Digital VLSI Design

Lecture 10: Routing

Semester A, 2016-17

Lecturer: Dr. Adam Teman

1 February 2017

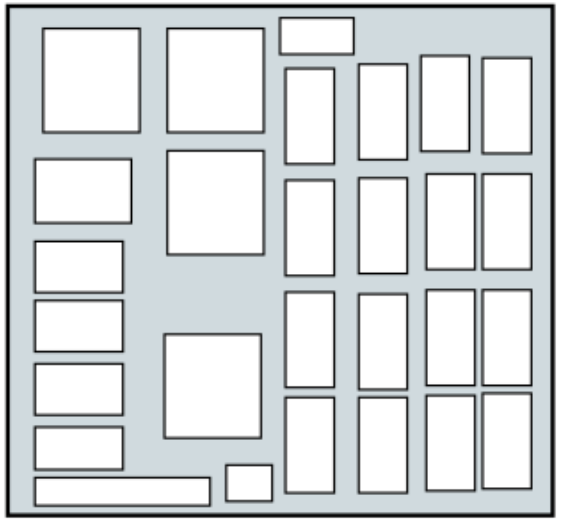
Emerging Nanoscaled Integrated Circuits and Systems Labs

Tradition of Excellence

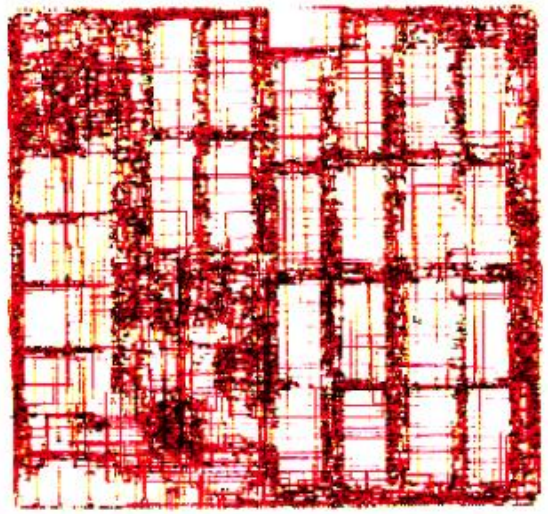
Bar-Ilan University
אוניברסיטת בר-אילן

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.

Problem Definition

- **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.
- **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

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:**

```
setNanoRouteMode -routeWithTimingDriven true -routeWithSiDriven true
routeDesign
```

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

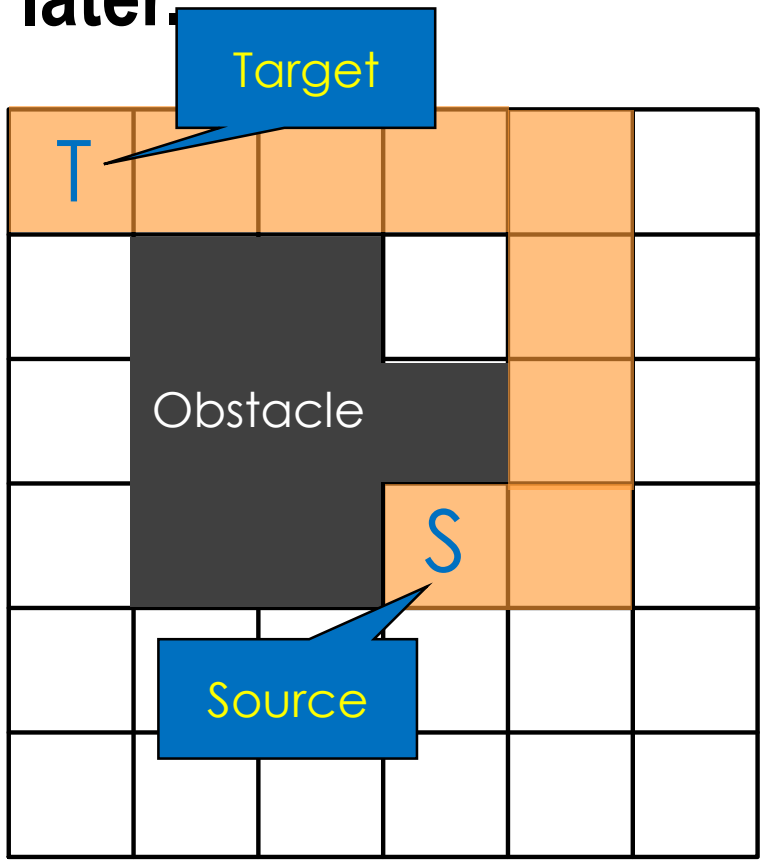
```
setNanoRouteMode -routeWithTimingDriven false \
        -droutePostRouteSpreadWire true -drouteUseMultiCutViaEffort high
routeDesign –wireOpt
setNanoRouteMode -routeWithTimingDriven true
optDesign –postRoute –setup -hold
```

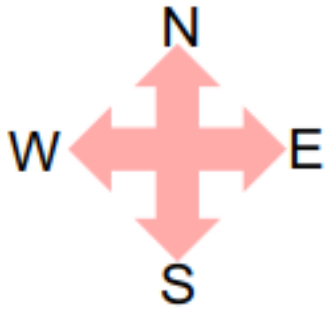
Routing Algorithms

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

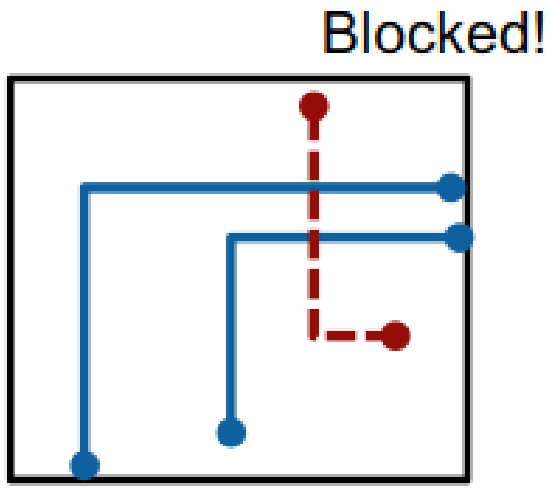
- **Despite the complexity of nanoscaled routing, we will use a grid assumption and add the complexity in later.**
  - Layout is a grid of regular squares
  - A legal wire is a set of connected grid cells through unobstructed cells.
  - Obstacles (or blockages) are marked in the grid.
  - Wires are strictly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





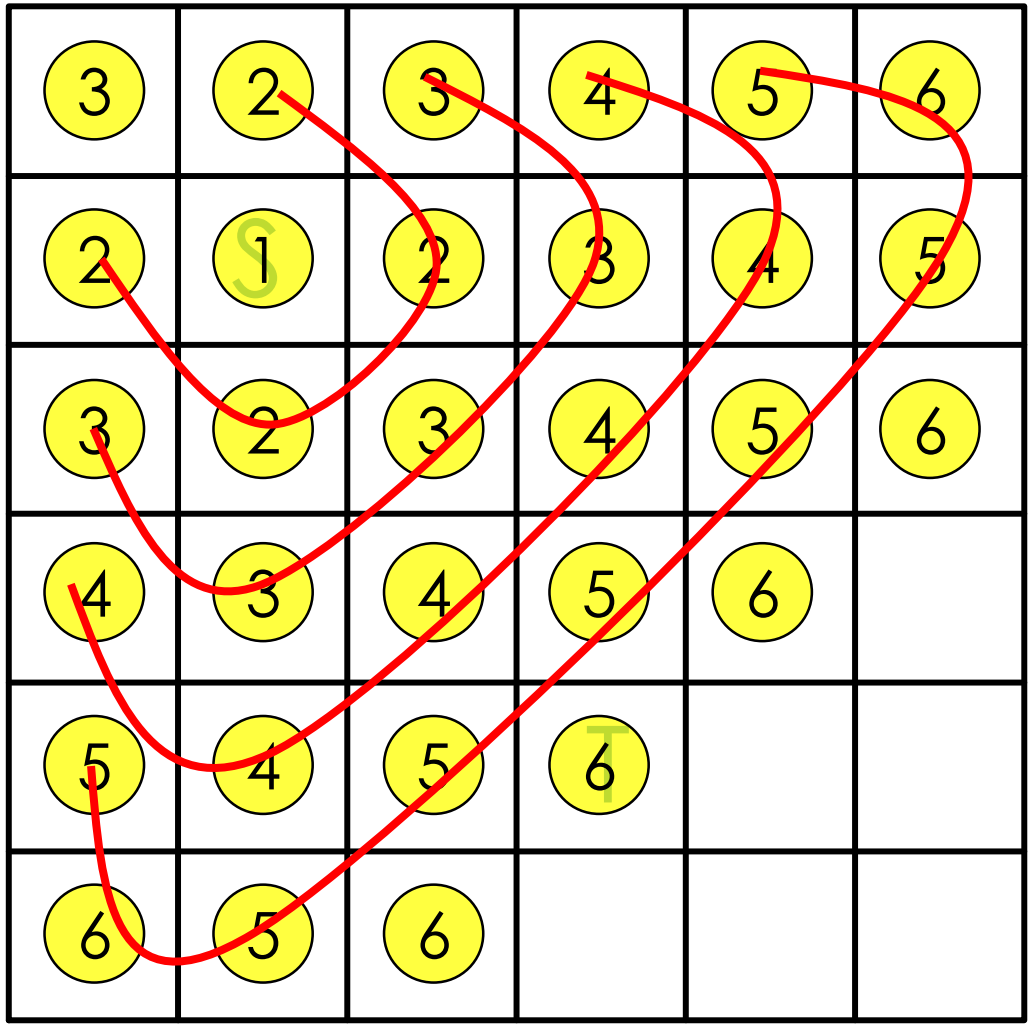
Maze Routers

- **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



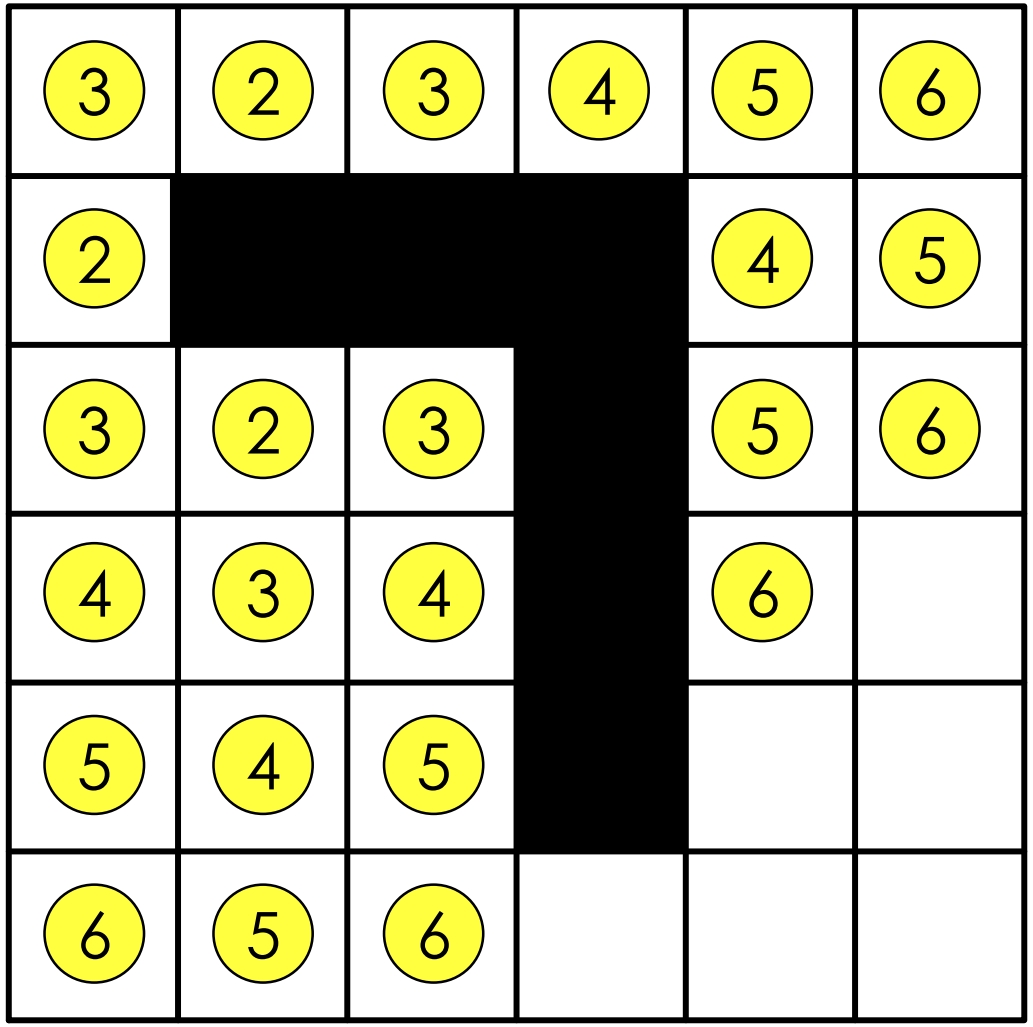
Maze Routing: Expansion

- **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.



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.

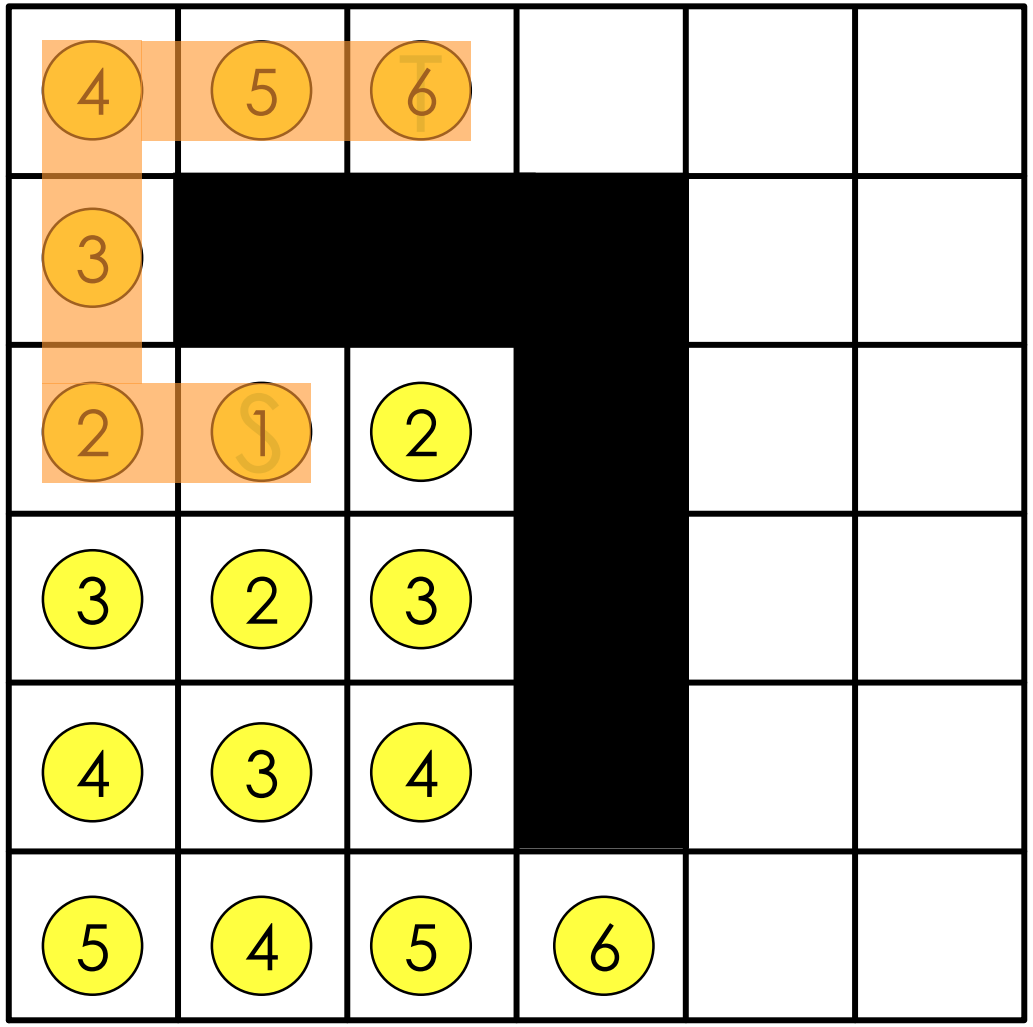


Maze Routing: Blockages

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

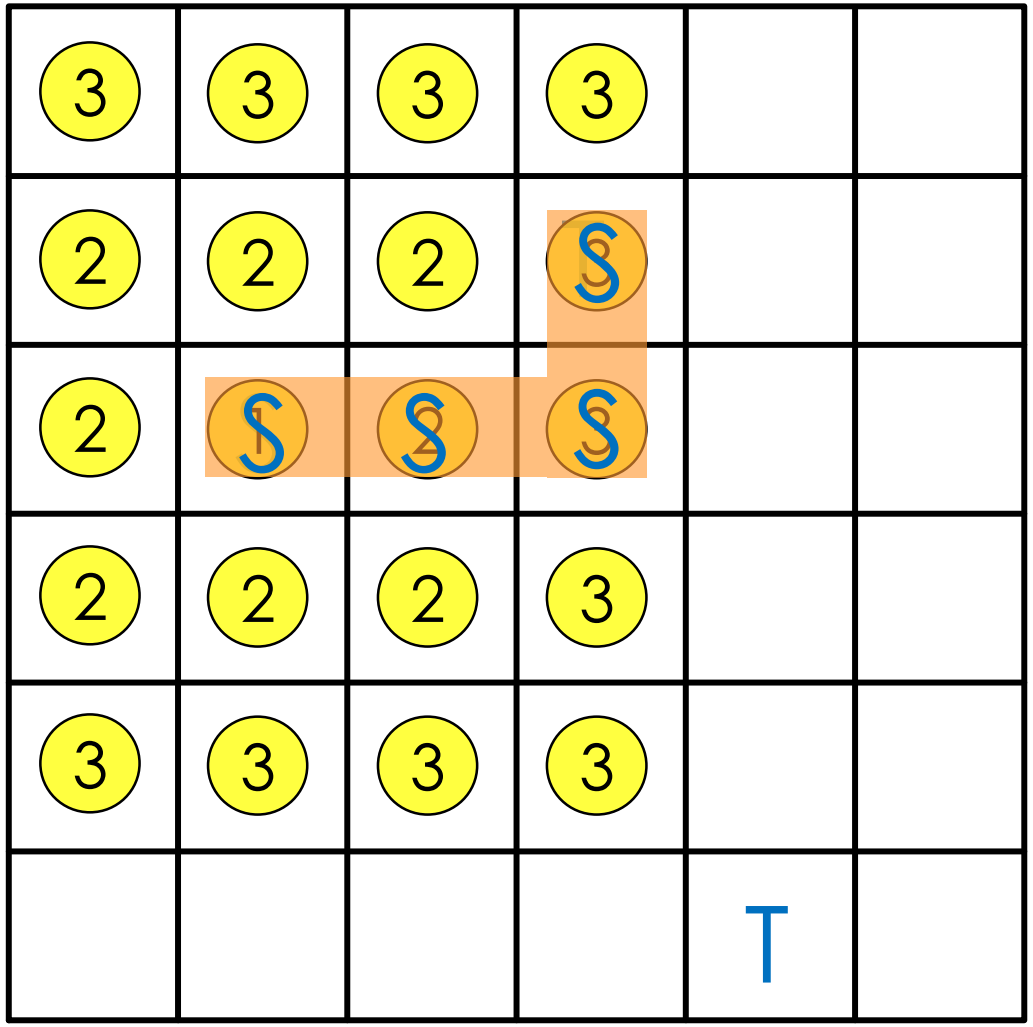
To summarize:

- **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.



