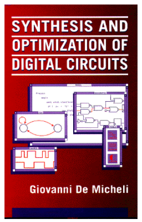
Elements of Physical Design

Giovanni De Micheli

Integrated Systems Laboratory

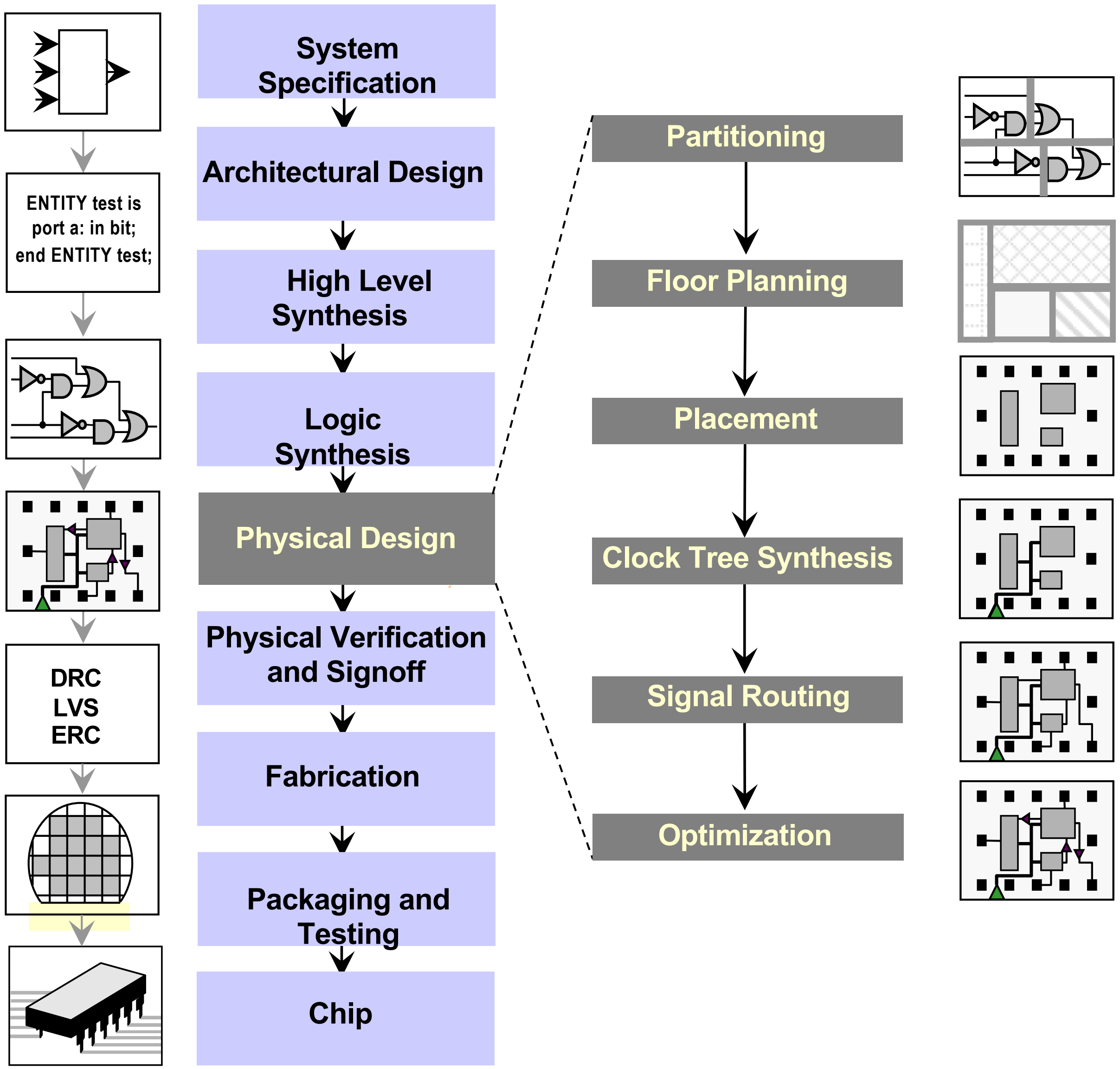
with credits to P. Gruenweld



Module 1

- **Objectives**
  - **Physical design**
  - **Basic and new rules in layout**

Big Picture: Physical Design in the Flow



Major Physical Design Algorithmic Problems

- **Partitioning:**
  - Divide the problem into sub-parts
    - such that the number of pins in between and other costs are minimized
- **Floor planning:**
  - Shape and place large non-overlapping modules
    - such that total area and wire length are minimized
- **Placement:**
  - Assign non-overlapping locations to gates
    - such that total wire length is minimized
- **Routing:**
  - Generate correct non-overlapping wires according to net list connectivity
    - such that total wire length and other 'costs' are minimized

Objectives & Constraints
What are we Optimizing for?

- Low Unit Cost implies smallest die area
  - No unused space
  - Small total gate area
  - Avoid local congested areas
- Maximum performance
  - Parallel execution (costs area)
  - Reduce logic depth (often costs area)
  - Reduce wire length (byproduct of small area)
  - Higher frequency (costs power)
- Power
  - Reduce wire length & area
  - Low leakage cells (costs performance and/or area)
  - Voltage regions (makes for a more complex floorplan)

Constraints vs Objectives

- **Constraints:**
  - Logically correct
  - DRC-correct (no shorts and opens)
  - LVS correct (layout versus schematic)
  - Frequency = speed

  } Must-have

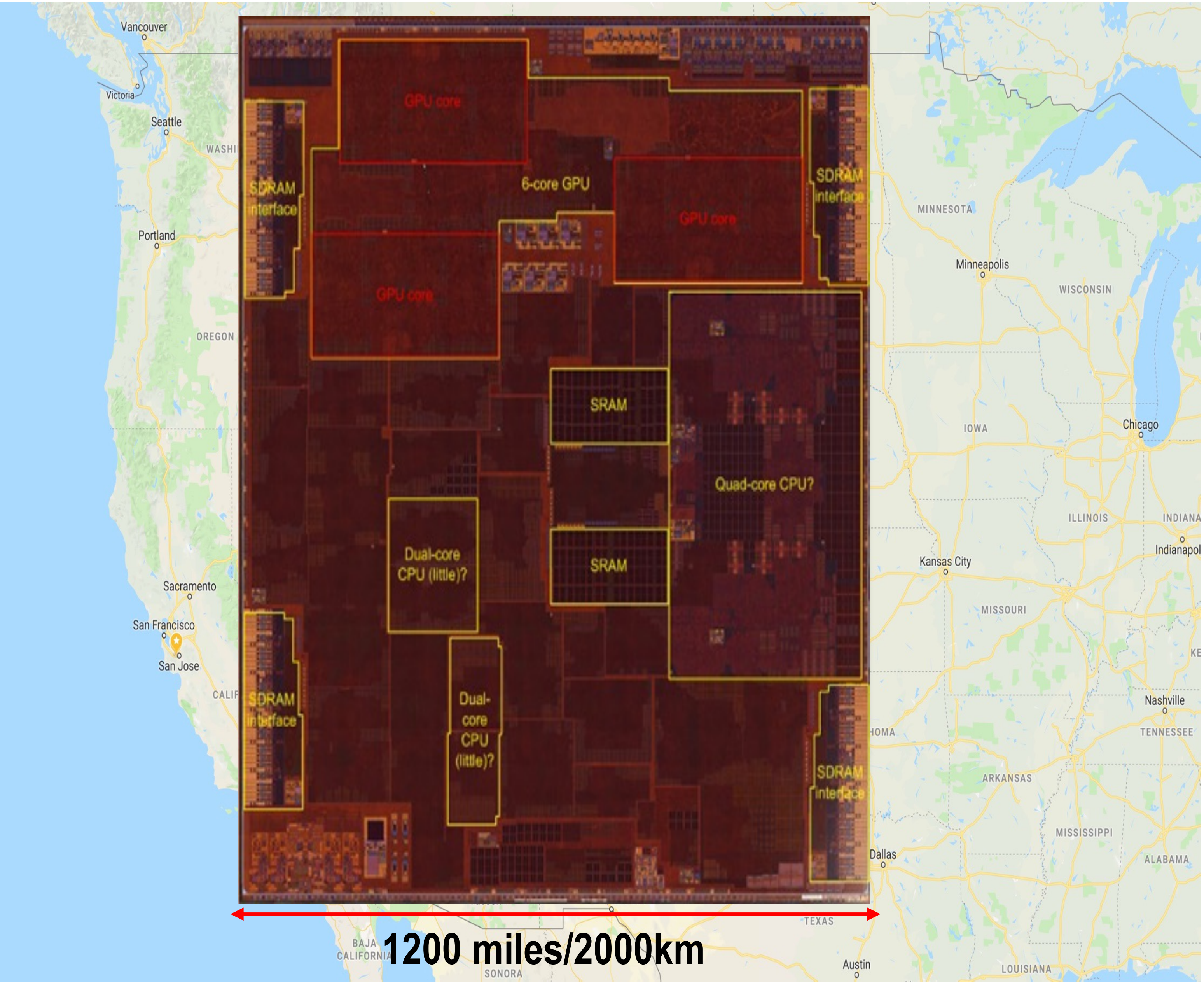
- **Objectives:**
  - Power
  - Area
  - Performance

