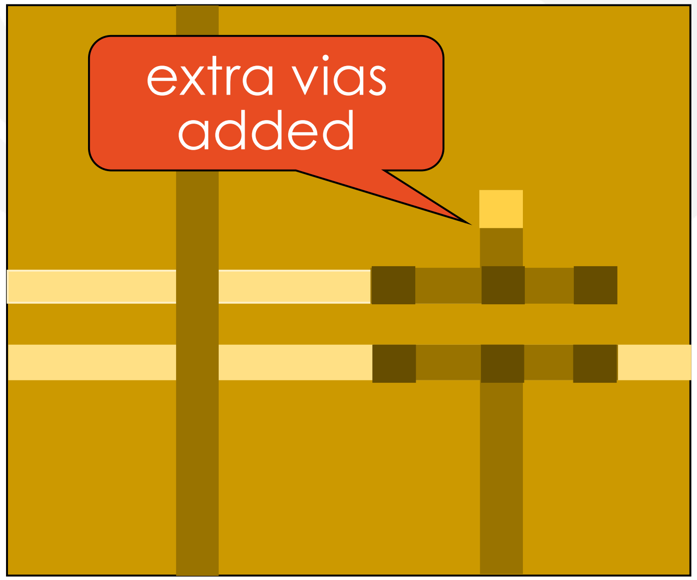


© Adam Teman, 2018

# DFM: Via Optimization

- **Post-Route Via Optimization, includes:**
  - Incremental routing for the minimization of vias.
  - Replacement of single vias with multi-cut vias.
- **These operations are required for:**
  - Reliability:
    - The ability to create reliable vias decreases with each process node. If a single via fails, it creates an open and the circuit is useless.
  - Electromigration:
    - Electromigration hazards are even more significant in vias, which are essentially long, narrow conductors.



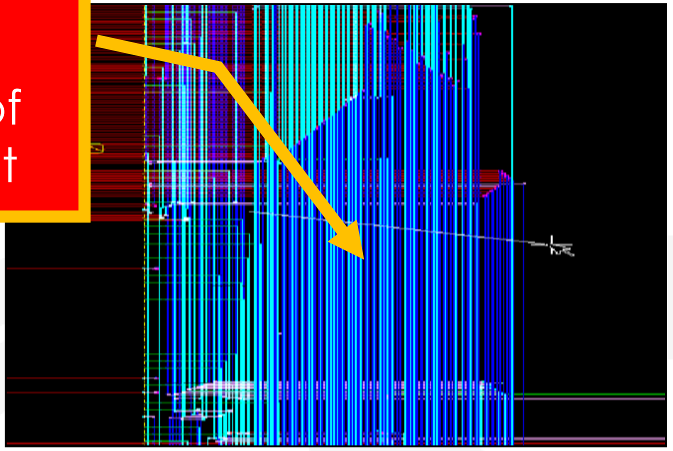


© Adam Teman, 2018

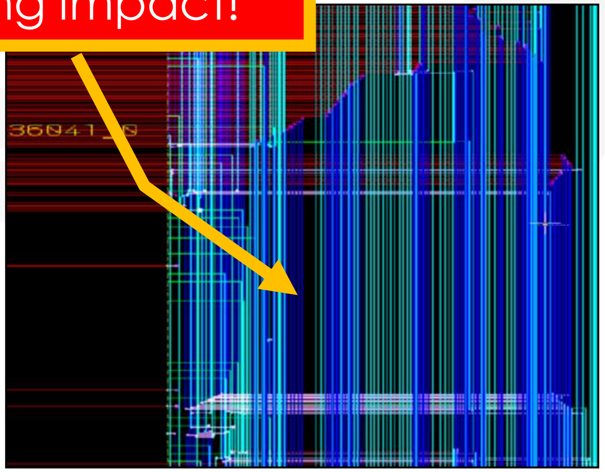
# DFM: Wire Spreading

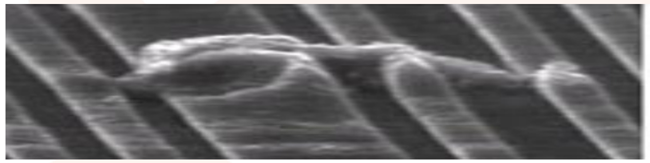
- **Wire spreading achieves:**
  - Lower capacitance and better signal integrity.
  - Lower susceptibility to shorts or opens due to random particle defects.

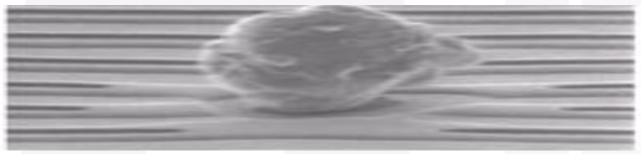
- High-density critical area
- High probability of yield-killing defect