Multi-Point Nets

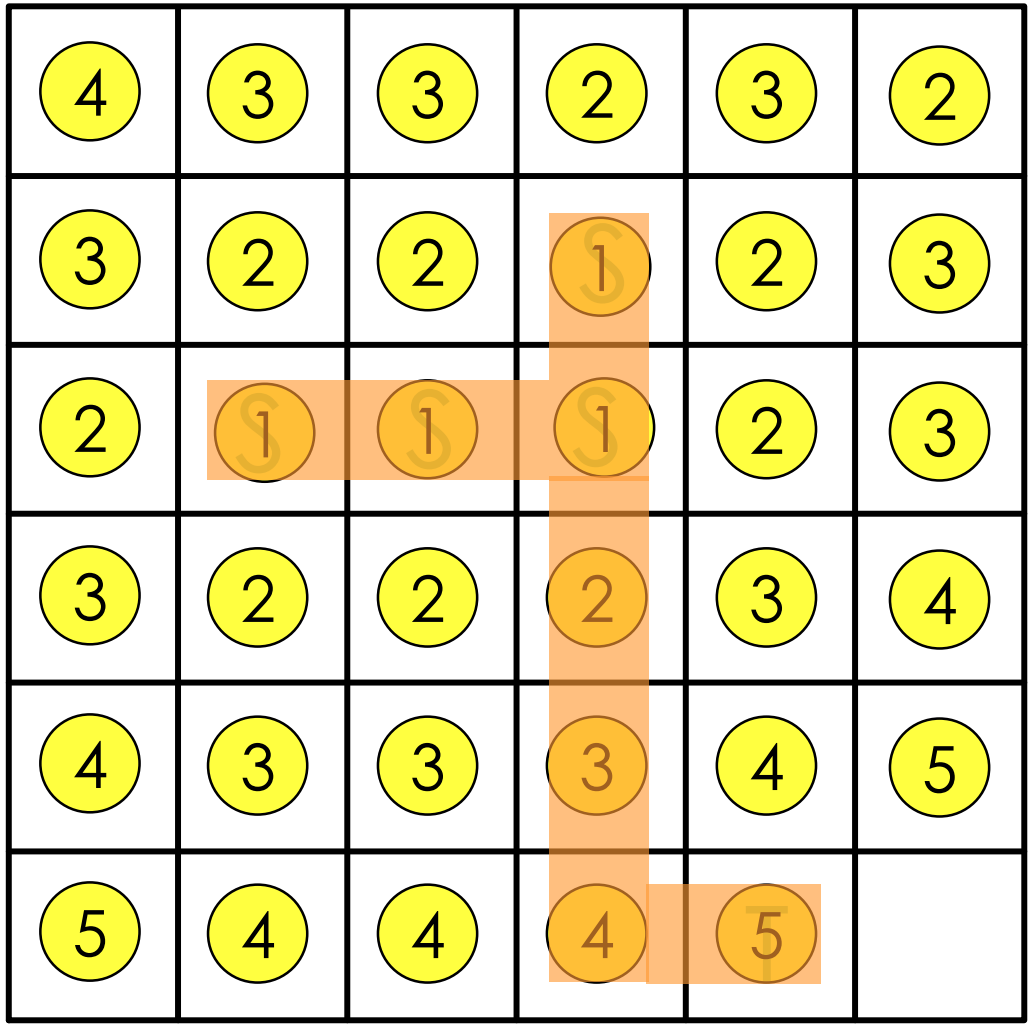
- **How do we go about routing a net with multiple targets?**
  - Actually, pretty straightforward.
  - Start with our regular maze routing algorithm to find the path to the nearest target.
  - Then re-label *all cells* on found path as *sources*, and re-run maze router using all sources simultaneously.



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.

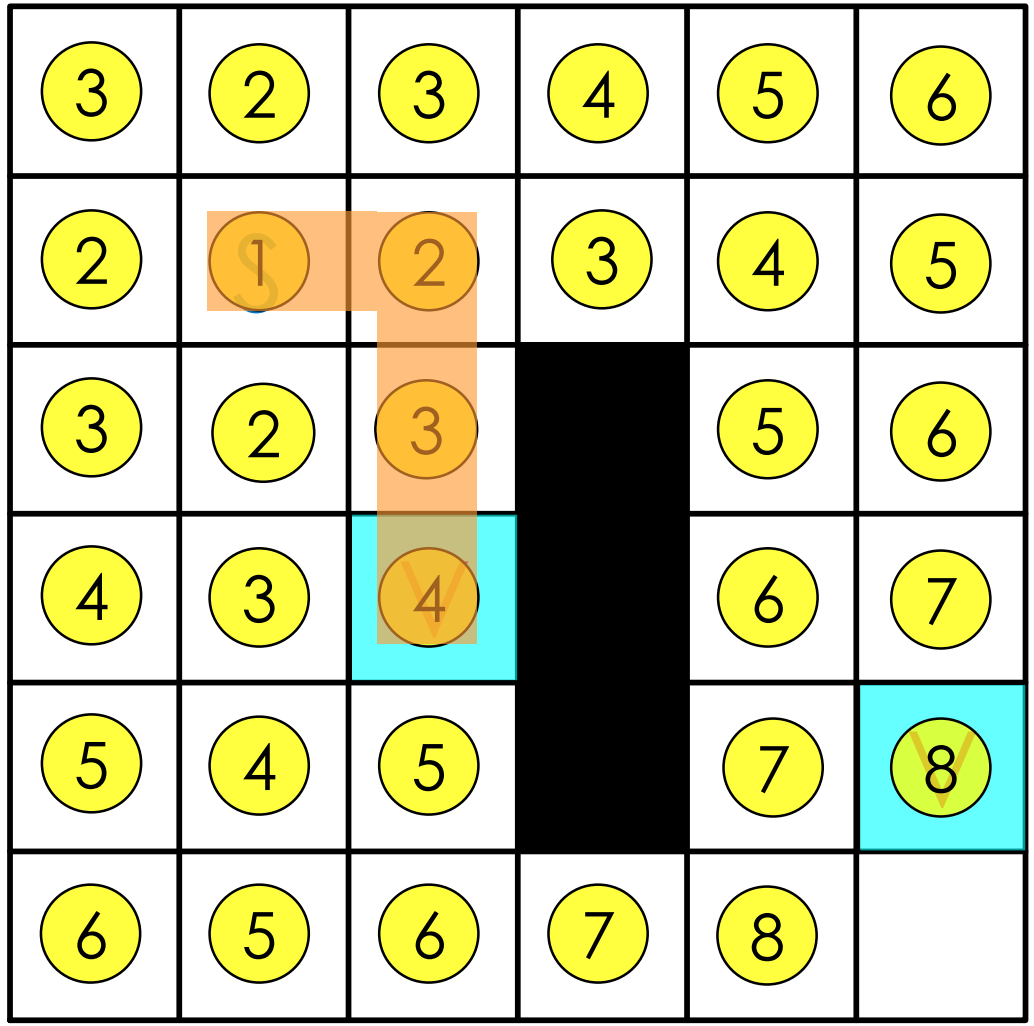
Note that this does not *guarantee* the shortest path (="Steiner Tree")



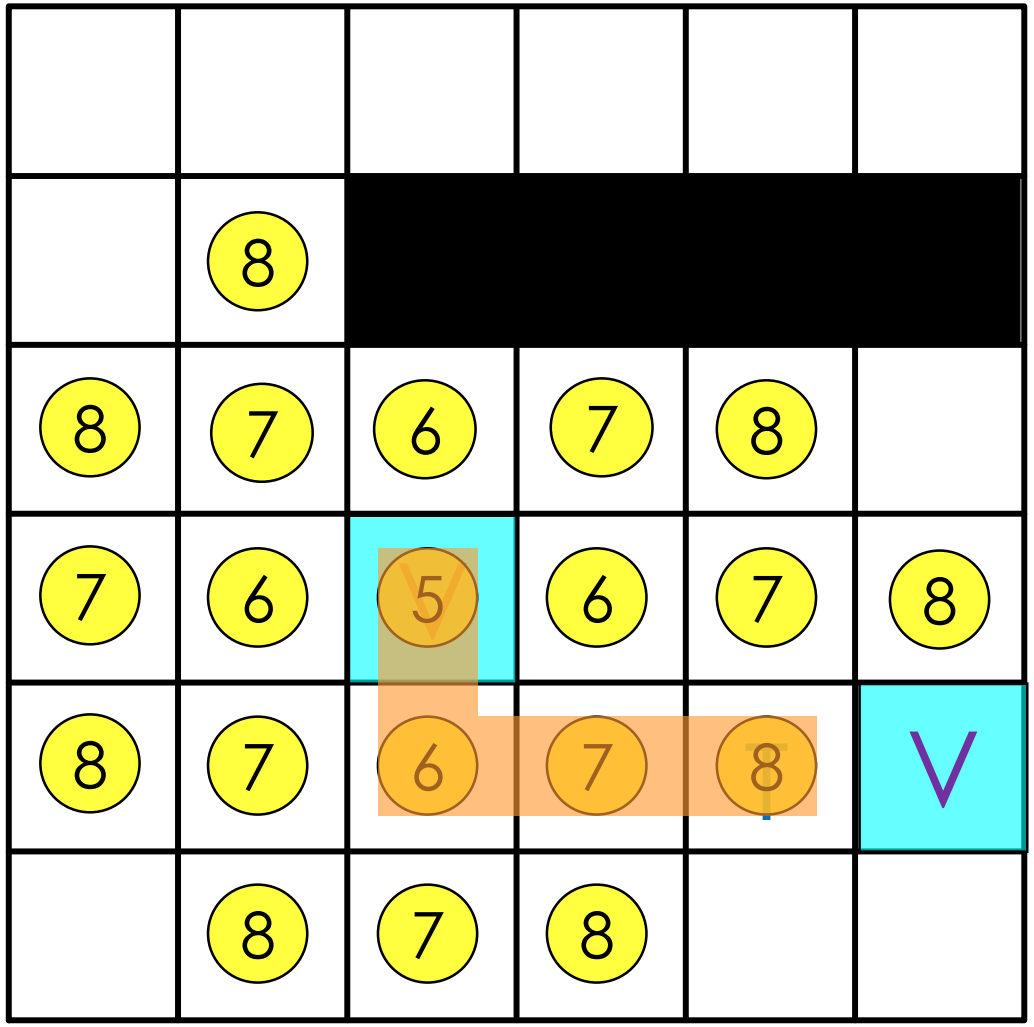
Multi-Layer Routing

- **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.

Multi-Layer Routing



Metal 1



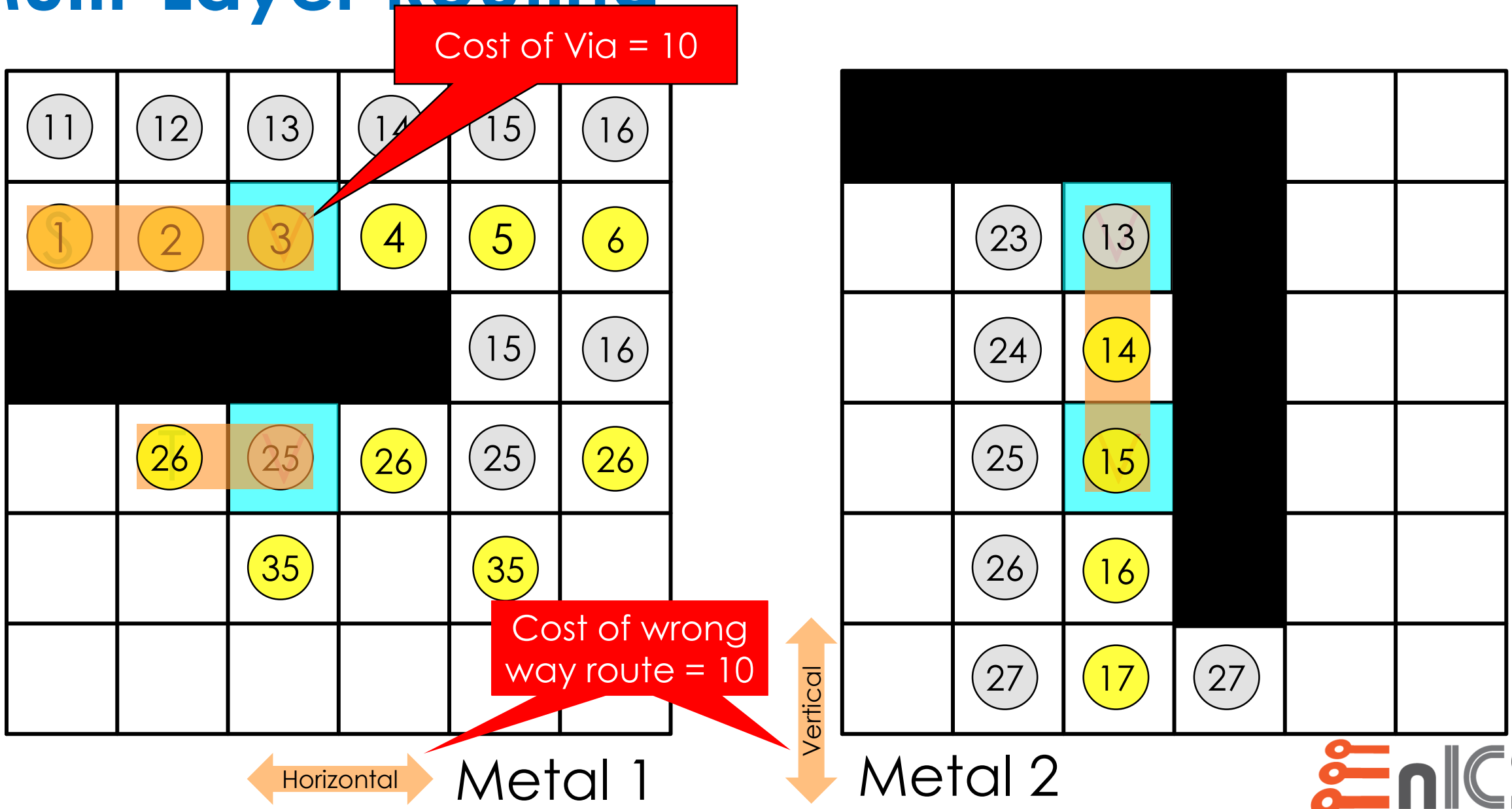
Metal 2

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*.

Multi-Layer Routing



How do we implement this?

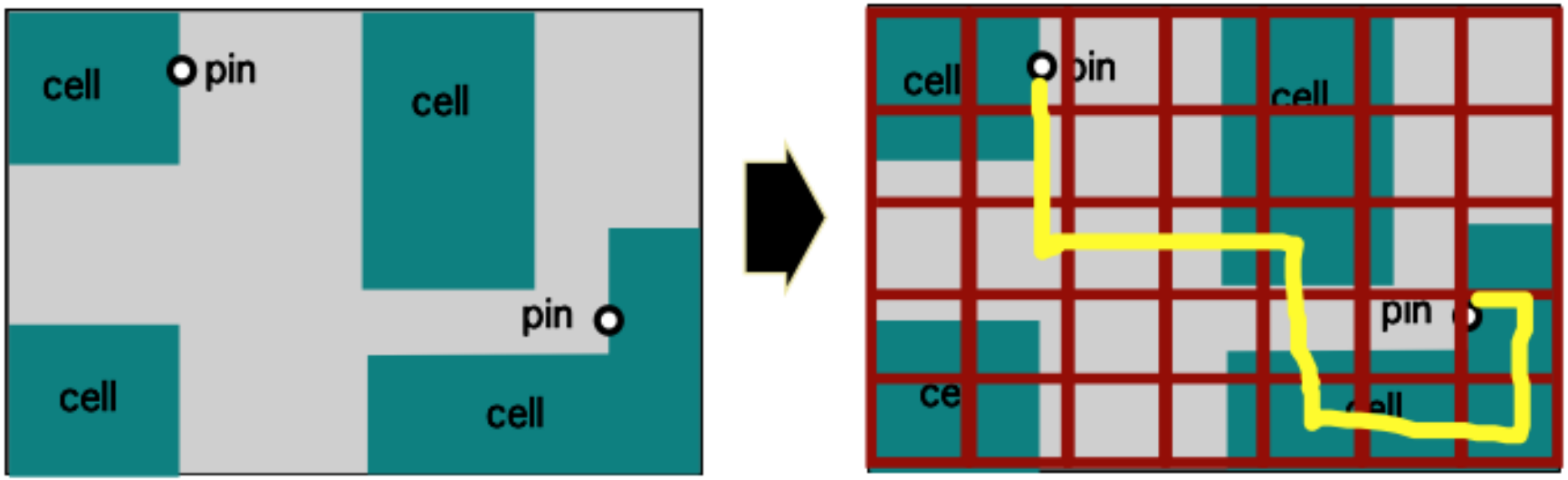
- **Grids are *huge*.**
  - Assume 1cm X 1cm chip.
  - Assume 100 nm track
  - Assume 10 routing layers
  - That is 10^{10} (100 billion) grid cells!
- **We need a low cost representation**
  - Only store the *wavefront*.
  - Remember which cells have been *reached*, at what *cost*, and from *which direction*.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

All of this is *hard*!