  } Nice to have

Module 2

- Objectives
  - Algorithms for physical design
  - Routing algorithms
    - Area routers
    - Channel routers
  - Placement algorithms
    - Constructive
    - Iterative

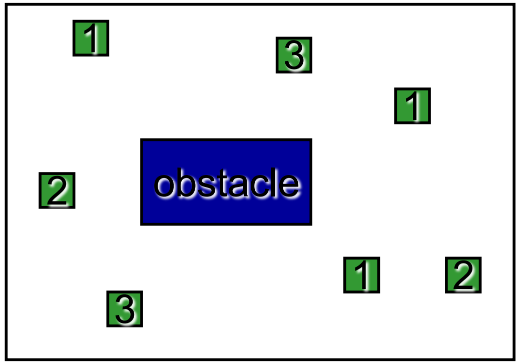
If the Wires on a 10nm Mobile SoC were as Wide as Roads…

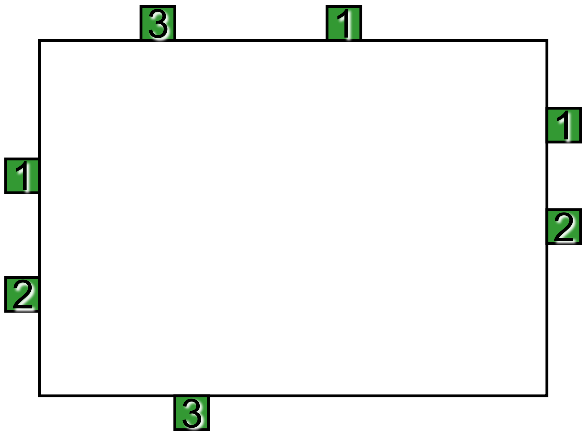


A chip contains ~10 million km wires in 10 layers. Connecting 4.4 Billion transistors in 0.2 Billion cells

The USA contains ~4.3 million km paved roads in 1 layer. Connecting 0.3 Billion people in 0.14 Billion homes

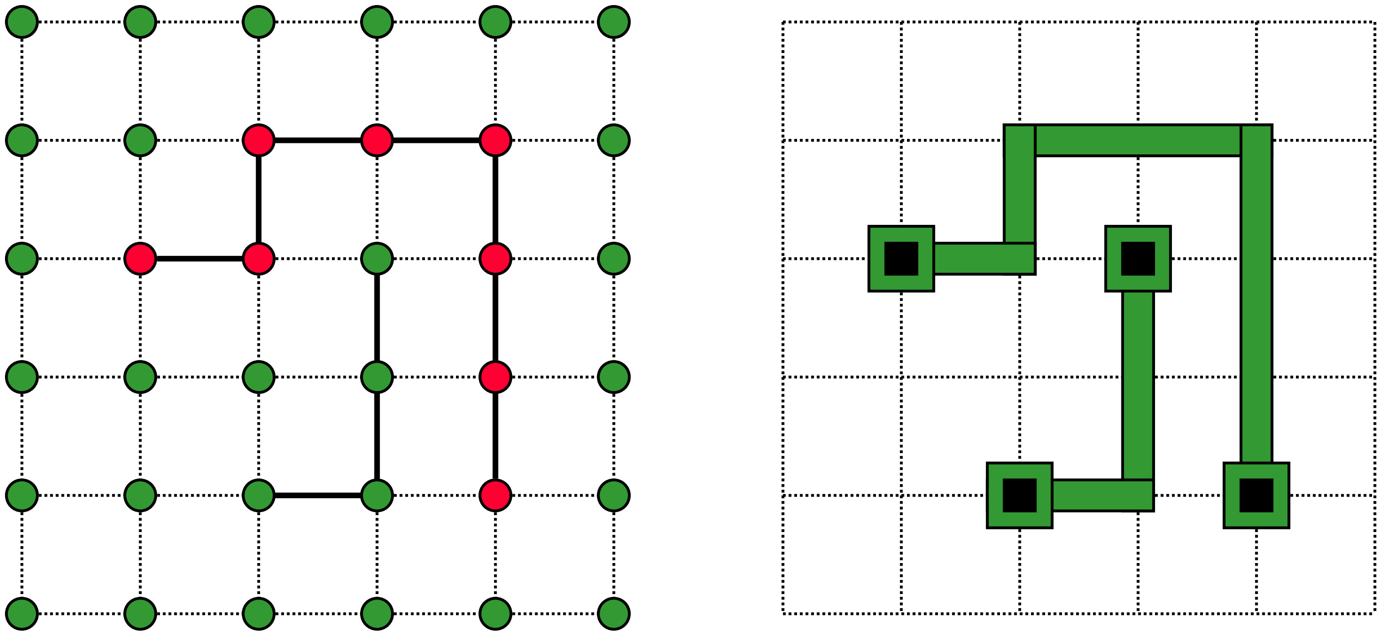
Maze routing





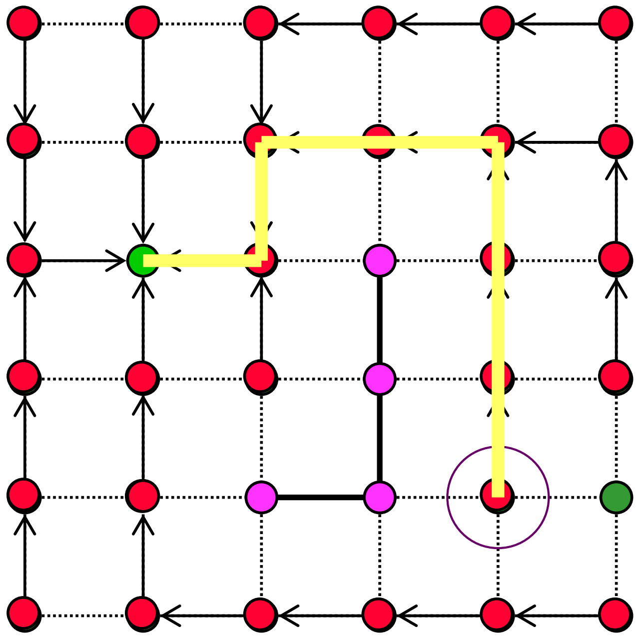
- Given:
  - An area model
  - Net list with pins
  - Design rules
- Constraints:
  - Connect all nets
  - Design rule correct
- Optimization criteria:
  - Minimize total wire length
  - Follow directives
  - Minimize vias