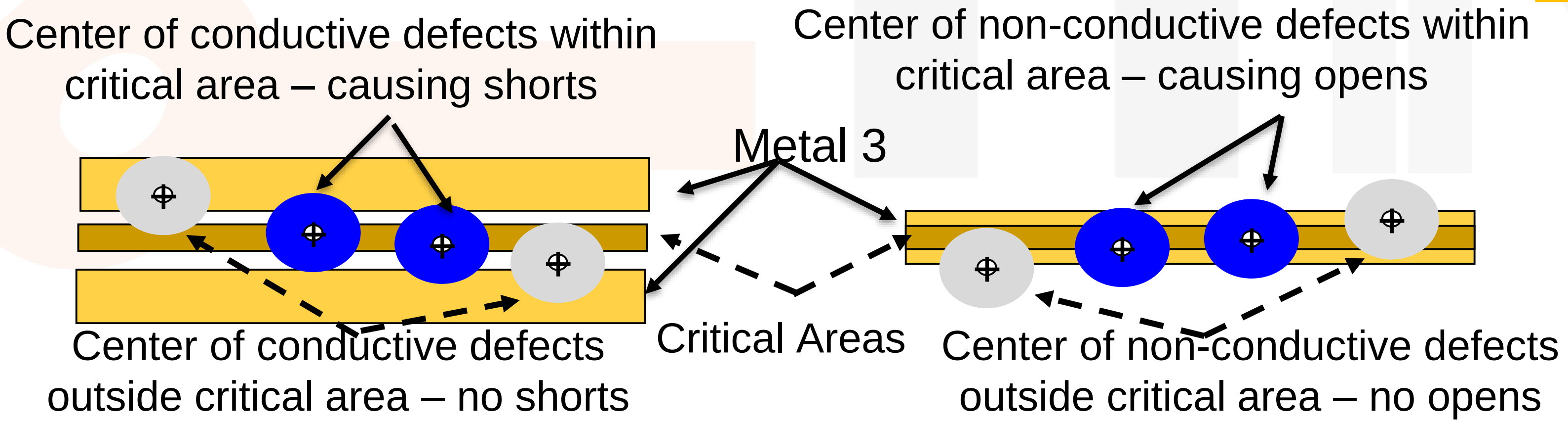


- Density reduced, yield risk reduced
- No timing impact!








Center of conductive defects within critical area – causing shorts
Center of non-conductive defects within critical area – causing opens
Metal 3
Critical Areas
Center of conductive defects outside critical area – no shorts
Center of non-conductive defects outside critical area – no opens

© Adam Teman, 2018

# Routing in Innovus/Encounter

- **The detailed routing engine used by Innovus/Encounter is called "NanoRoute"**
  - NanoRoute provides concurrent timing-driven and SI-driven routing.
  - In addition, it can perform multi-cut via insertion, wire widening and spacing.
- **The commands for running a route with NanoRoute are:**

```
set_db route_design_with_timing_driven true
set_db route_design_with_si_driven true
route_design
```

- **Following detailed route wire optimization and timing optimization:**

```
set_db route_design_with_timing_driven false
set_db route_design_detail_post_route_spread_wire true
set_db route_design_detail_use_multi_cut_via_effort high
route_design –wire_opt
set_db route_design_with_timing_driven true
opt_design -post_route –setup -hold
```

© Adam Teman, 2018

# Routing in Innovus/Encounter

- **To achieve a high percentage of multi-cut vias:**

```
set_db route_design_concurrent_minimize_via_count_effort high
set_db route_design_detail_use_multi_cut_via_effort high
```

- **To check your design after routing:**

```
report_route; # provide routing statistics
report_wires; # provides wire statistics including wirelength
time_design -post_route; # check timing after routing
check_drc; # Run a DRC check – in new techs: “verify_drc”
check_connectivity; # Run an LVS check
```

- **To perform incremental routing (ECO routing):**

```
set_db route_design_with_eco true
route_global_detail
```

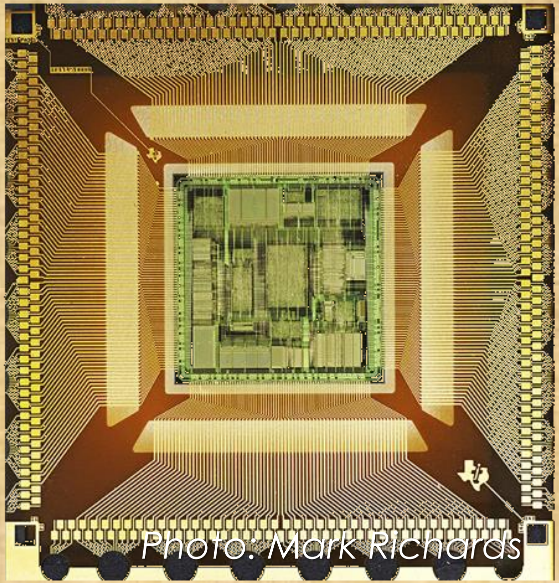
© Adam Teman, 2018

# The Chip Hall of Fame

- **With RISC processors a central part of our computing lives today, we should really thank the revolution of the**

## Sun Microsystems SPARC

- Scalable Processor Architecture
- Release date: 1987 SPARC v7 32-bit Architecture
- The first major commercial RISC processor taking Patterson's ideas at Berkeley into a product.
- "SPARC will take Sun from a $500M/year company to a $1B/year company".
- The first SPARC powered the Sun-4 workstations, which made Sun a $1B/year company.
- Terminated in 2017 by Oracle, but now part of Fujitsu.



SPARCstation 1+ "pizzabox"

2017 Inductee to the IEEE Chip Hall of Fame

## Main References

- Rob Rutenbar "From Logic to Layout" 2013
- Synopsys University Courseware
- IDESA
- Kahng, et al. "VLSI Physical Design: From Graph Partitioning to Timing Closure" – Chapter 6

© Adam Teman, 2018