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

Divide and Conquer: Global Routing

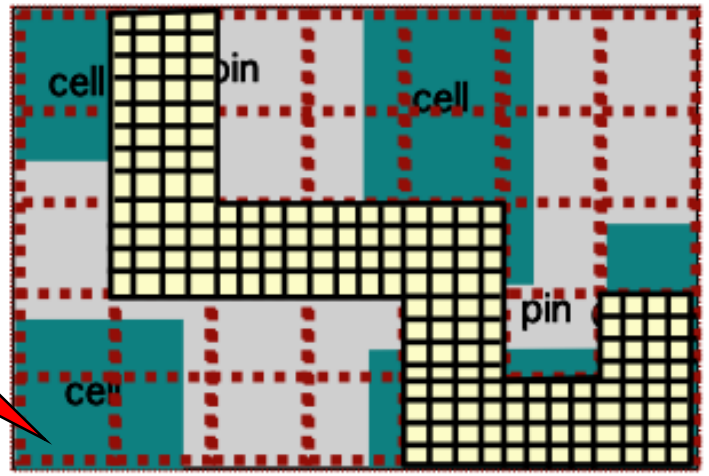
- **To deal with a big chip, we make our problem smaller**
  - Divide the chip into big, course regions
  - E.g., 200 X 200 tracks each.
  - These are called GBOXes.
- **Now Maze Route through the GBOXes**



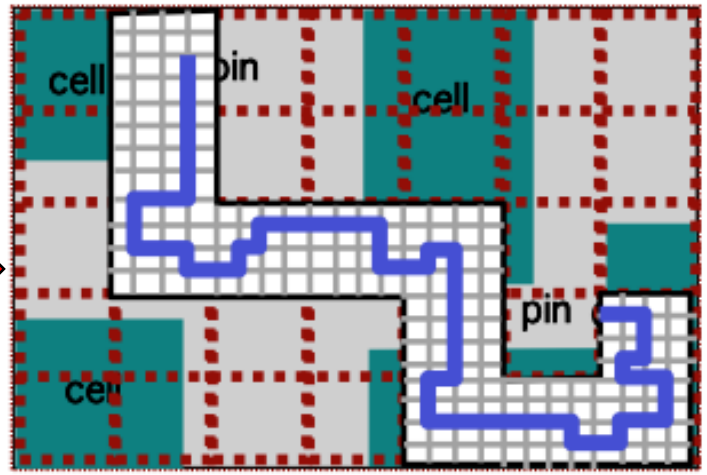
Divide and Conquer: Global Routing

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

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



Global router tells us to search for detailed paths only here



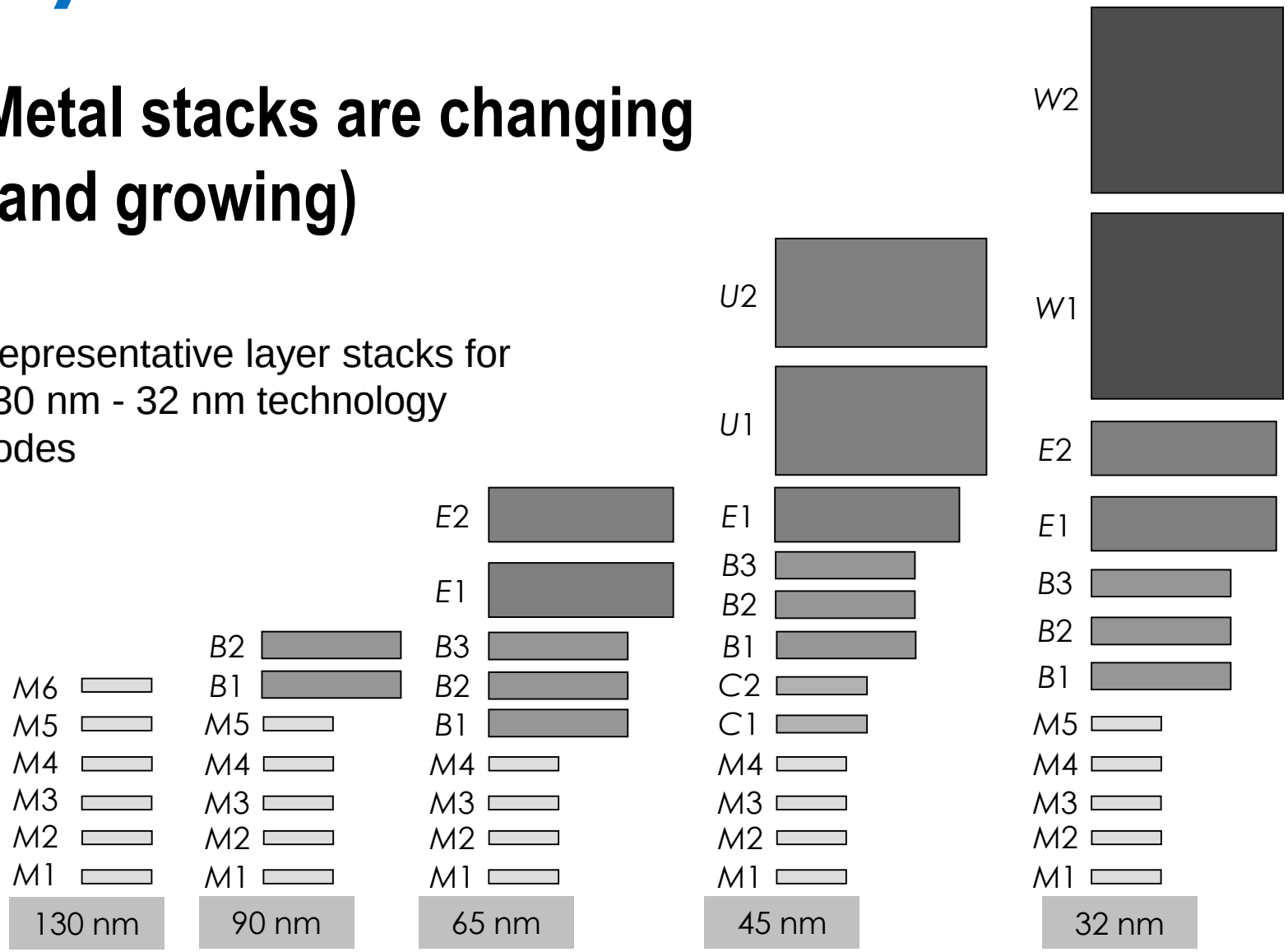
Detailed router tells us exact final path in this region

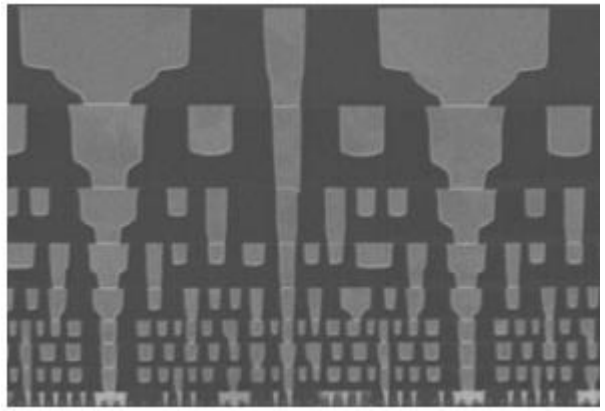
Routing in practice

Layer Stacks

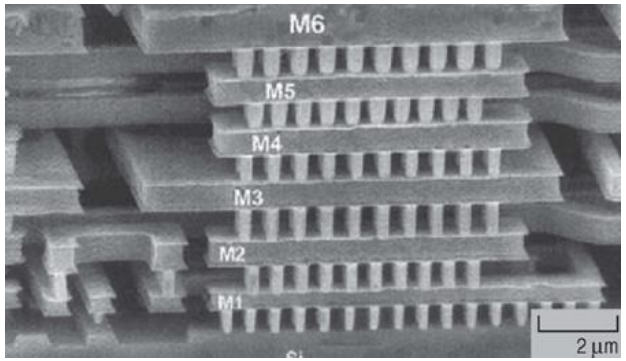
- **Metal stacks are changing (and growing)**

Representative layer stacks for 130 nm - 32 nm technology nodes





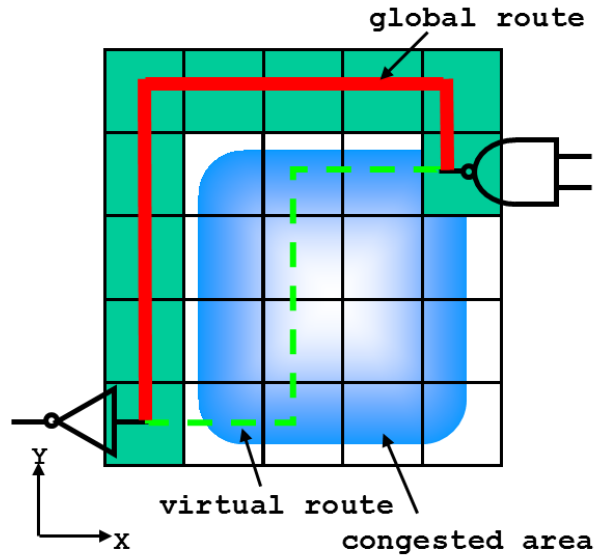
Intel 45nm 8 metal stack

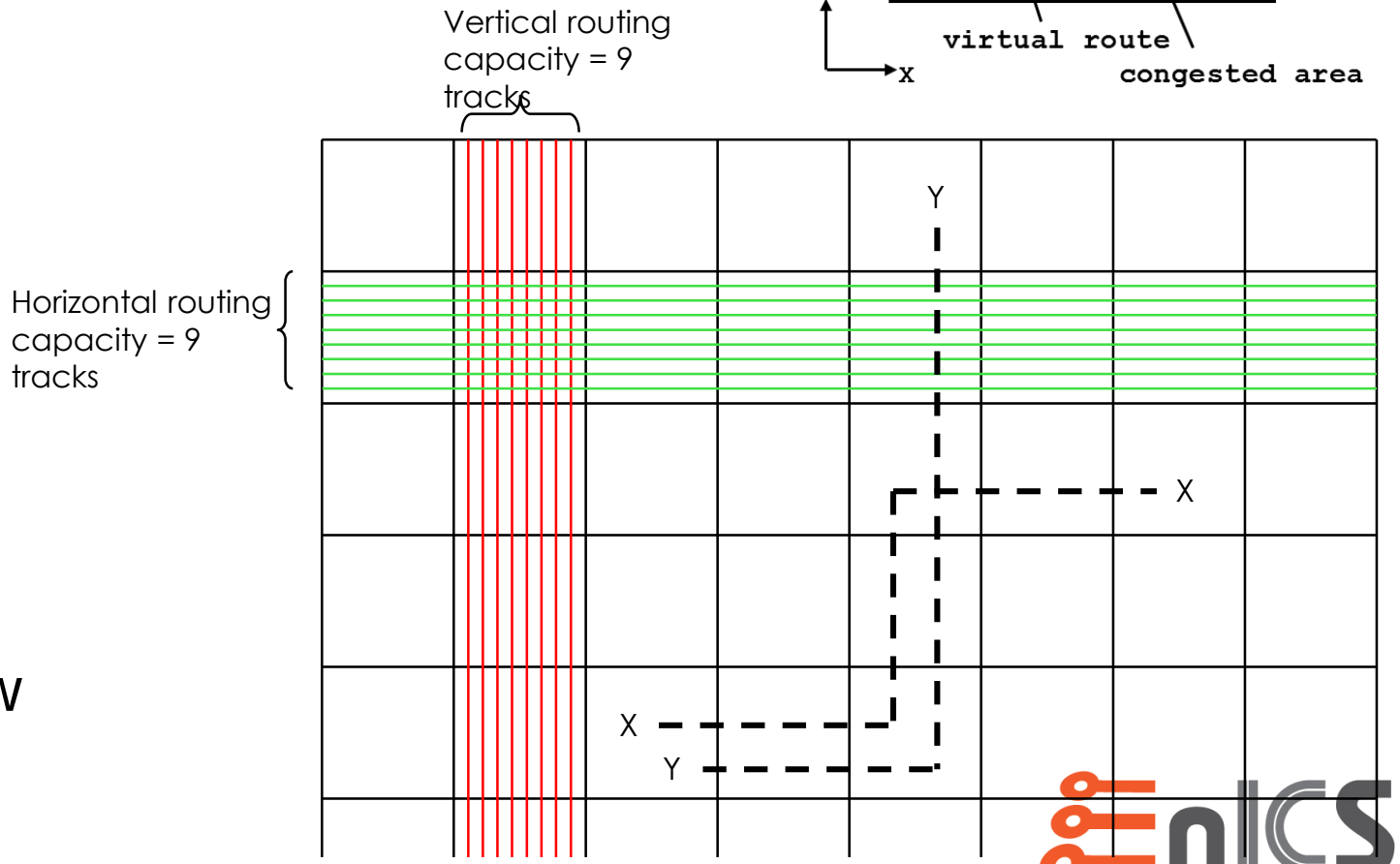


UMC 6 metal stack

Global Route

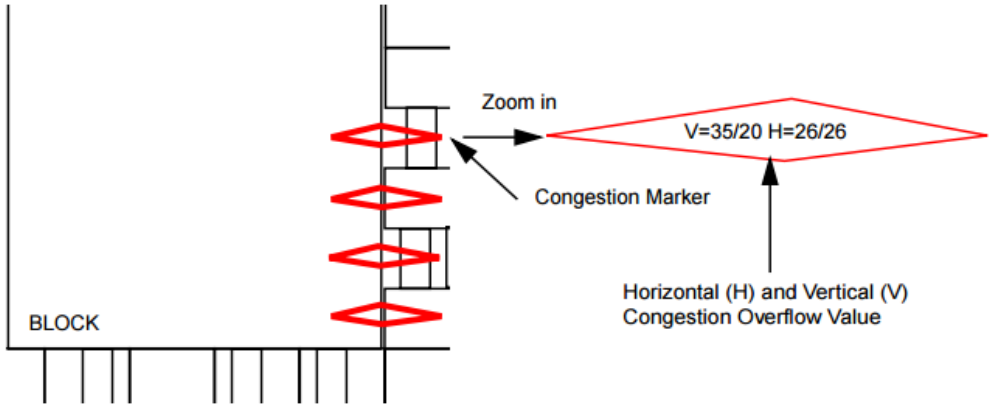
- **Divide floorplan into GCells**
  - Approximately 10 tracks per layer each.
- **Perform fast grid routing:**
  - Minimize wire-length
  - Balance Congestion
  - Timing-driven
  - Noise/SI-driven
  - Keep buses together
- **Also used for trial route**
  - During earlier stages of the flow



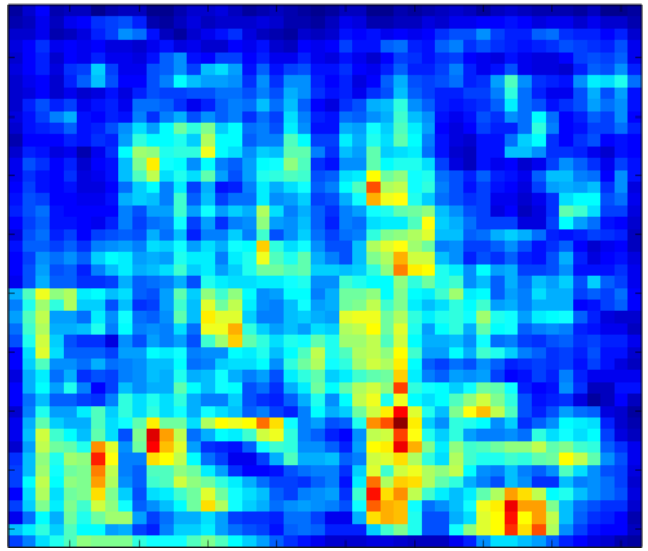


Congestion Map

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



Congestion map



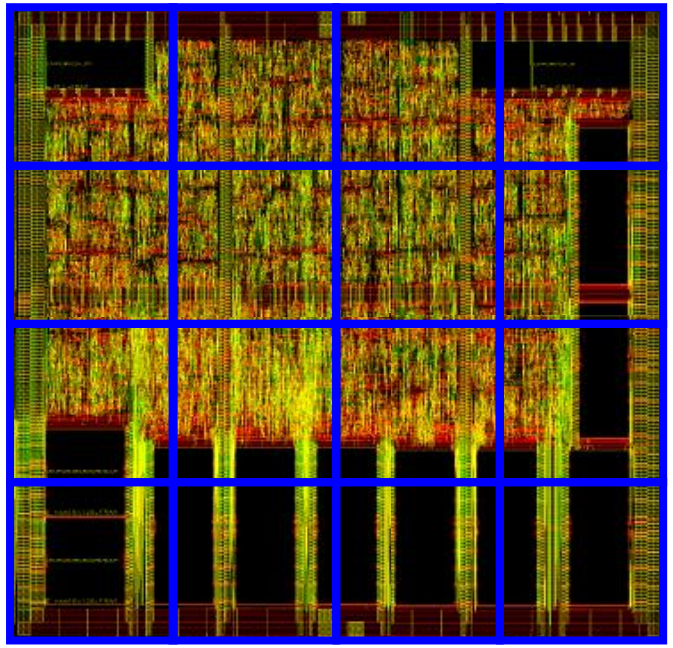
Congestion Report

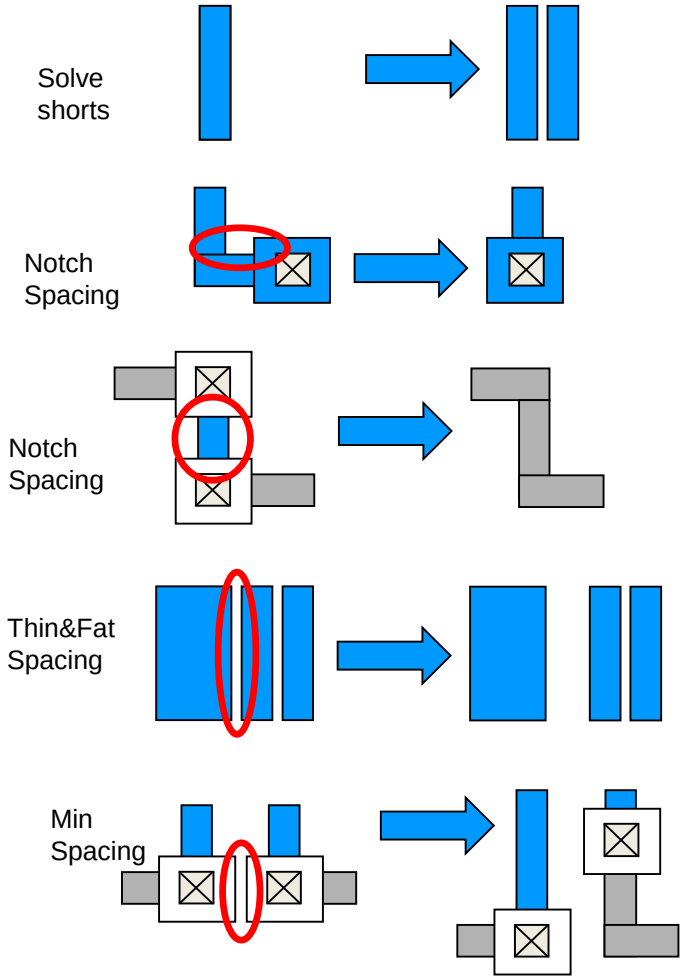
```
#                Routing   #Avail      #Track      #Total      %Gcell
#  Layer       Direction    Track     Blocked       Gcell     Blocked
#  ---------------------------------------------------------------------
#  Metal 1             H     7607        9692     1336335      62.57%
#  Metal 2             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.
```