Grid graph as framework



The proper choice of the spacing ensures DRC

Dijkstra's algorithm



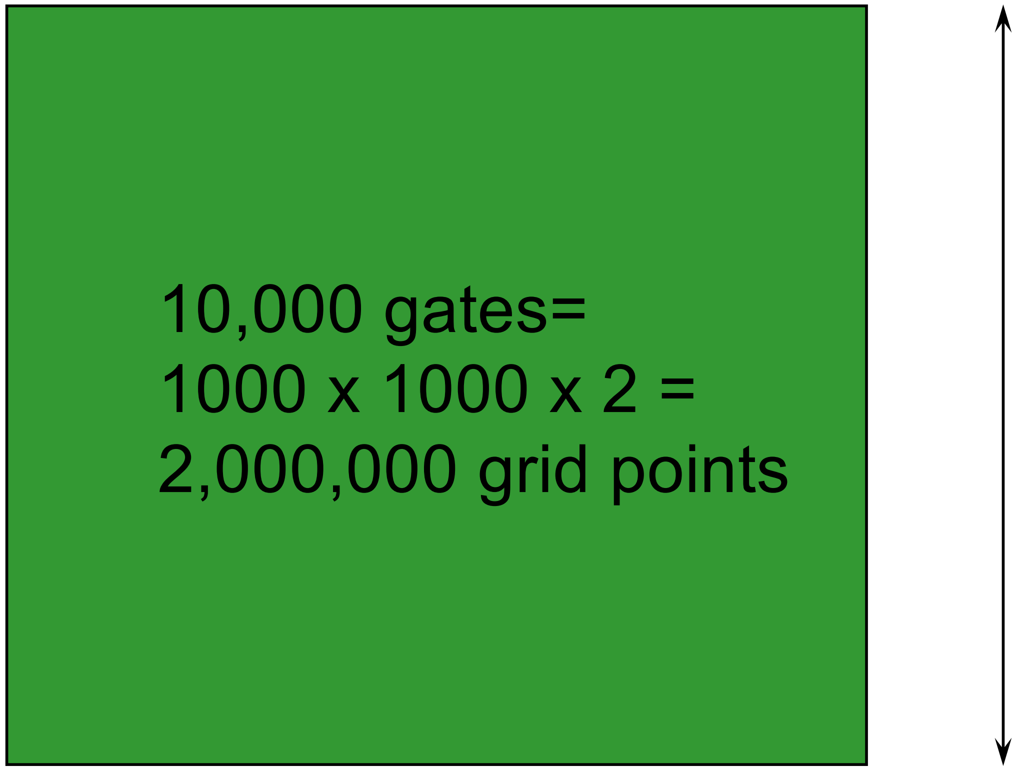
- Guarantees to find shortest path, if it exists.
- Quadratic behavior: Slow, especially in 'sparse' designs.
- Modeling on grid has limitations.

Lee-style maze router

In the early 60s, Lee published a router based on *Dijkstra's shortest path* algorithm. The algorithm is run sequentially for each net.

- **No guarantee for routing solution, even it one exists**
  - **Each net is routed independently of all others**
- **Decent operation requires hacking and tuning**
- **Too expensive for large circuits**
- **Suggested improvements:**
  - **Speed-up: partitioning**
  - **Rip-up-and-reroute**
  - **Spreading congestion by global routing**

Taming quadratic behaviour



100 gates=
1000 grid

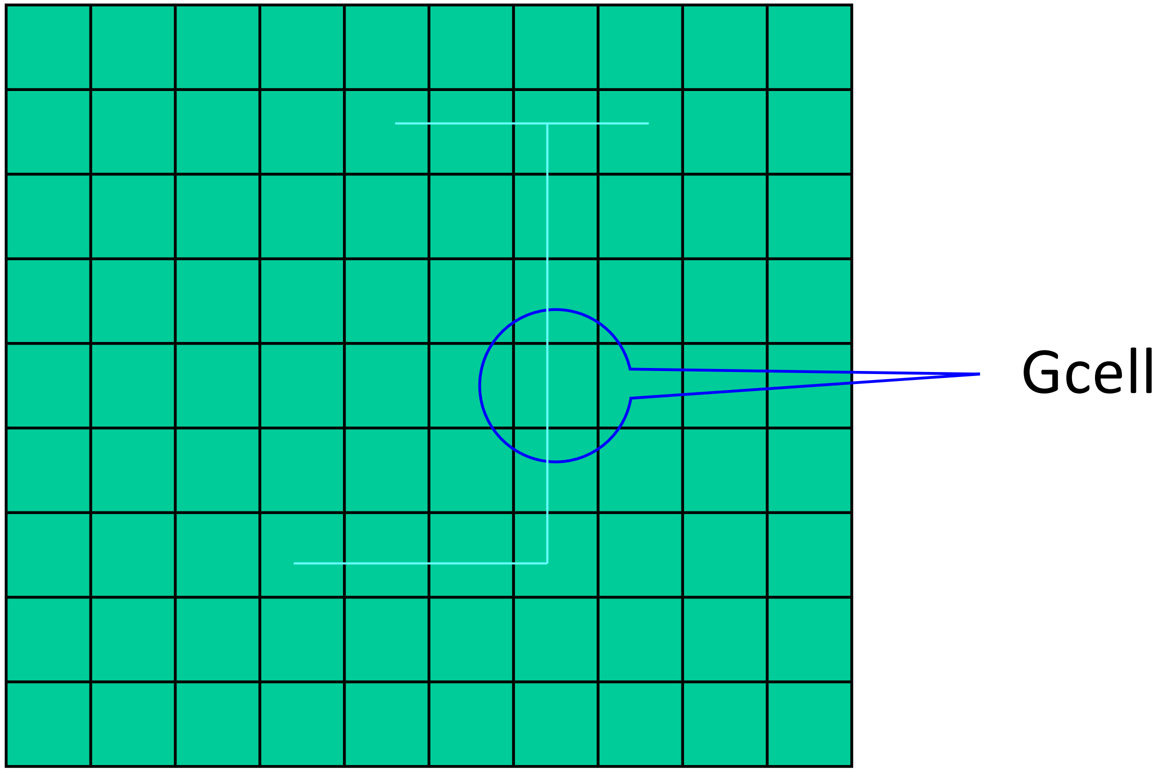
Solution: "Global routing"

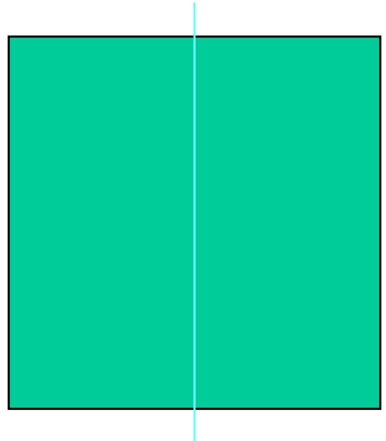
Bring hierarchy in the routing problem:

1) Global route on coarse grid
2) Detailed route on fine grid

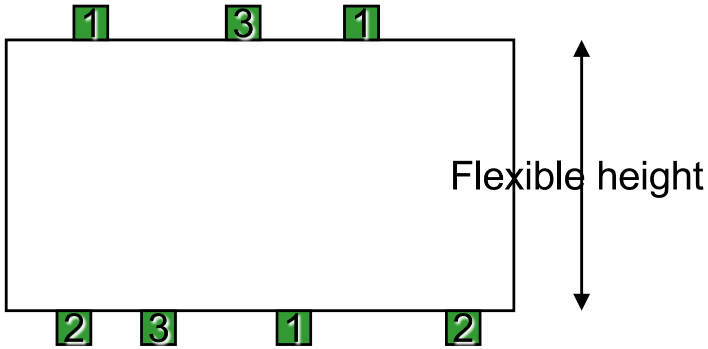
Task of a global router:

Find coarse path and layer assignment for each net, such that:
wire density is spread evenly