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)

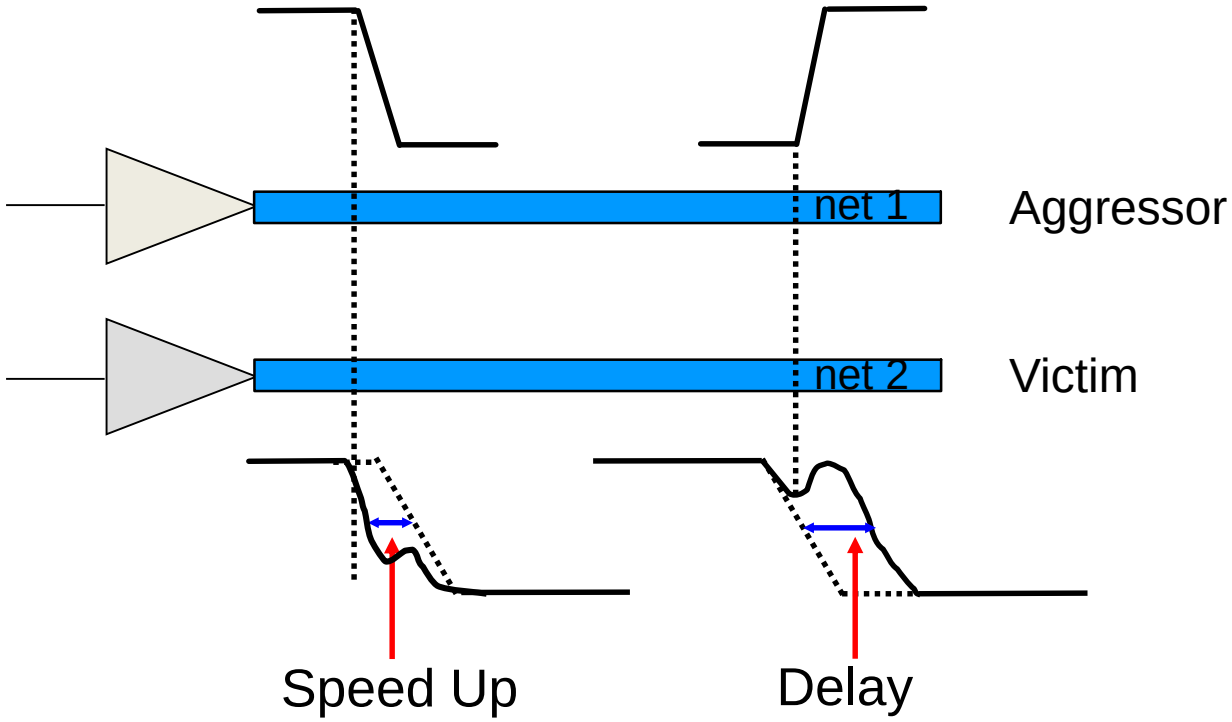
Detailed Route Boxes





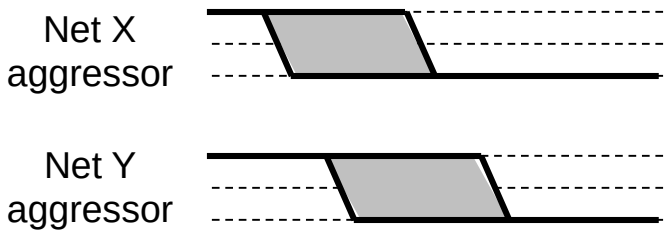
Signal Integrity (SI)

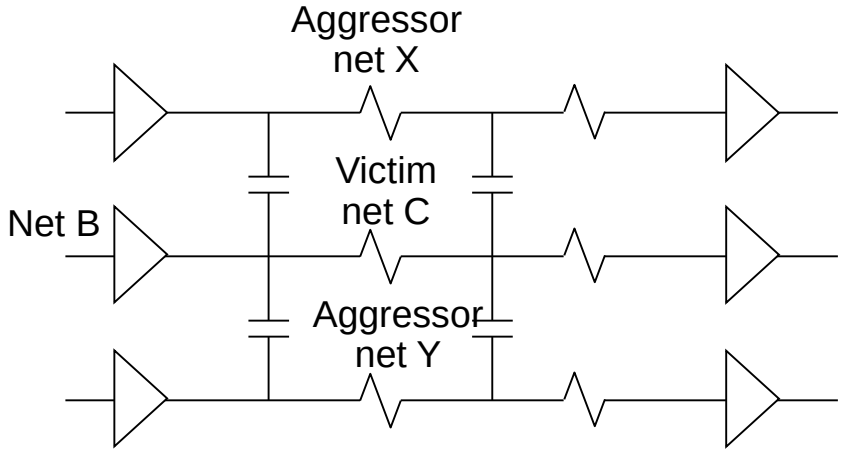
- **Signal Integrity during routing is synonymous with *Crosstalk*.**
  - A switching signal may affect a neighboring net.
  - The switching net is called the *Aggressor*.
  - The affected net is called the *Victim*.
- **Two major effects:**
  - Signal slow down
    - When the aggressor and victim switch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



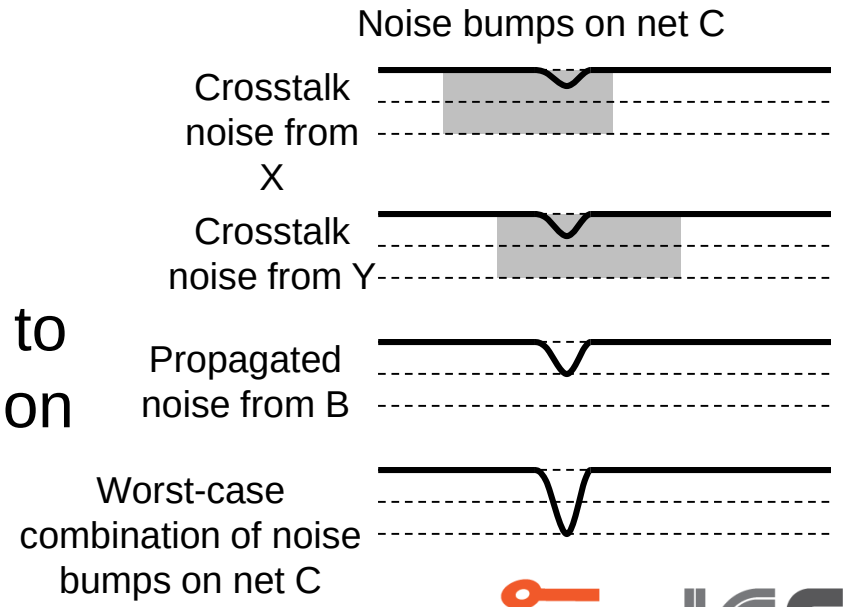
SI Multi-Aggressor Timing Analysis

- **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.



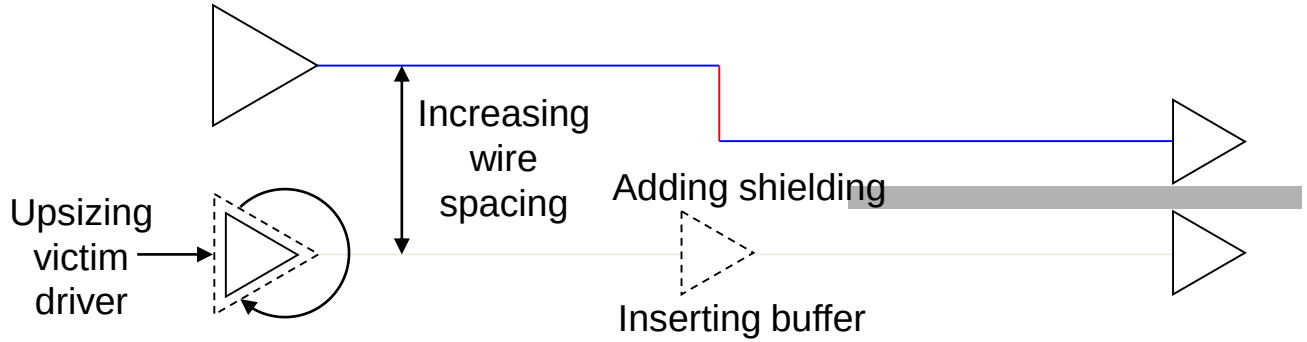


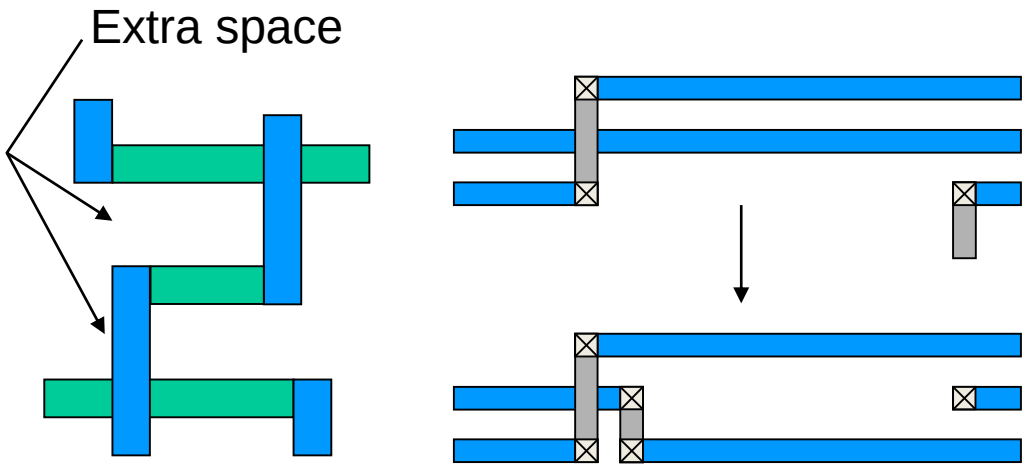
- **Propagated Noise Analysis**
  - Min/Max vectors are propagated through the design to create a transition window for all aggressors in relation to a certain victim.
  - Noise is only applied at the overlap of the two windows to determine the worst case noise bump.



Signal Integrity - Solutions

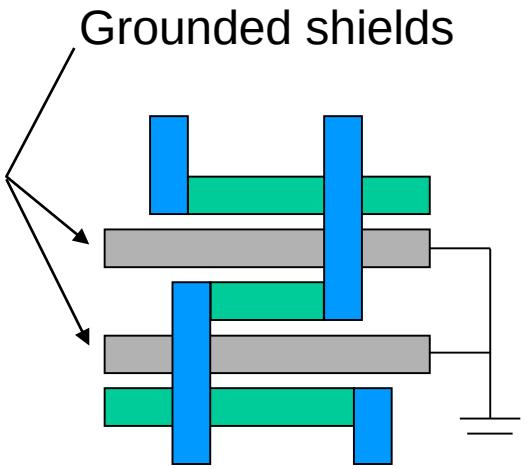
- **Crosstalk Prevention**
  - Limit length of parallel nets
  - Wire spreading
  - Shield special nets
  - Upsize driver or buffer



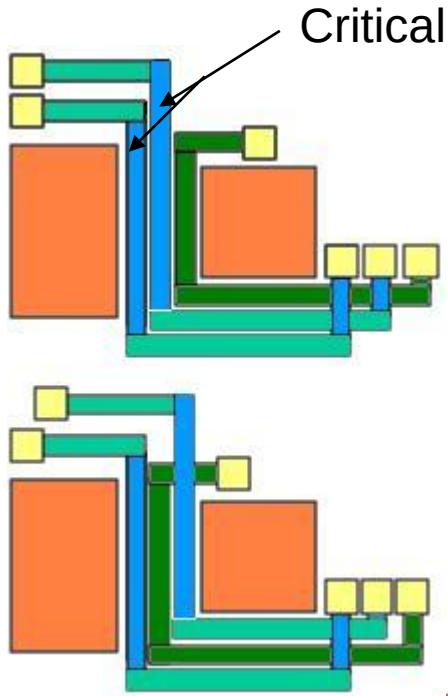


Spacing

Wire Spreading



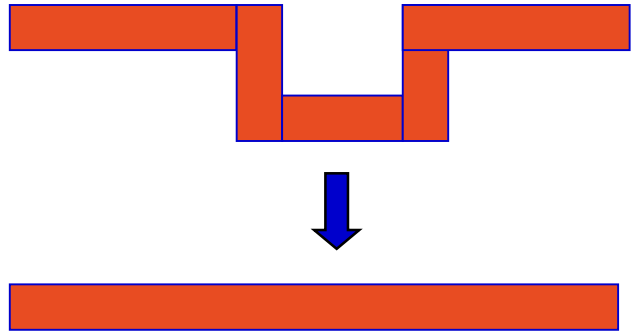
Shielding
Same layer (H)
Adjacent layers (V)



Net Ordering

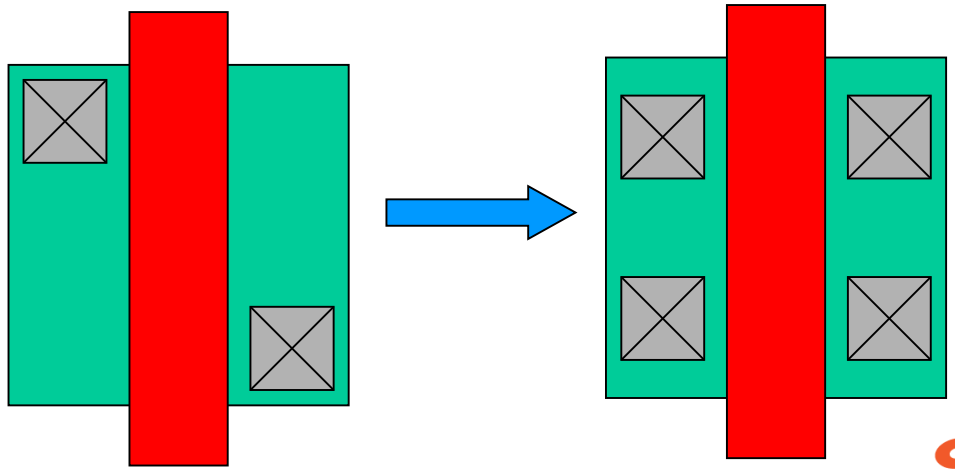
Design For Manufacturing

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



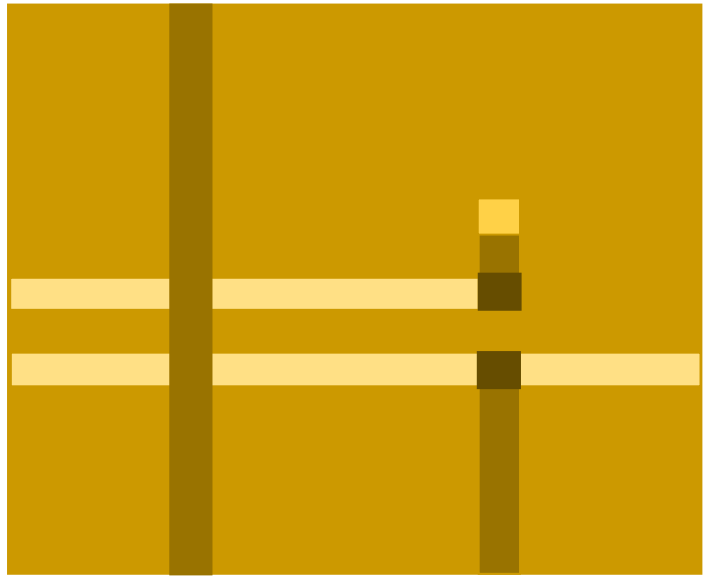
Wire straightening (reduce jogs)

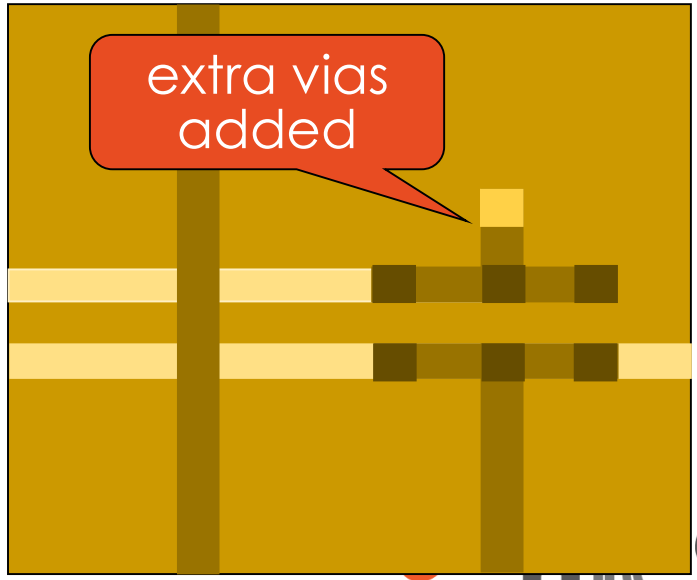
Avoid asymmetrical contacts



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.

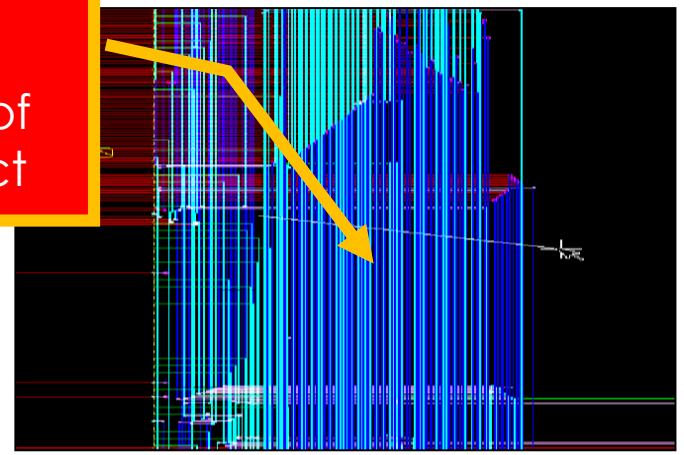


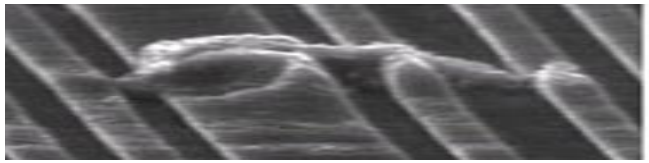


Wire Spreading

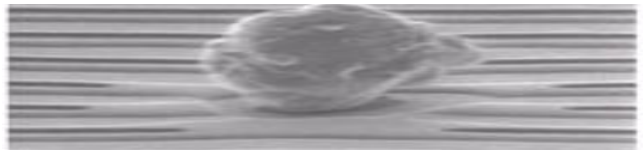
- **Wire spreading achieves:**
  - Lower capacitance and better signal integrity.
  - Lower susceptibility to shorts or opens due to random particle defects.

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

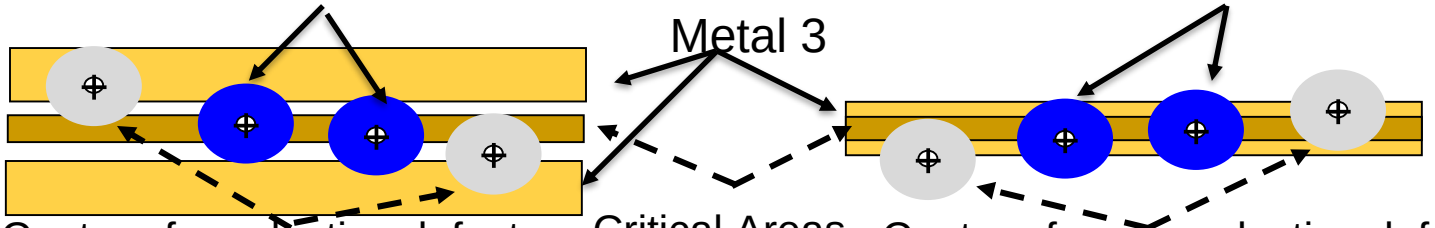




Center of conductive defects within critical area – causing shorts



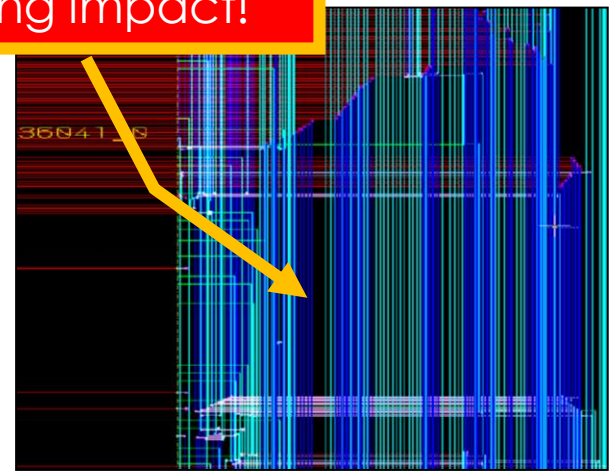
Center of non-conductive defects within critical area – causing opens



Center of conductive defects outside critical area – no shorts

Center of non-conductive defects outside critical area – no opens

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



Sign-off Timing

Advanced Concepts in Static Timing Analysis

Additional Timing Margins

- **Remember those pesky timing margins, we mentioned in the lecture about static timing analysis?**

$$T + \delta_{\text{skew}} > t_{CQ} + t_{\text{logic}} + t_{SU} + \delta_{\text{margin}}$$

$$t_{CQ} + t_{\text{logic}} - \delta_{\text{margin}} > t_{hold} + \delta_{\text{skew}}$$

  - Well, we discussed *skew* and *jitter*, but is that all?
  - If it was, there's a good chance that all the CAD engineers could retire…
- **So what/why/how do we apply additional timing margins?**

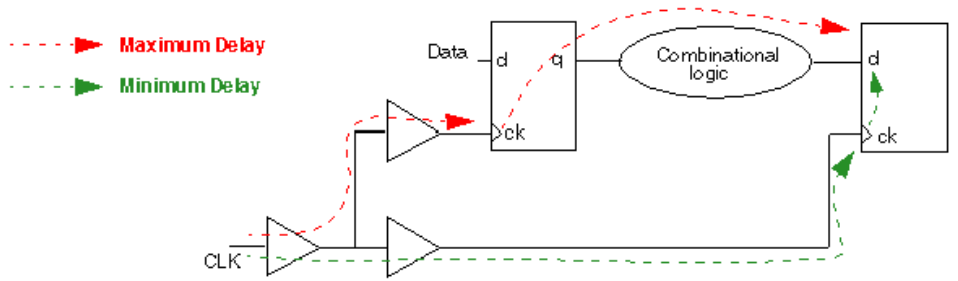
On Chip Variation

- **Spatial variation**
  - Chips are “big” and delay elements can be far from each other.
  - Process/Voltage/Temperature (PVT) variation can affect different parts of the timing path in opposite directions.
  - So, why don’t we just assume the worst possible case

```
setAnalysisMode \
        -analysisType
        onChipVariation
```

- **During setup:**
  - The launch (data) path is extra slow.
  - The capture (clock) path is super fast.

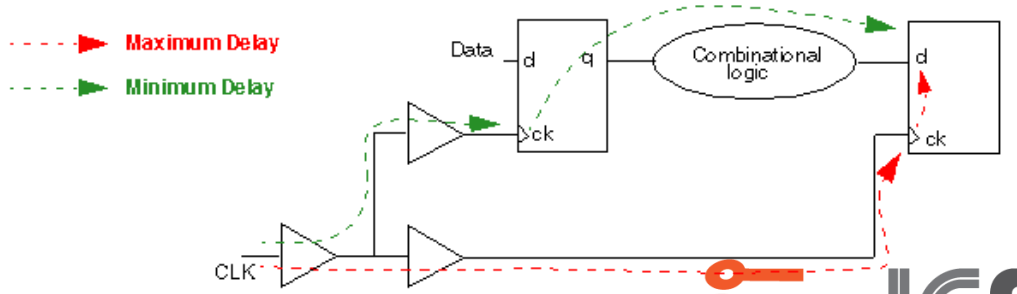
```
set_timing_derate –max -early 0.9 -late 1.2
```



Courtesy: Synopsys

- **During hold:**
  - The launch (data) path is super fast.
  - The capture (clock) path is extra slow.

```
set_timing_derate –min -early 1.2 -late 0.9
```

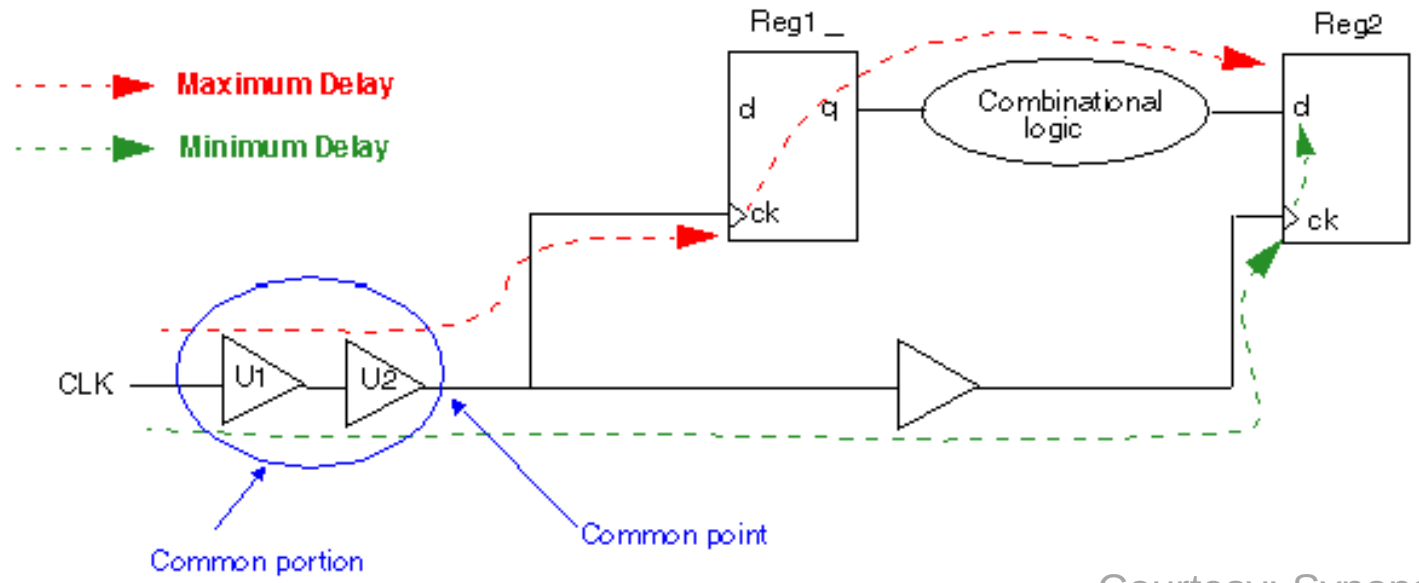


Courtesy: Synopsys

Clock Reconvergence Pessimism Removal

- **To limit the pessimism of OCV, apply CRPR**
  - This basically removes the derating from the clock path shared by both the launch and capture paths.

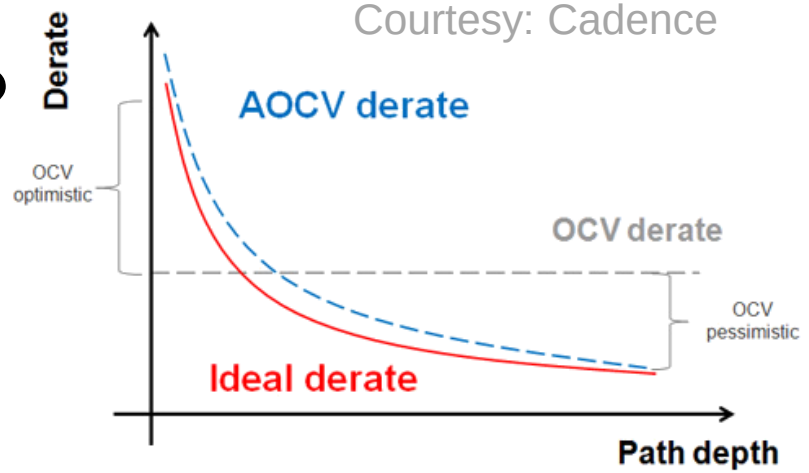
```
setAnalysisMode -analysisType onChipVariation –crpr both
```



Courtesy: Synopsys

Advanced on-chip variation (AOCV)

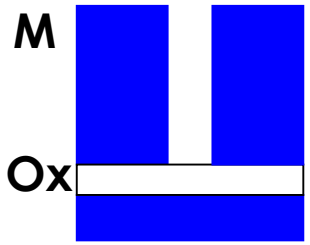
Courtesy: Cadence



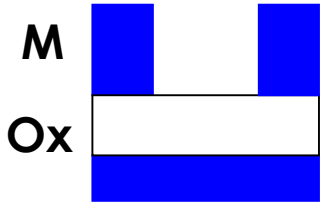
- **Well, OCV derating is still a bit over-doing it, no?**
- **So let's introduce *Advanced OCV*:**
  - For long paths, the local variation averages out, and therefore, the longer the path, the less variation.
  - So let's apply a model based on the path length and give it a new name!
- **Is that enough?**
  - Of course not!
  - Location based OCV adds less variation to closer elements.
  - Parametric OCV (POCV) provides more accuracy based on statistical libraries.
  - Or let's go for the real target – Statistical Static Timing Analysis (SSTA)

RC Extraction

- **Usually done with an external tool providing several extraction options:**
  - Cworst
    - largest cell delay, small R
  - Cbest
    - smallest cell delay, large R
  - RCbest
    - medium C, small R
  - RCworst
    - medium C, large R

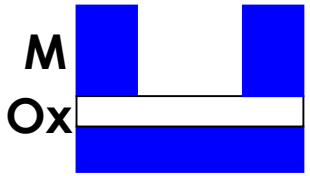


Cworst



Cbest

RCbest

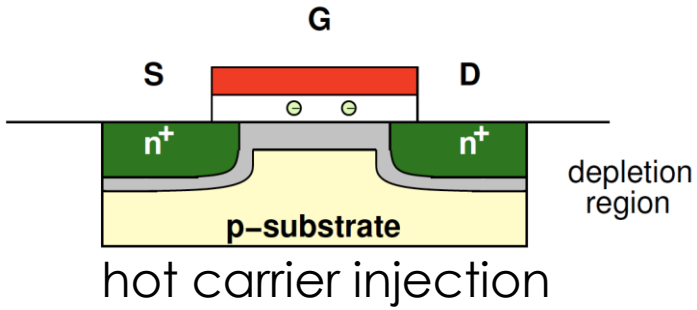


RCworst

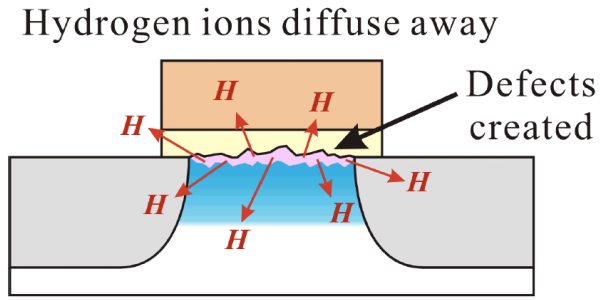
- **Which one to use?**
  - Only C actually affects the delay…
  - Maybe use Cworst for setup, Cbest for hold.

A note about Aging

- **Another big issue in modern VLSI design is aging**
  - Device characteristics change over time.
  - Hot Carrier Injection (HCI)
  - Negative Bias Temperature Instability (NBTI)
  - Time Dependent Dielectric Breakdown (TDDB)
  - Electromigration



hot carrier injection



NBTI

- **Dealing with aging effects:**
  - Operate with low VDD.
  - Model aging as an additional timing margin.
  - Use aged library models for signoff timing.
  - Add aging sensors and adjust frequency/voltage for compensation.

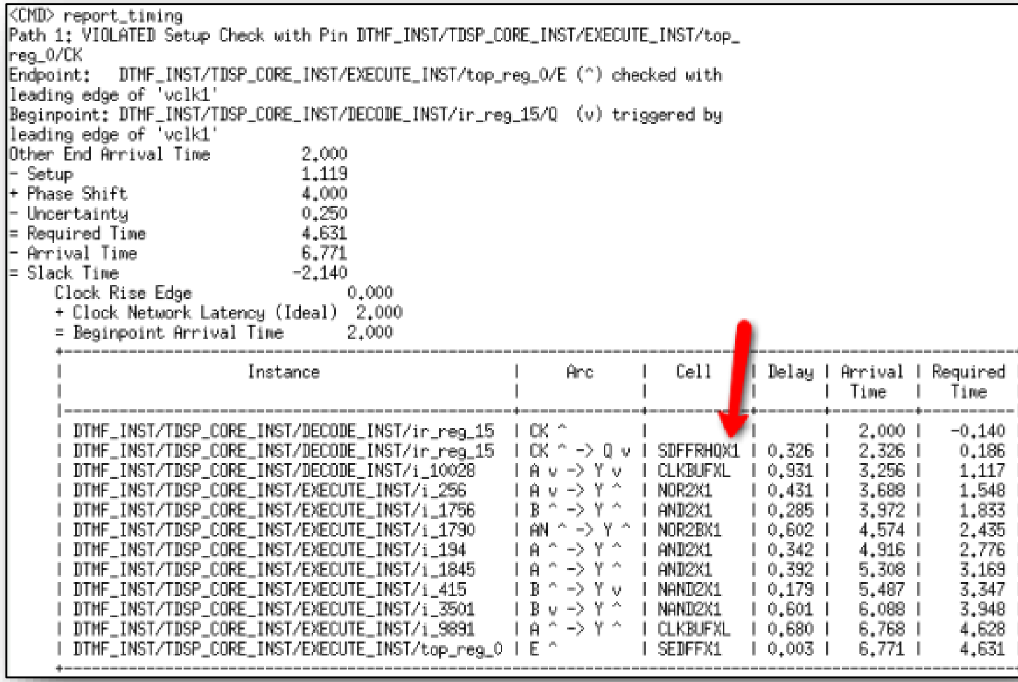
General Sign-off Timing Flow

- **Use integrated signoff timing**

```
timeDesign -signoff -outDir final_setup
timeDesign -hold -outDir final_hold
```