Channel routing



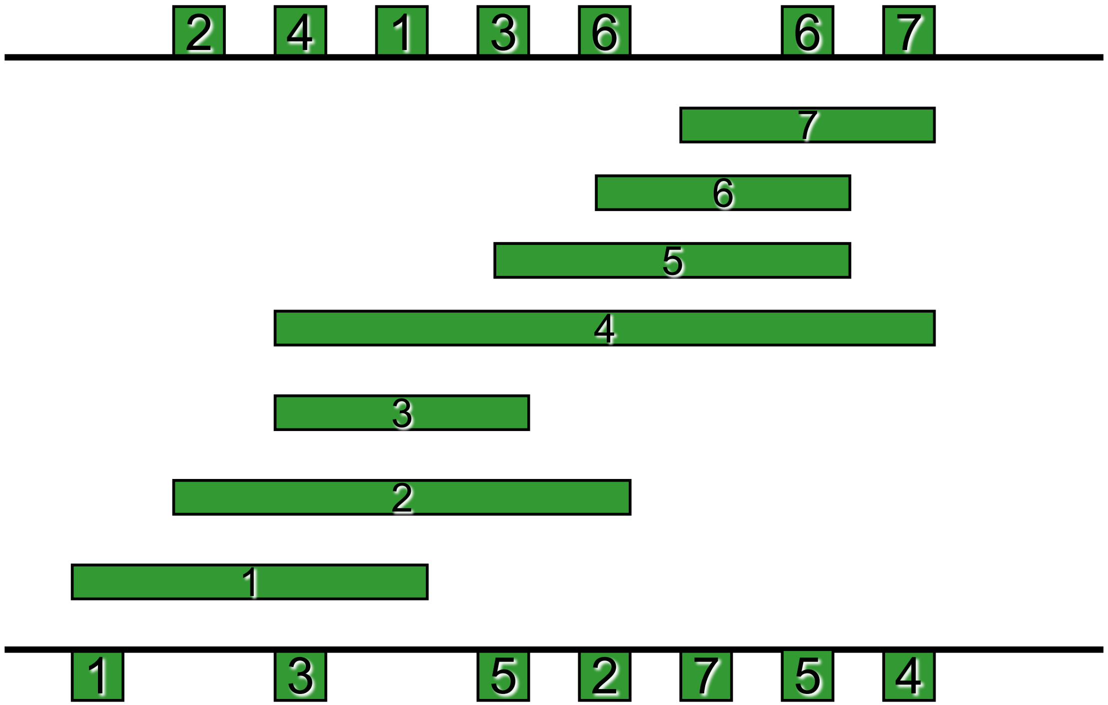
Guaranteed

To fulfill constraints!

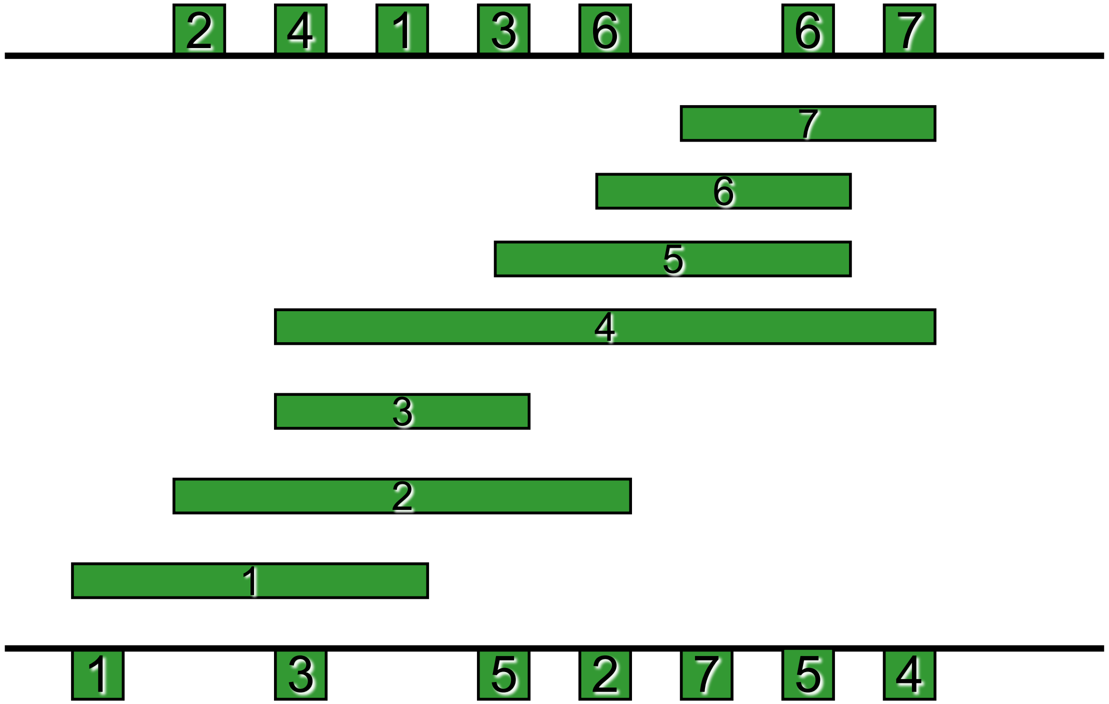
- Given:
  - An area model (flexible)
  - Net list with pins
  - Design rules
- Constraints:
  - Connect all nets
  - Design rule correct
- Optimization criteria:
  - Minimize channel height
  - Minimize vias
  - Minimize wire length

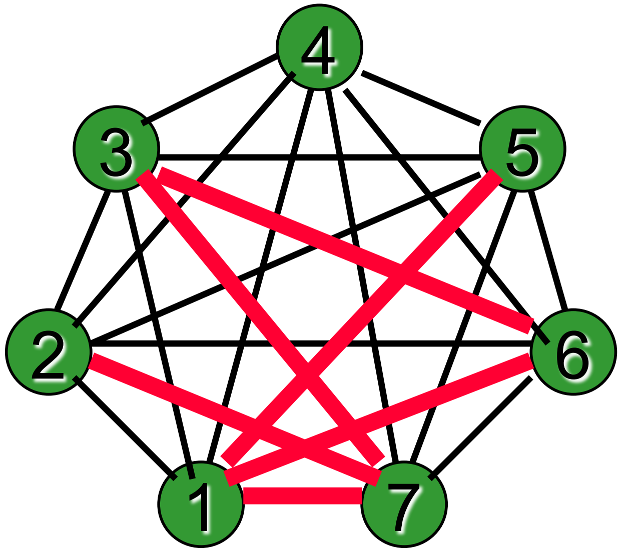
Reduce search space:

- Implement each net by a single horizontal wire (trunk)
- Connect pins to wire by vertical branches
- ‘Tetris’-style compaction: share the rows
- The trunk spans the entire net:



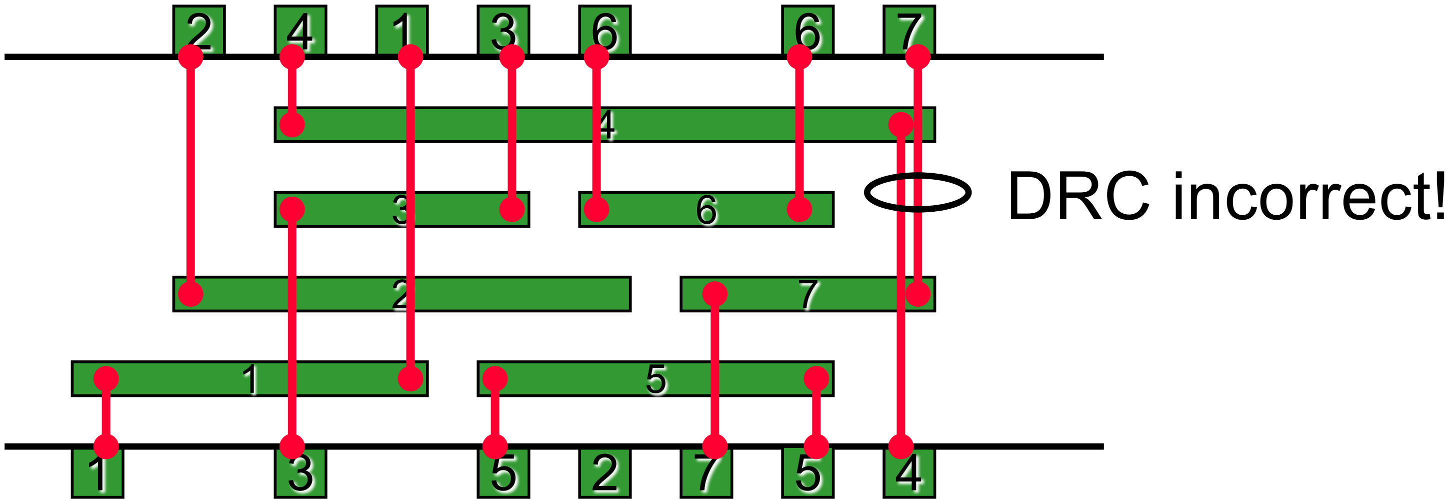
Horizontal Constraint Graph





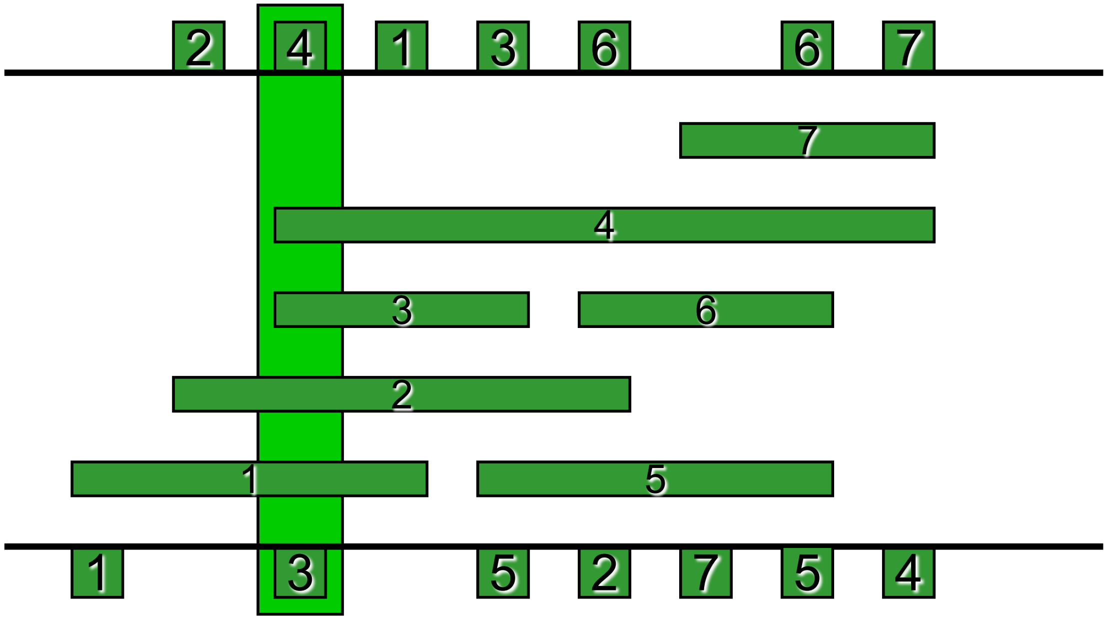
The complement of the horizontal constraint graph contains segments which can be teamed into one track

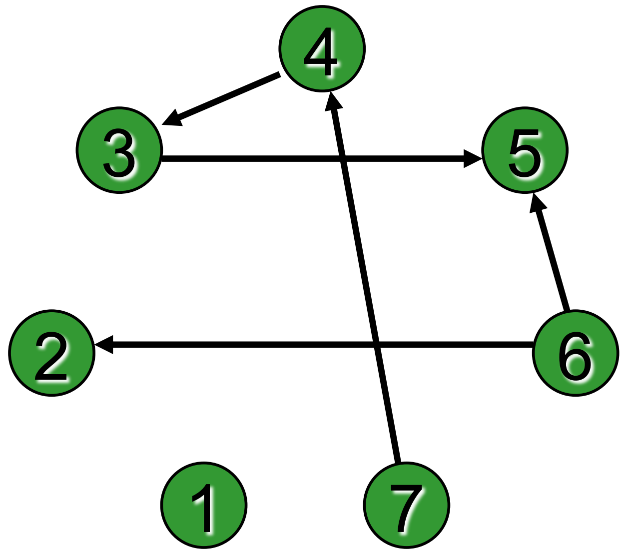
‘Left edge’ algorithm



Channel Density = 4

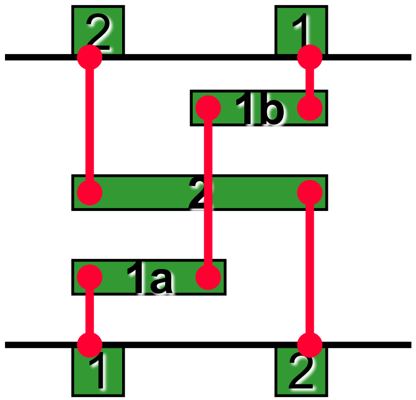
Vertical Constraints

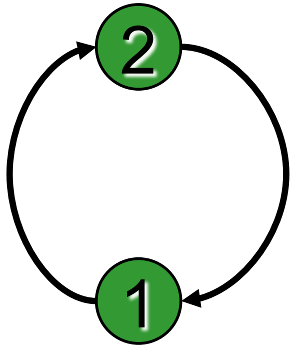


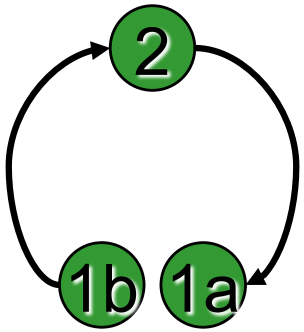


Channel needs to satisfy both horizontal and vertical constraints!

Cycles in the VC Graph

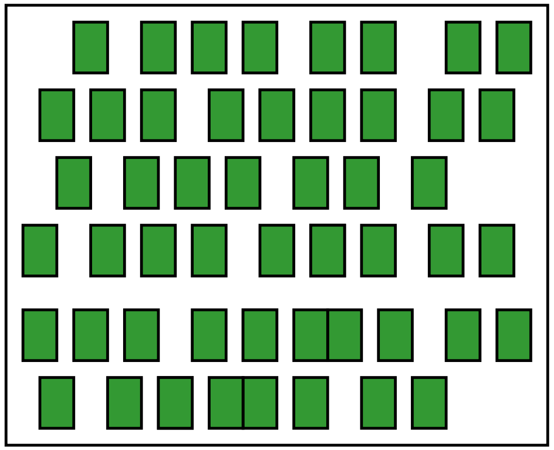






Insert a ‘dogleg’

Placement



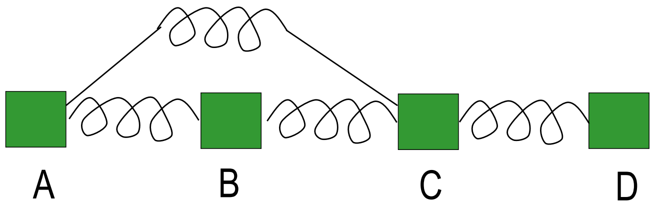
- Given:
  - An area model
  - Net list with cells
  - Cell geometries
- Constraints:
  - Cells may not overlap
- Optimization criteria:
  - Minimize area
  - Minimize total wire length

Placement algorithms

- Constructive methods
  - Quadratic placement
- Basic model
  - Abstract modules as points
  - Abstract connectivity as attractive force
  - Use pinout constraints to find a balanced solution
- Mechanical analogy
  - Find minimum energy configuration

Example

- 1-dimensional placement of 4 modules



$$B = \begin{bmatrix} k_1+k_4 & -k_1 & -k_4 & 0 \\ -k_1 & k_1+k_2 & -k_2 & 0 \\ -k_4 & -k_2 & k_2+k_3+k_4 & -k_3 \\ 0 & 0 & -k_3 & k_3 \end{bmatrix}$$

- C = interconnection matrix
- D = diagonal matrix $d_{ii} = \sum c_{ij}$
- B = D-C
- At equilibrium $Bx = 0$
- B is singular
  - Need to fix endpoints