- **Export design to sign-off extraction tool**
  - Netlist, GDS
- **Load parasitics into sign-off timing tool**
  - SPEF for each corner

```
spefIn rc_corner1.spef -rc_corner rc_corner1
```



```
<CMD> report_timing
Path 1: VIOLATED Setup Check with Pin DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/top_
reg_0/CK
Endpoint:   DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/top_reg_0/E (^) checked with
leading edge of 'vclk1'
Beginpoint: DTMF_INST/TDSP_CORE_INST/DECODE_INST/ir_reg_15/Q  (v) triggered by
leading edge of 'vclk1'
Other End Arrival Time          2.000
- Setup                         1.119
+ Phase Shift                   4.000
- Uncertainty                   0.250
= Required Time                 4.631
- Arrival Time                  6.771
= Slack Time                   -2.140
     Clock Rise Edge                      0.000
     + Clock Network Latency (Ideal)      2.000
     = Beginpoint Arrival Time            2.000
     +-------------------------------------------------------------------------------------------------
     |                  Instance                     |     Arc      |   Cell     | Delay | Arrival | Required
     |                                               |              |            |       |  Time   |   Time
     |-----------------------------------------------+--------------+------------+-------+---------+---------
     | DTMF_INST/TDSP_CORE_INST/DECODE_INST/ir_reg_15 | CK ^         |            |       |  2.000  |  -0.140
     | DTMF_INST/TDSP_CORE_INST/DECODE_INST/ir_reg_15 | CK ^ -> Q v  | SDFFRHQX1  | 0.326 |  2.326  |   0.186
     | DTMF_INST/TDSP_CORE_INST/DECODE_INST/i_10028   | A v -> Y v   | CLKBUFXL   | 0.931 |  3.256  |   1.117
     | DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/i_256    | A v -> Y ^   | NOR2X1     | 0.431 |  3.688  |   1.548
     | DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/i_1756   | B ^ -> Y ^   | AND2X1     | 0.285 |  3.972  |   1.833
     | DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/i_1790   | AN ^ -> Y ^  | NOR2BX1    | 0.602 |  4.574  |   2.435
     | DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/i_194    | A ^ -> Y ^   | AND2X1     | 0.342 |  4.916  |   2.776
     | DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/i_1845   | A ^ -> Y ^   | AND2X1     | 0.392 |  5.308  |   3.169
     | DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/i_415    | B ^ -> Y v   | NAND2X1    | 0.179 |  5.487  |   3.347
     | DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/i_3501   | B v -> Y ^   | NAND2X1    | 0.601 |  6.088  |   3.948
     | DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/i_9891   | A ^ -> Y ^   | CLKBUFXL   | 0.680 |  6.768  |   4.628
     | DTMF_INST/TDSP_CORE_INST/EXECUTE_INST/top_reg_0 | E ^         | SEDFFX1    | 0.003 |  6.771  |   4.631
     +-------------------------------------------------------------------------------------------------
```

- **Run Static Timing Analysis in sign-off tool**
  - Provide SDF for ECO
  - Provide SDF for post-layout simulations (Dynamic Timing Analysis)

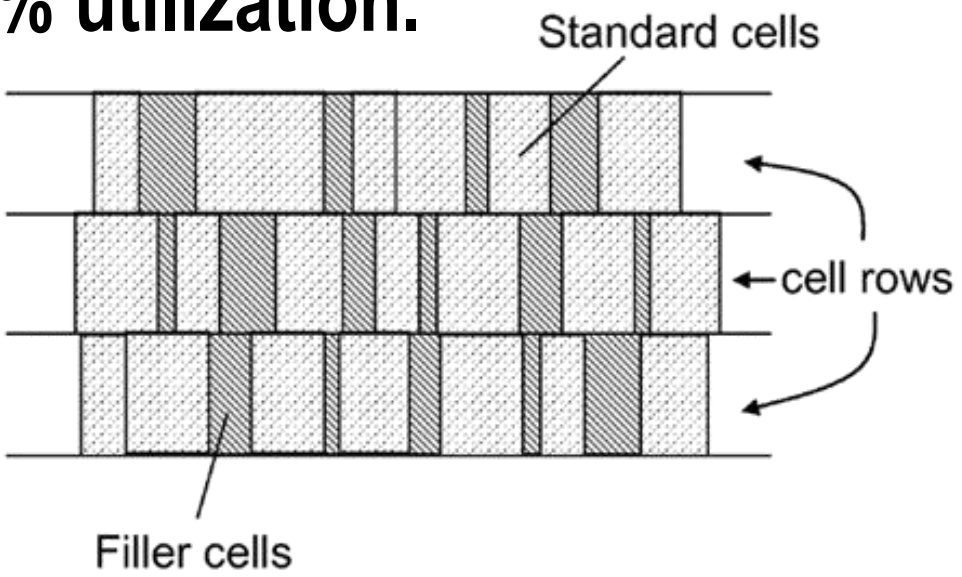
Chip Finishing and Sign-off

Chip Finishing Overview

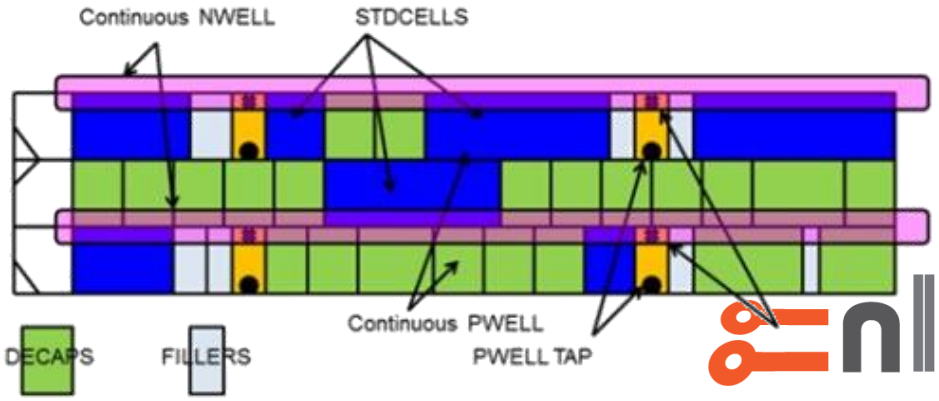
- **Chip Finishing for Signoff includes, at the very least:**
  - Insertion of fillers and DeCaps.
  - Application of Design for Manufacturing (DFM) and Design for Yield (DFY) rules.
  - Antenna checking.
  - Metal filling and slotting for metal density rules.
  - IR Drop and Electromigration Analysis
  - Logic Equivalence
  - Layout (Physical) Verification
  - Add Sealring

Filler Cell Insertion

- **Standard cell placement never reaches 100% utilization.**
- **We need to "fill in the blanks"**
  - Ensure continuous wells across the entire row.
  - Ensure VDD/GND rails (follow pins) are fully connected.
  - Ensure proper GDS layers to pass DRC.
  - Ensure sufficient diffusion and poly densities.
  - In scaled processes, provide regular poly/diffusion patterning.

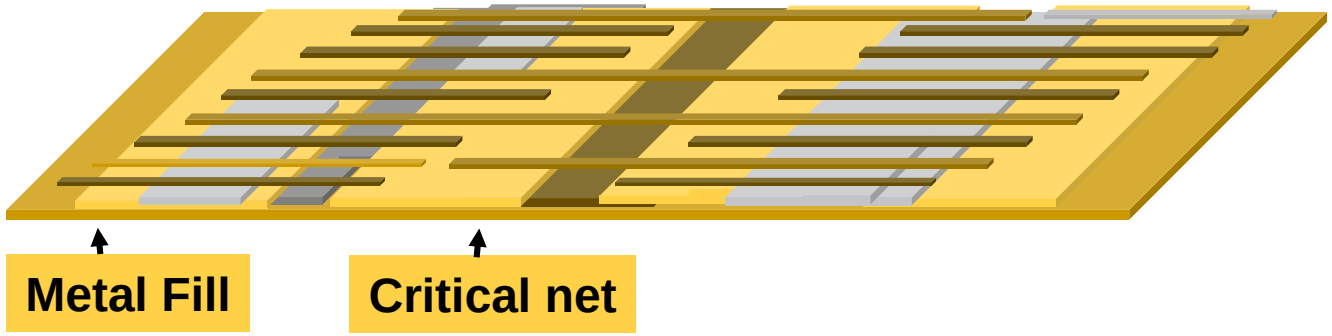


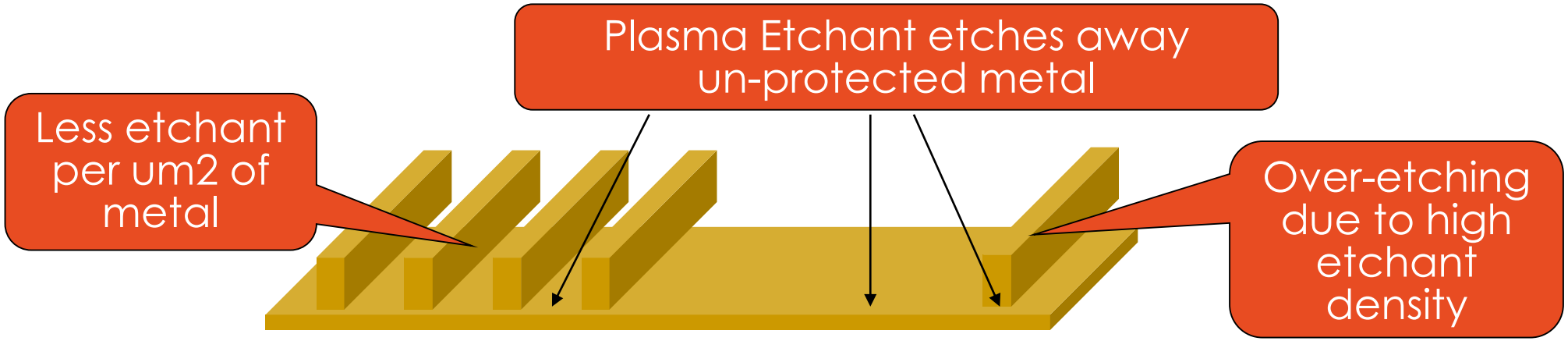
- **We can also add DeCap cells as fillers.**



Metal Density Fill

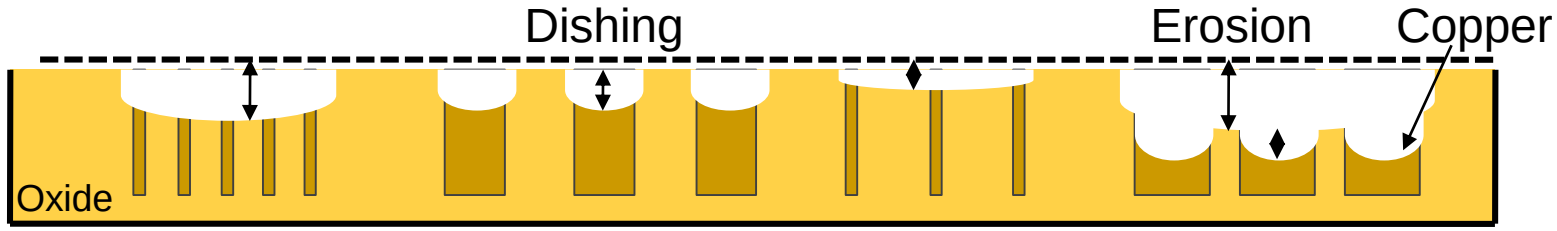
- **Density issues due to etching:**
  - A narrow metal wire separated from other metal receives a higher density of etchant than closely spaced wires, such that the narrow metal can get over-etched.
- **Solution:**
  - Minimum metal density rules
  - But, be aware of critical nets!



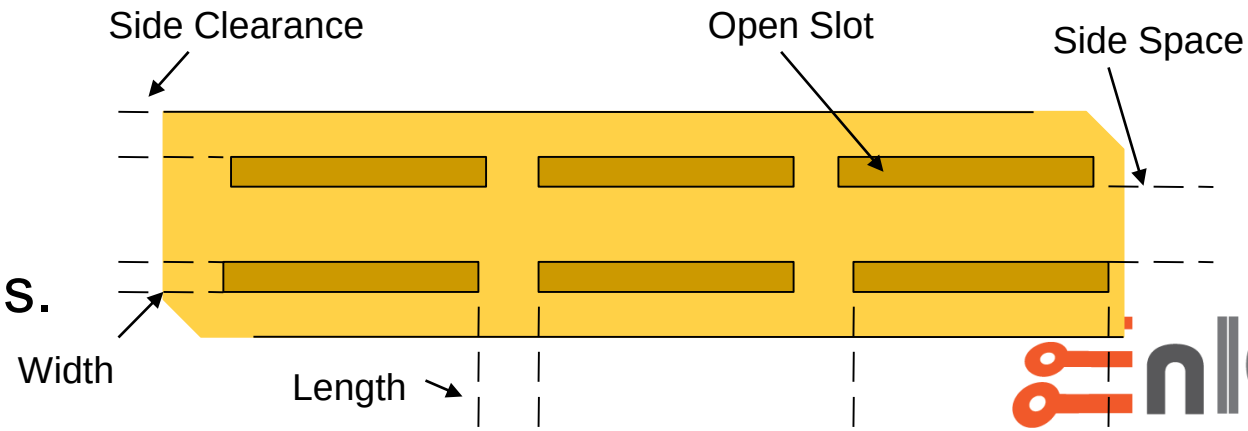


Metal Density Fill

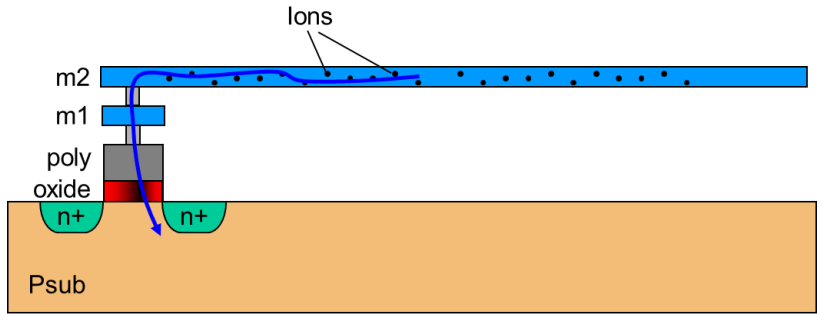
- **Density issues due to CMP:**
  - Chemical Mechanical Polishing (CMP) is the stage during which the wafer is planarized.
  - Since metals are mechanically softer than dielectrics, metal tops are susceptible to “dishing”, and very wide metals become thin (erosion).



- **Solution:**
  - Maximum metal density rules
  - Apply “Slotting”
  - Also solves “metal liftoff” problems.



Antenna Fixing

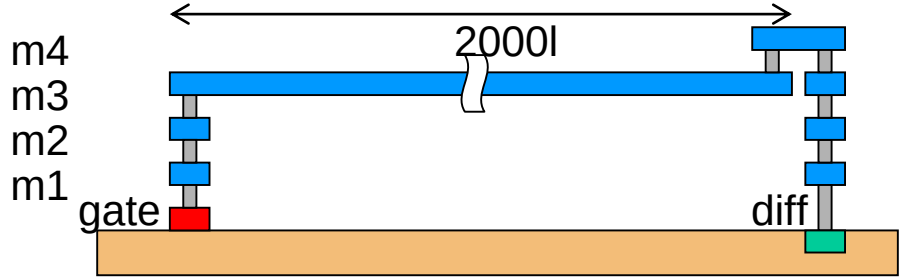


Antenna Ratios:

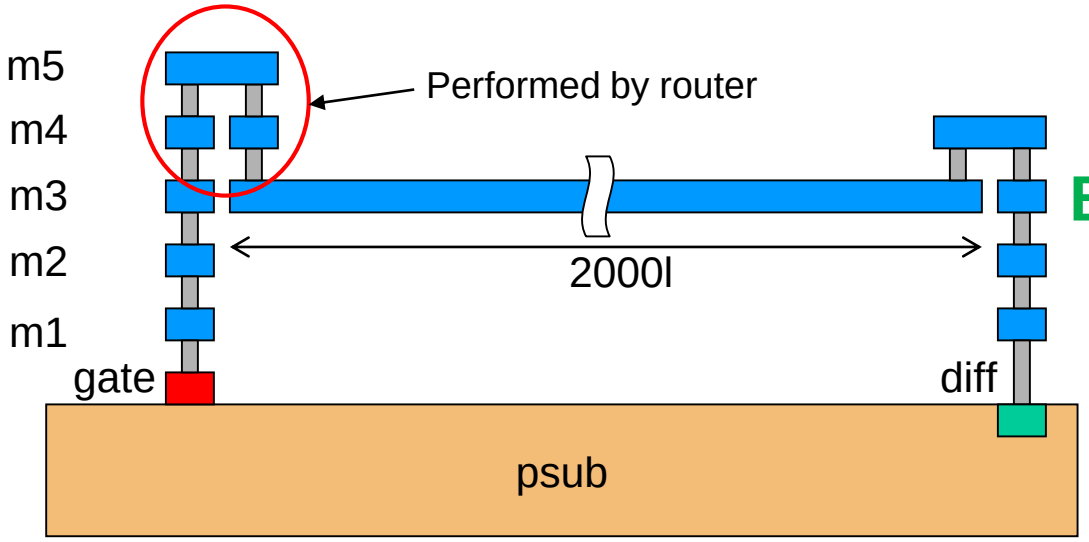
Area of Metal Connected to Gate
Combined Area of Gate
Or
Area of Metal Connected to Gate
Combined Perimeter of Gate

- **Antenna Hazards:**
  - During metal etch, strong EM fields are used to stimulate the plasma etchant resulting in voltage gradients at MOSFET gates that can damage the thin oxide
  - Antenna hazards occur when the ratio of the metal area to gate area during a process step is large.
- **Fixing Antenna Problems:**