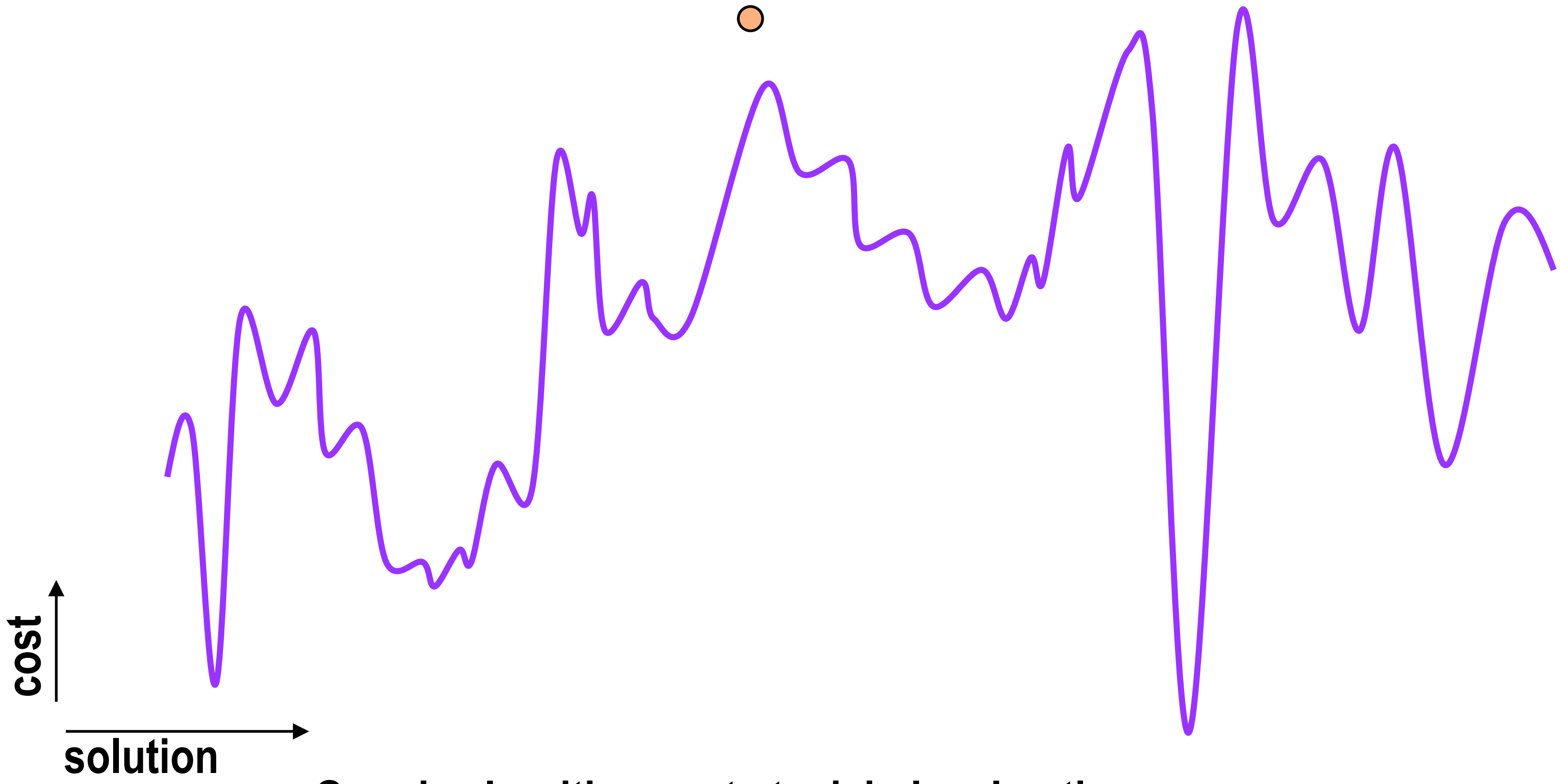
Eigenvalue methods

- Minimize energy
  - Quadratic form $x^T B x$
  - B symmetric matrix
  - Hence $\lambda_{min} \leq x^T B x / x^T x \leq \lambda_{max}$
- Quadratic form is minimum when x is the eigenvector corresponding to λ_{min}
- For two dimensional placement
  - Compute quadratic form in x and y
  - Consider eigenvectors related to two smallest eigenvalue
    - To avoid to place elements on a diagonal

Iterative methods

- Start from initial placement
- While cost function decreases
  - Swap two elements or displace one element
- Cost function is overall wiring length
- Often solution is a local minimum

The Cost Landscape



Greedy algorithms get stuck in local optimum

Simulated annealing

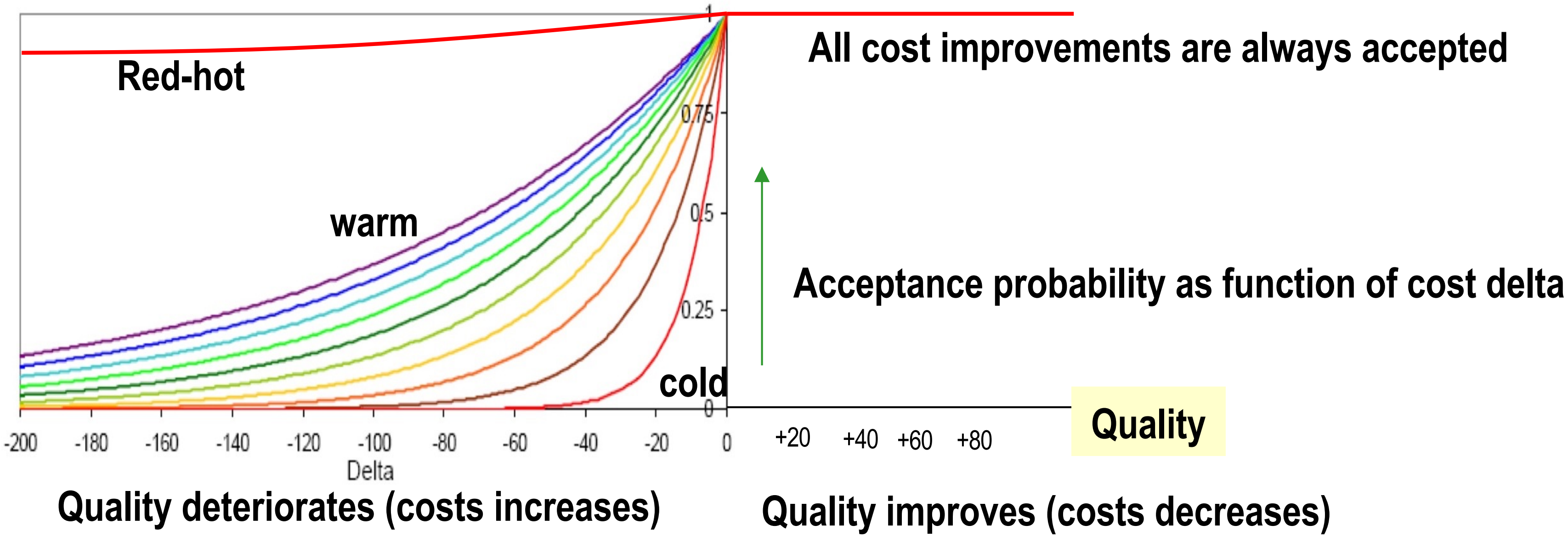
- Iterative methods with probabilistic escape from local minima
  - Allow uphill moves with a certain probability
  - Always allow downhill moves
- Drive probability of uphill moves slowly to zero
- Physical analogy
  - Annealing in metals
    - Warm up over melting point
    - Cool down slowly to allow crystal to attain minimum energy configuration
- Key factor is cooling schedule

Metropolis algorithm

- Simulation of gas at given temperature T
- Generate random displacement/interchange of particles
- Compute difference in energy
  - If difference is negative -- accept
  - If difference is positive -- accept with probability $\min(1, e^{-\Delta E/kT})$
- After a large set of moves, the simulated system is in equilibrium at T (Boltzman distribution)
- Simulated annealing is like running Metropolis algorithm with a temperature schedule
  - Key factor: cool down slowly