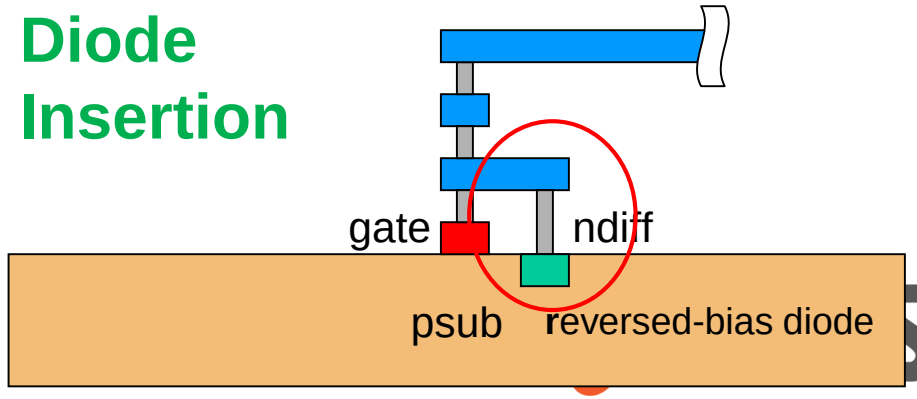
Original Topology



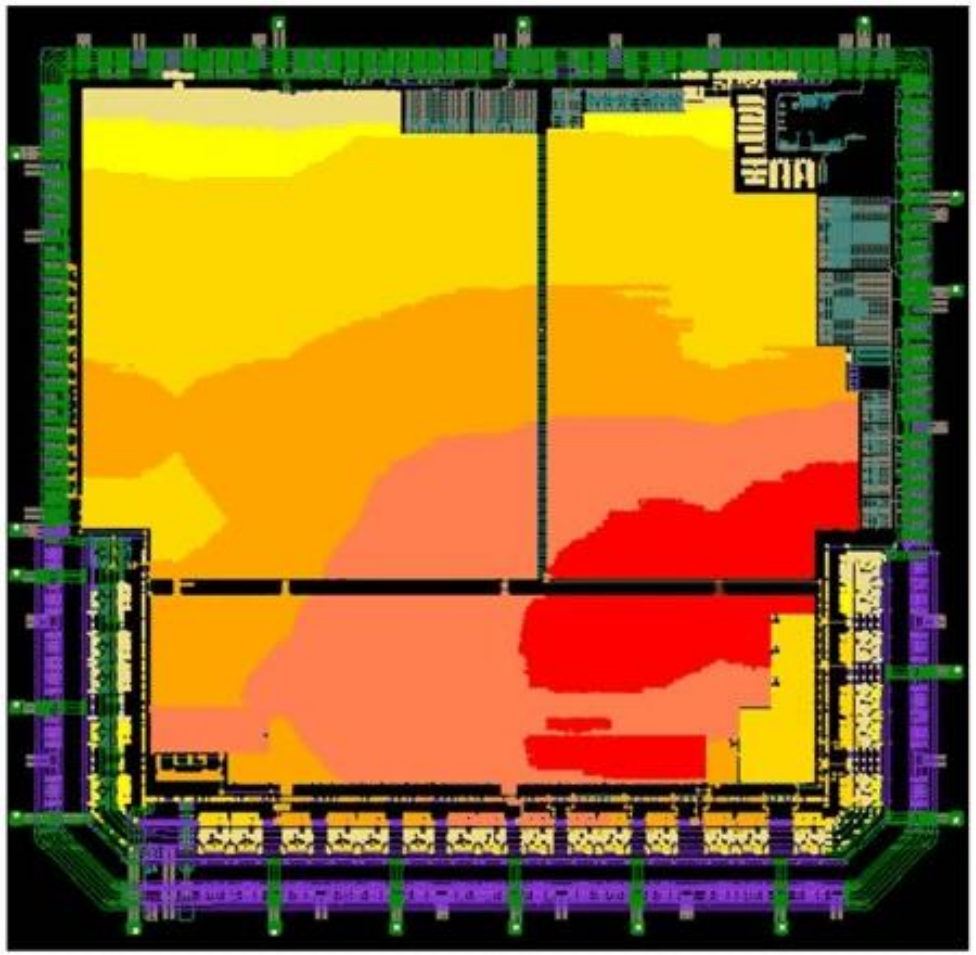
Bridging

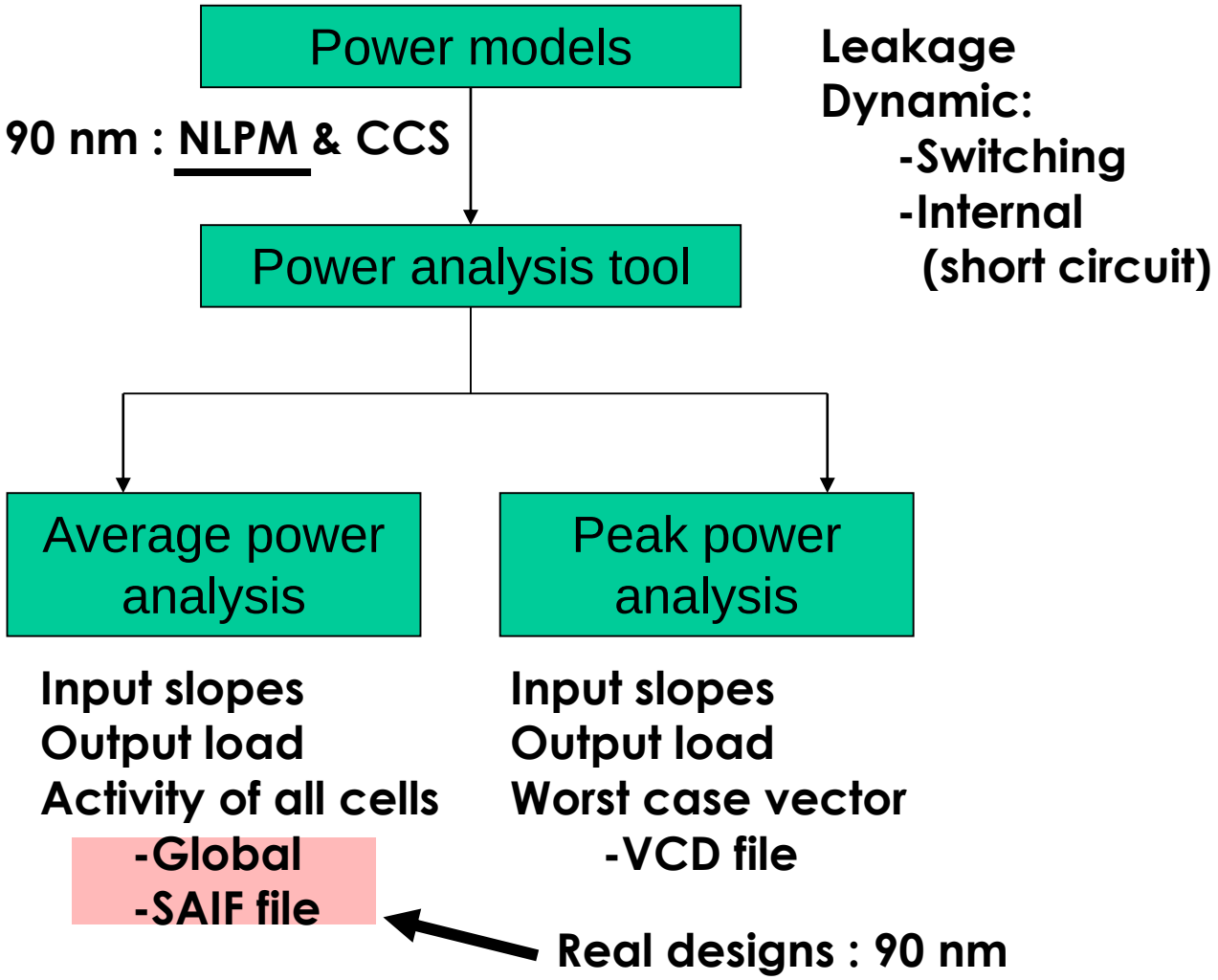


Diode Insertion

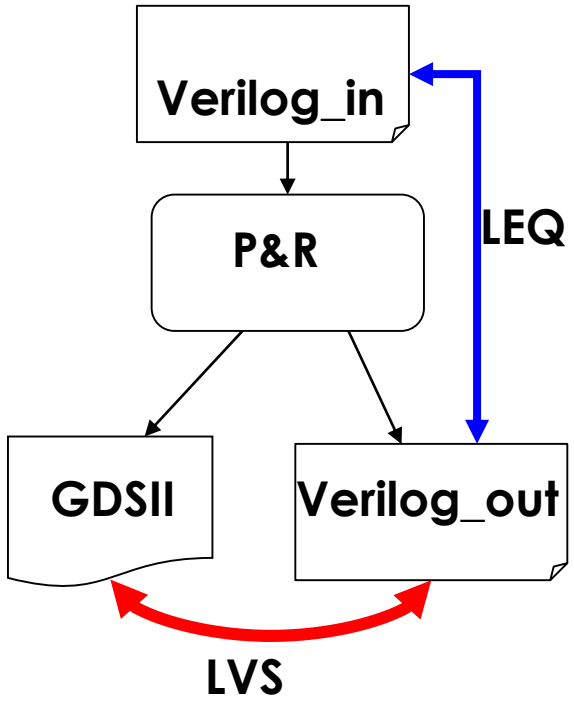


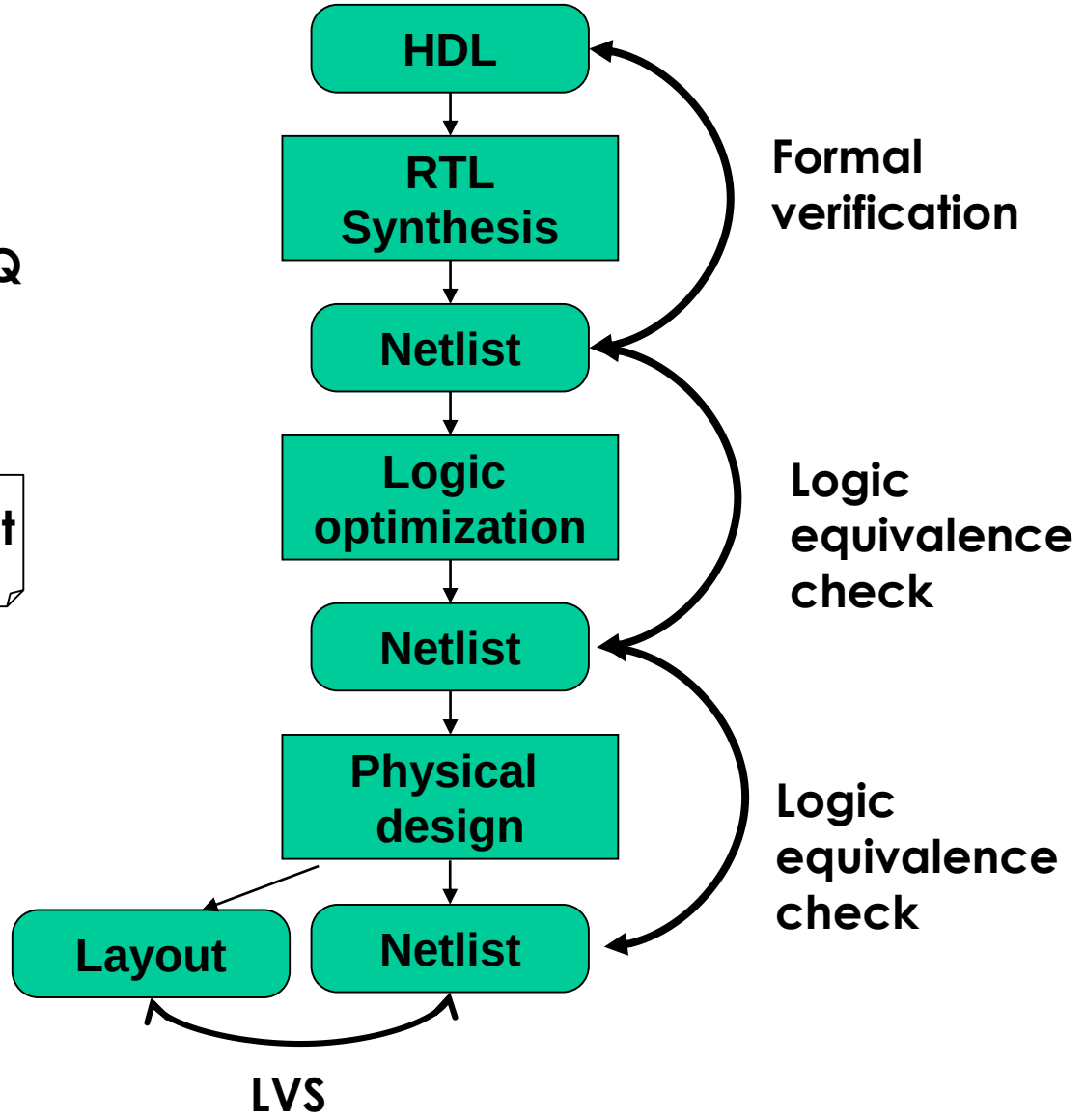
IR Drop and EM Analysis

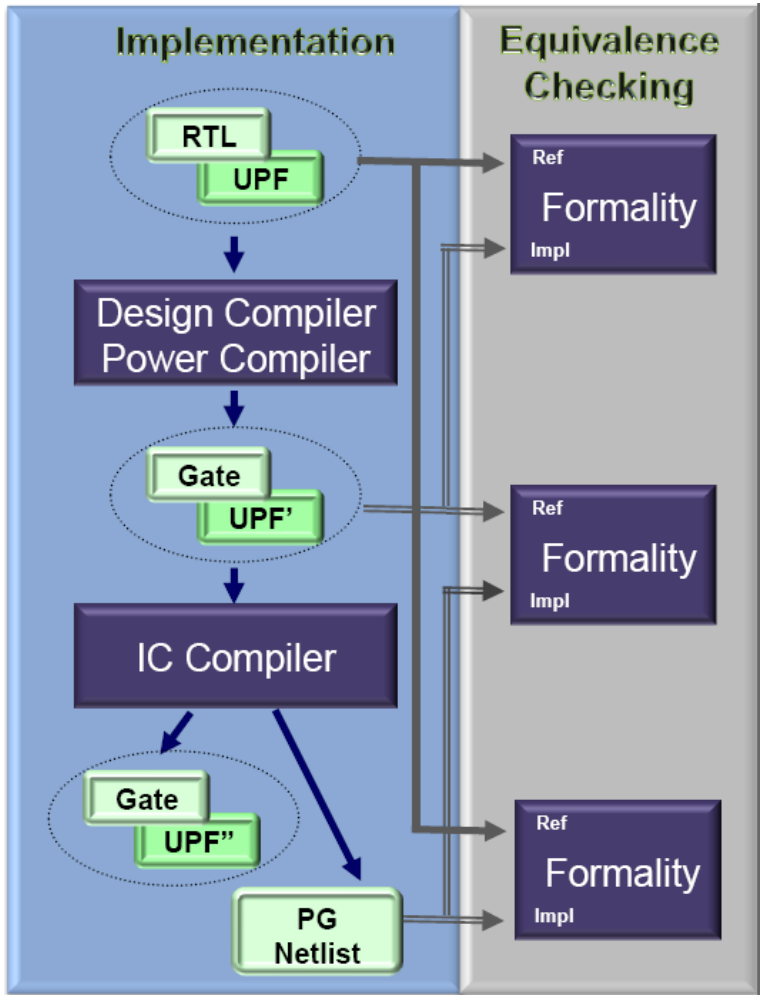




Logic Equivalence

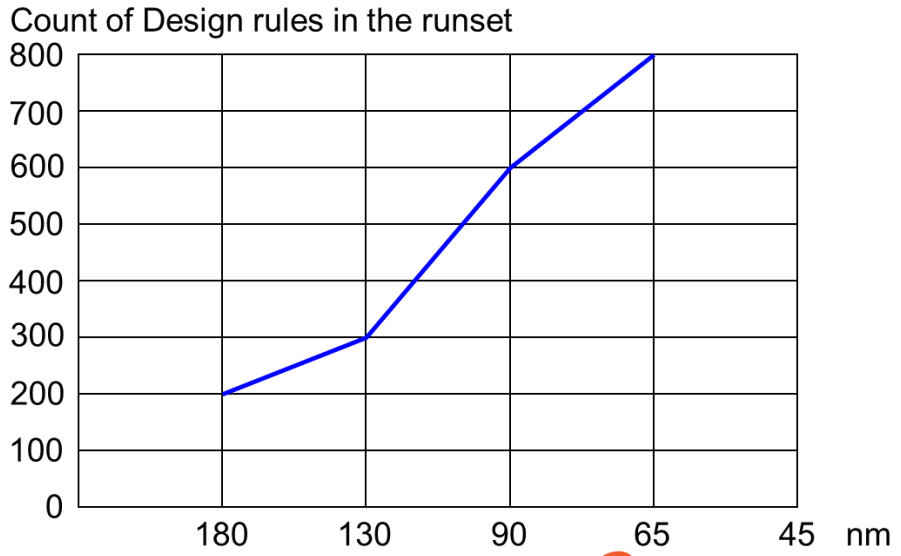






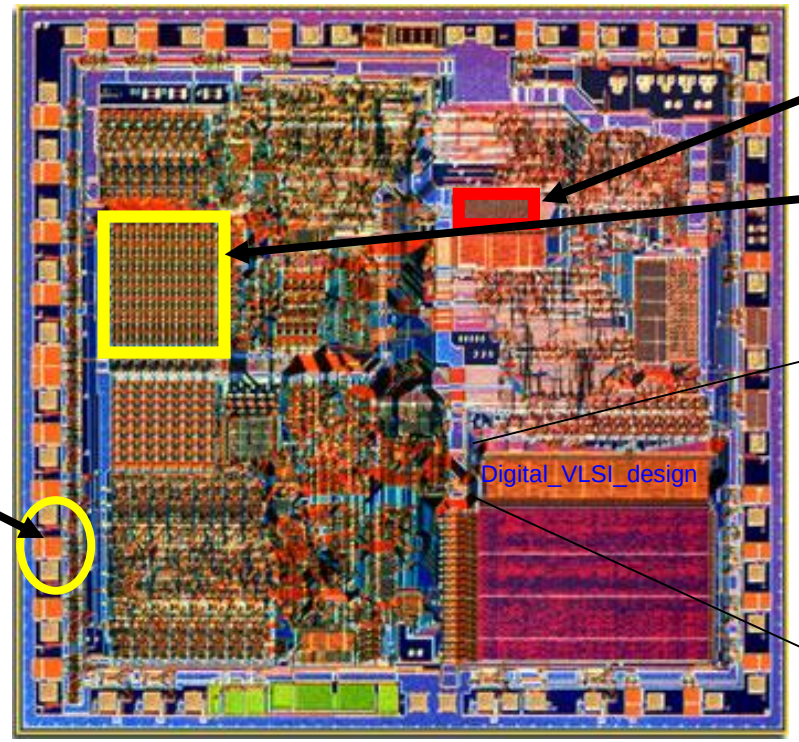
Layout (Physical) Verification

- **Design Rule Check (DRC)**
  - DRC run at the fullchip level on a sign-off DRC Tool.
  - Extra checks for fullchip are considered, including DFM recommended rules.
  - Applied to GDS streamed out from P&R tool with the addition of bonding pads, density fillers, toplevel markings, sealring, and labels.
- **Layout vs Schematic (LVS)**
  - Extract layout (GDS) and build Spice netlist
    - Sometimes need to black-box sensitive layouts.
  - Export verilog and translate into Spice netlist
  - Compare the two with a sign-off LVS tool.
- **Electrical Rule Check (ERC)**
  - Part of LVS. Checks for shorts, floating nets, well biasing.