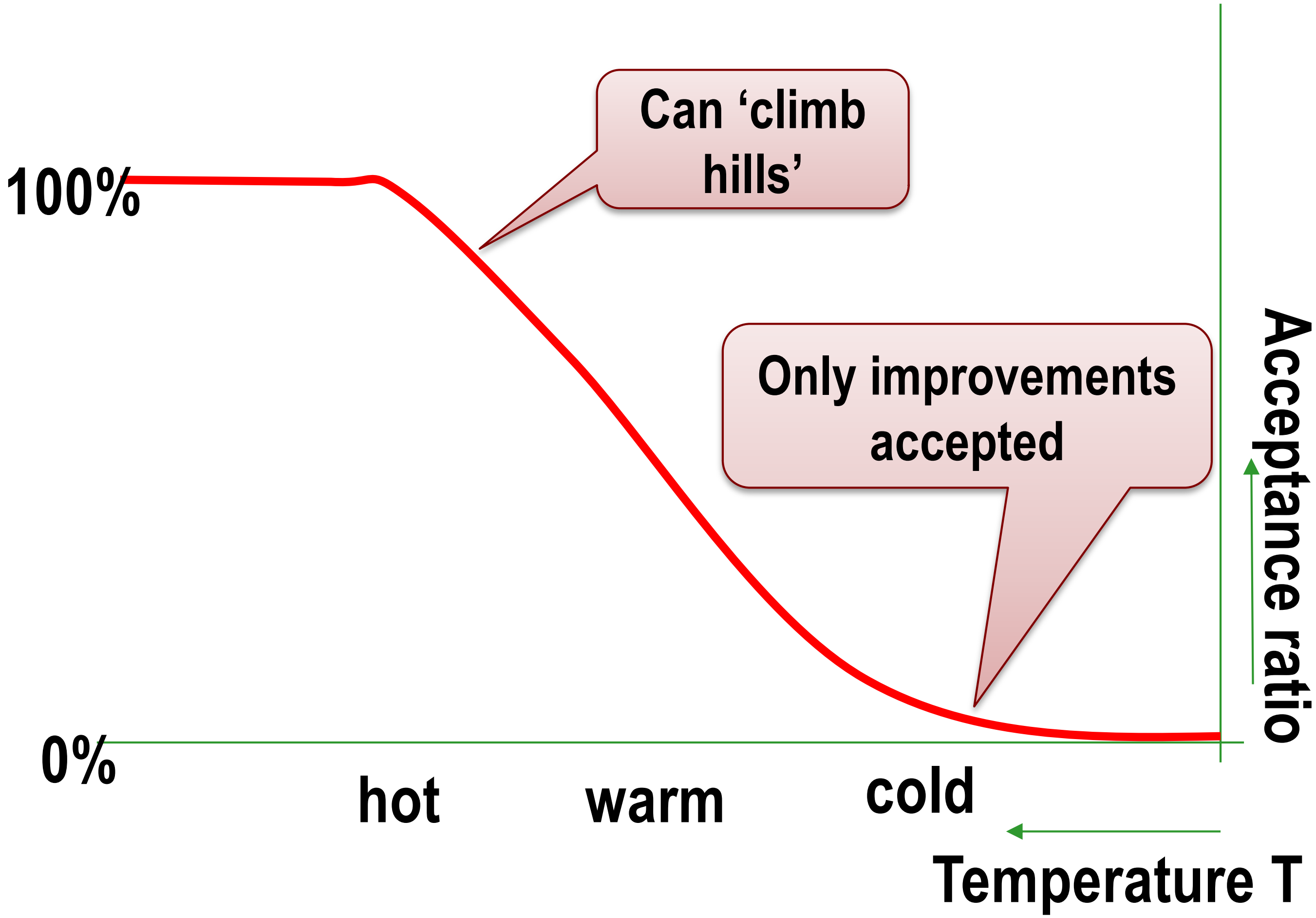
Simulated Annealing

- Base idea: slowly cool from hot to cold
- The energy state E of the system corresponds to the cost of a configuration.
- Energy (cost) increases are accepted with probability
- Energy decreases are always accepted. Very low temperature is equivalent to greedy improvement.



Kirkpatrick, S.; Gelatt Jr, C. D.; Vecchi, M. P. (1983). "Optimization by Simulated Annealing". *Science*. 220 (4598): 671–680

Simulated annealing: acceptance ratio



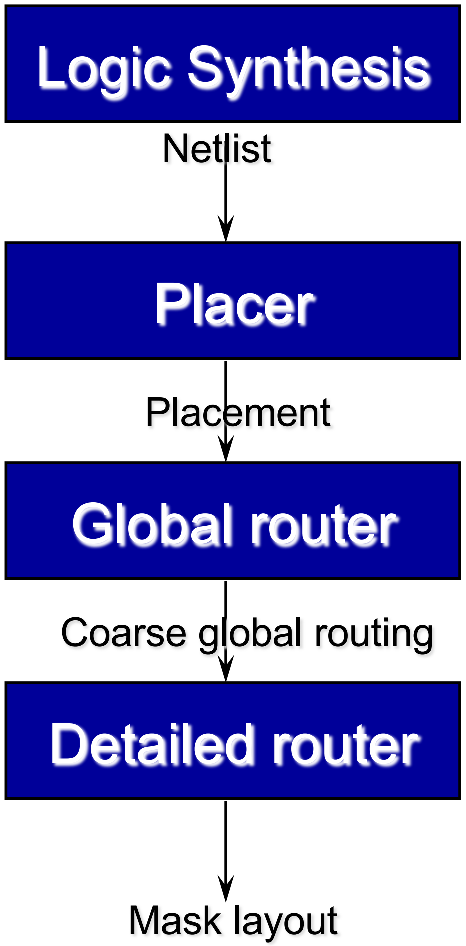
Simulated annealing

- The simulated annealing algorithm attains the global minimum if:
  - The process reaches equilibrium at each temperature
    OR
  - The cooling schedule is $T_k = c / \ln (k + \alpha)$ with $\alpha > 1$ and c the max depth of local minima
- Theoretical value only:
  - An infinite number of moves are required
  - Very good heuristic algorithm
  - Highly tunable

Module 3

- Objectives
  - Physical design flows
  - Fixed timing approach
  - Theory of logical effort

Vanilla flow



Places standard cells such that they do not overlap

Finds the approximate path of all nets which cross regions.

Routes the regions one by one

No completion guarantee in current technologies !!

Spoiling the fun: parasitics

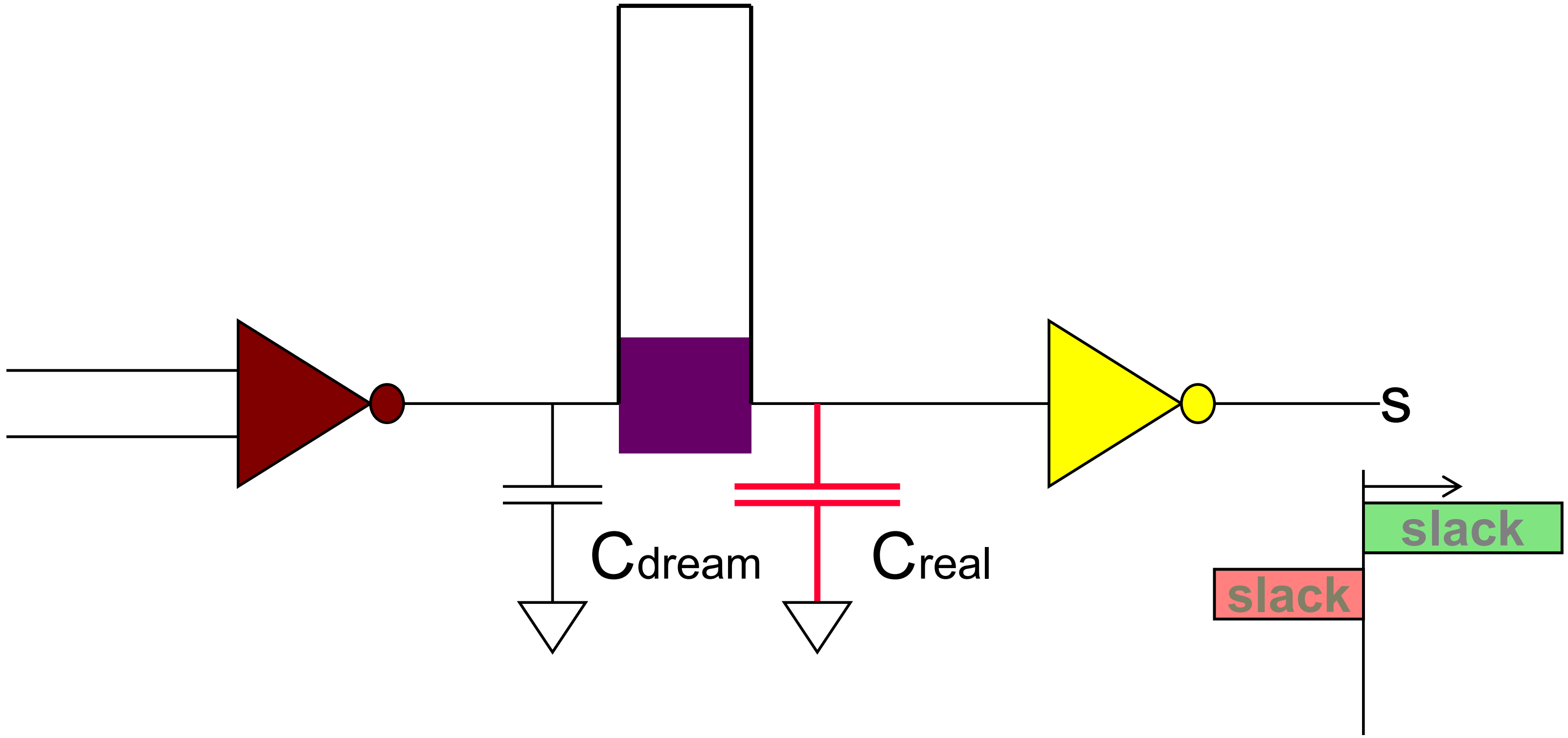
Goal: make circuit as fast as possible

- **Speed is determined *entirely* by parasitic capacitances and resistances**
- **Parasitics are tiny and depend on the exact layout**
- **Parasitics are extremely hard to estimate beforehand**