Layout (Physical) Verification

- **Special GDS additions for tapeout:**
  - Special marker layers are used by DRC and LVS
  - Text labels are used for LVS and for commenting
  - Chip logo added in toplayer for identification
  - Reticle alignment or “fiducial” markings for alignment.



E.g : special DRC rules for RAM

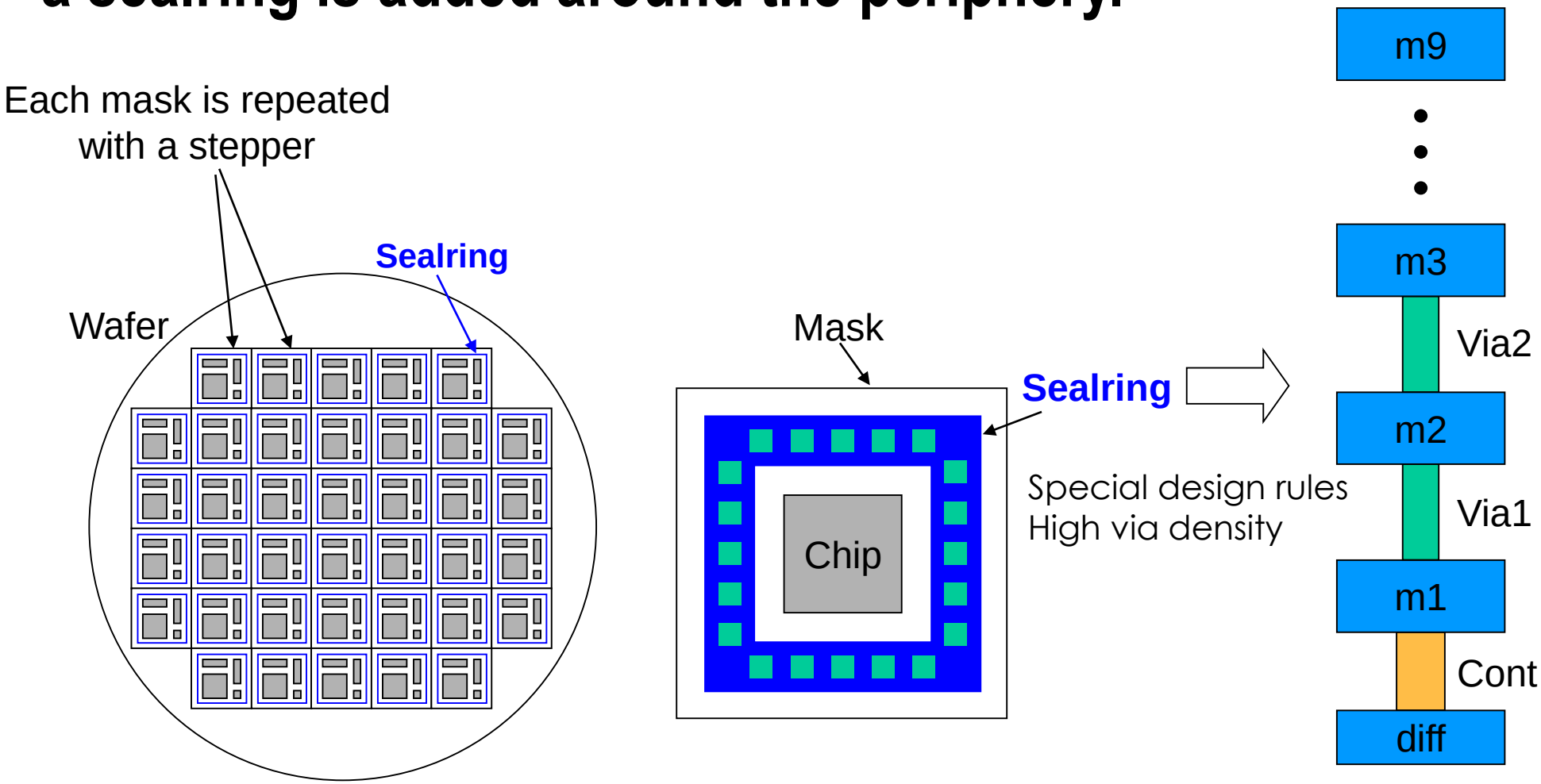
E.g : Layer to indicate no metal fill in this analog block

Reset

Logo

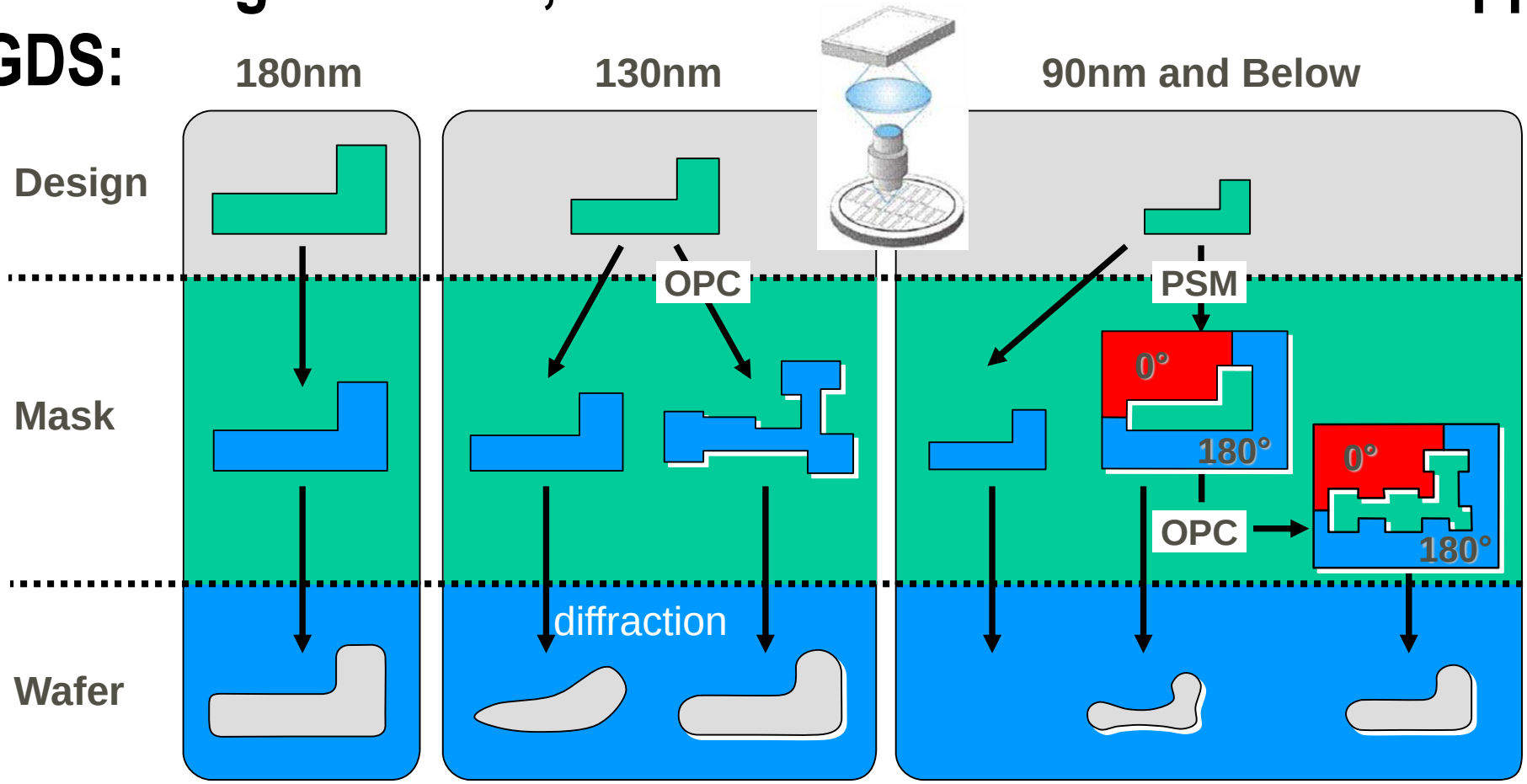
Adding a sealring

- **To protect the chip from damaging during dicing (sawing), a sealring is added around the periphery.**



Resolution Enhancement Techniques (RET)

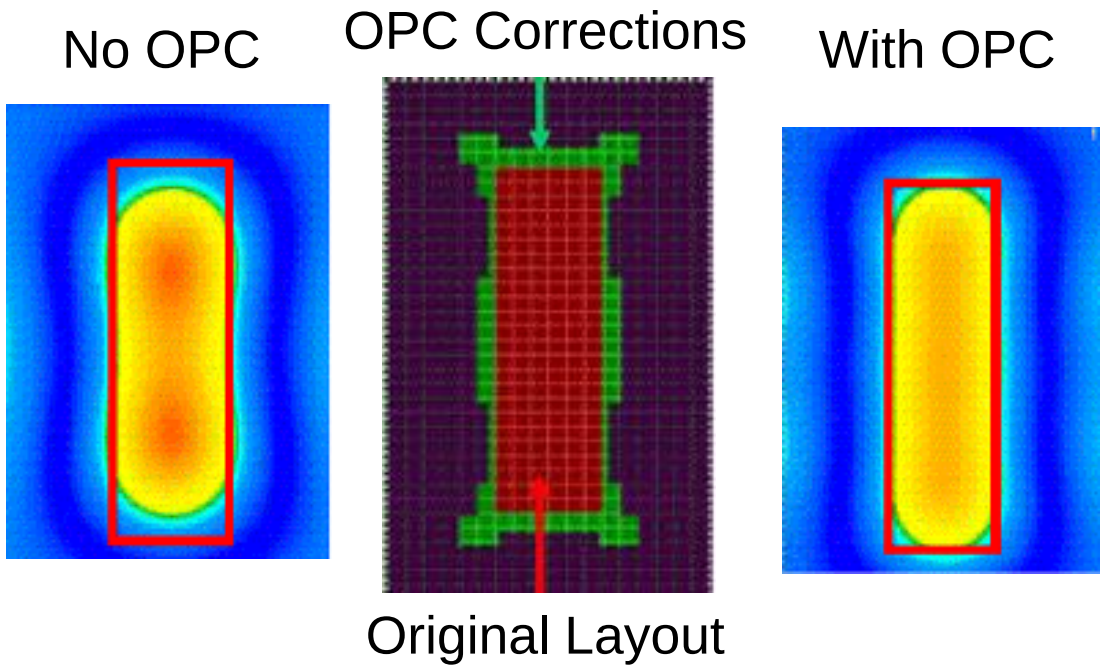
- **Before writing the mask, additional transformations are applied to the GDS:**

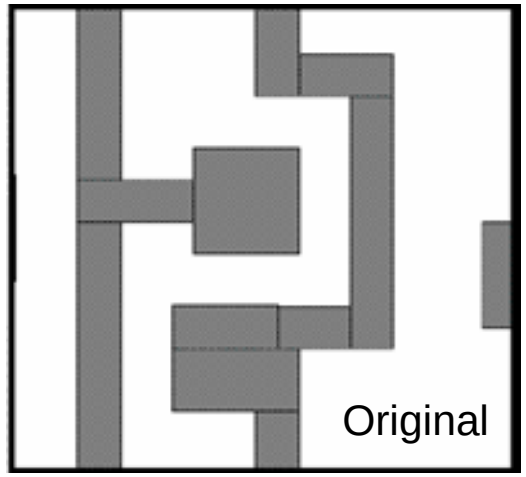


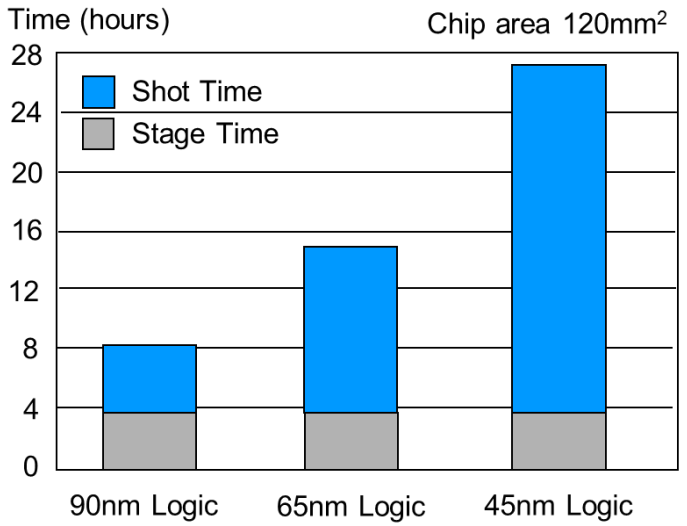
feature sizes
➔ light wave length

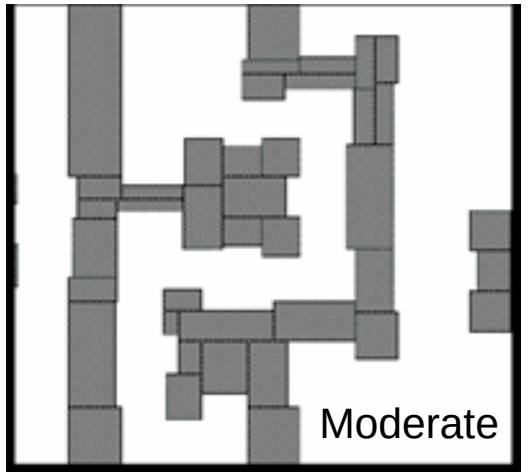
Resolution Enhancement Techniques (RET)

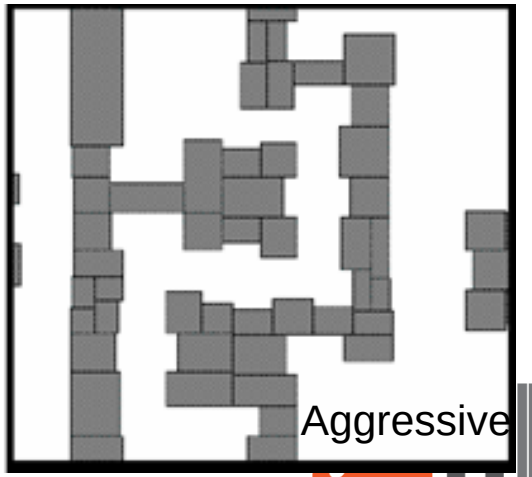
- **Optical Proximity Correction (OPC)**





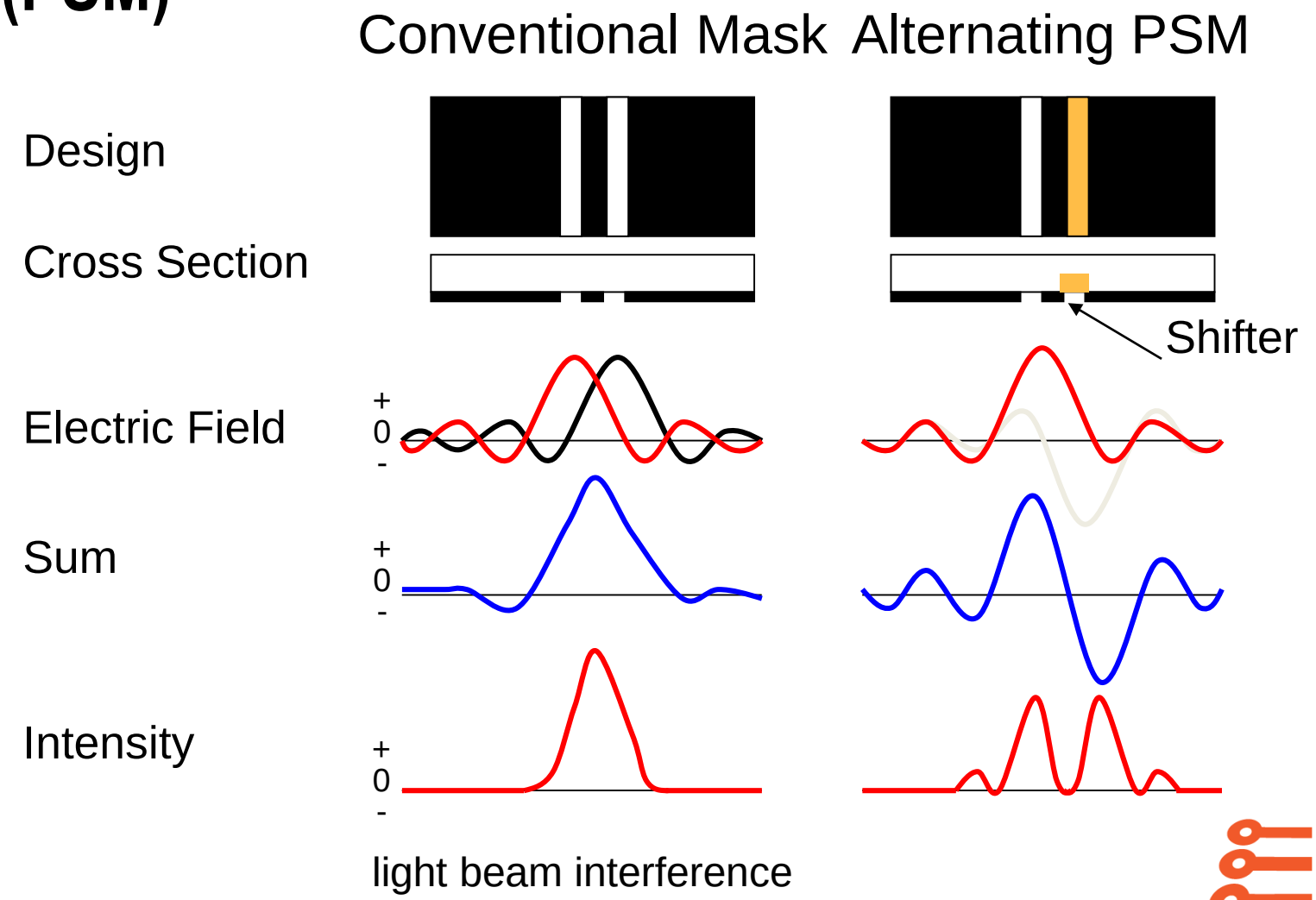






Resolution Enhancement Techniques (RET)

- **Phase Shift Masks (PSM)**



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