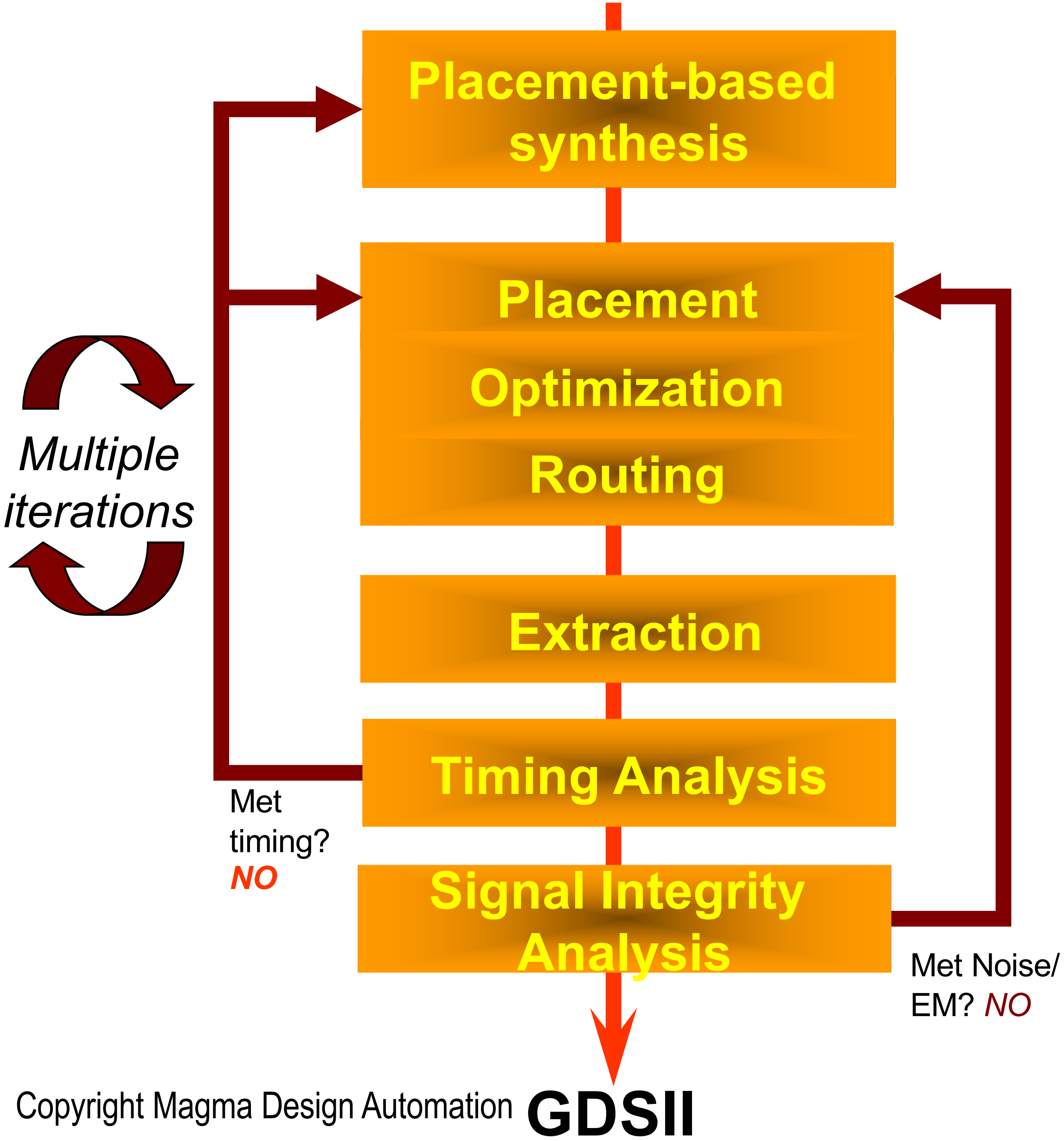
No more correct by construction

Timing is a result of the placement

- The bad news: the worst timing sets the clock speed!



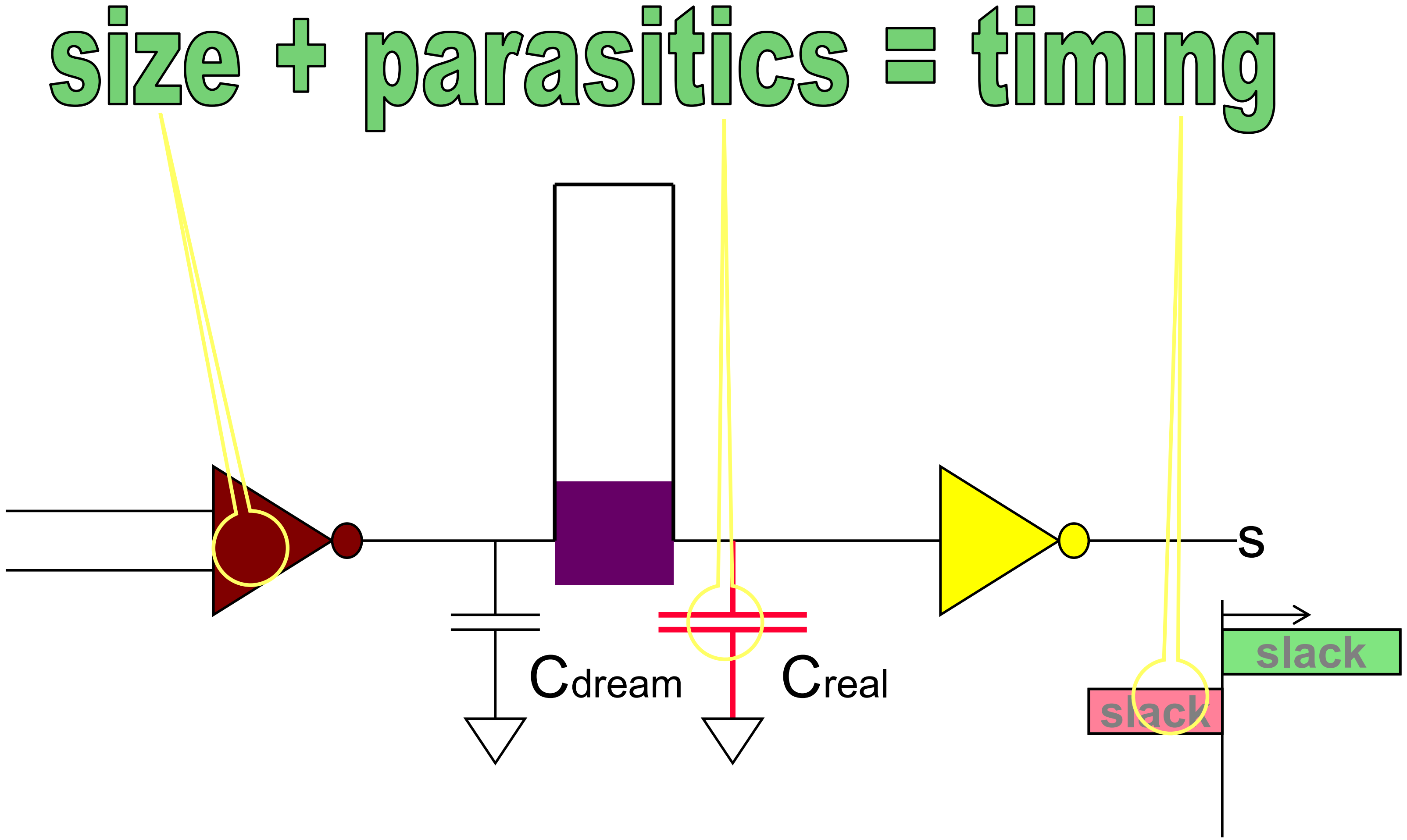
Design Flows



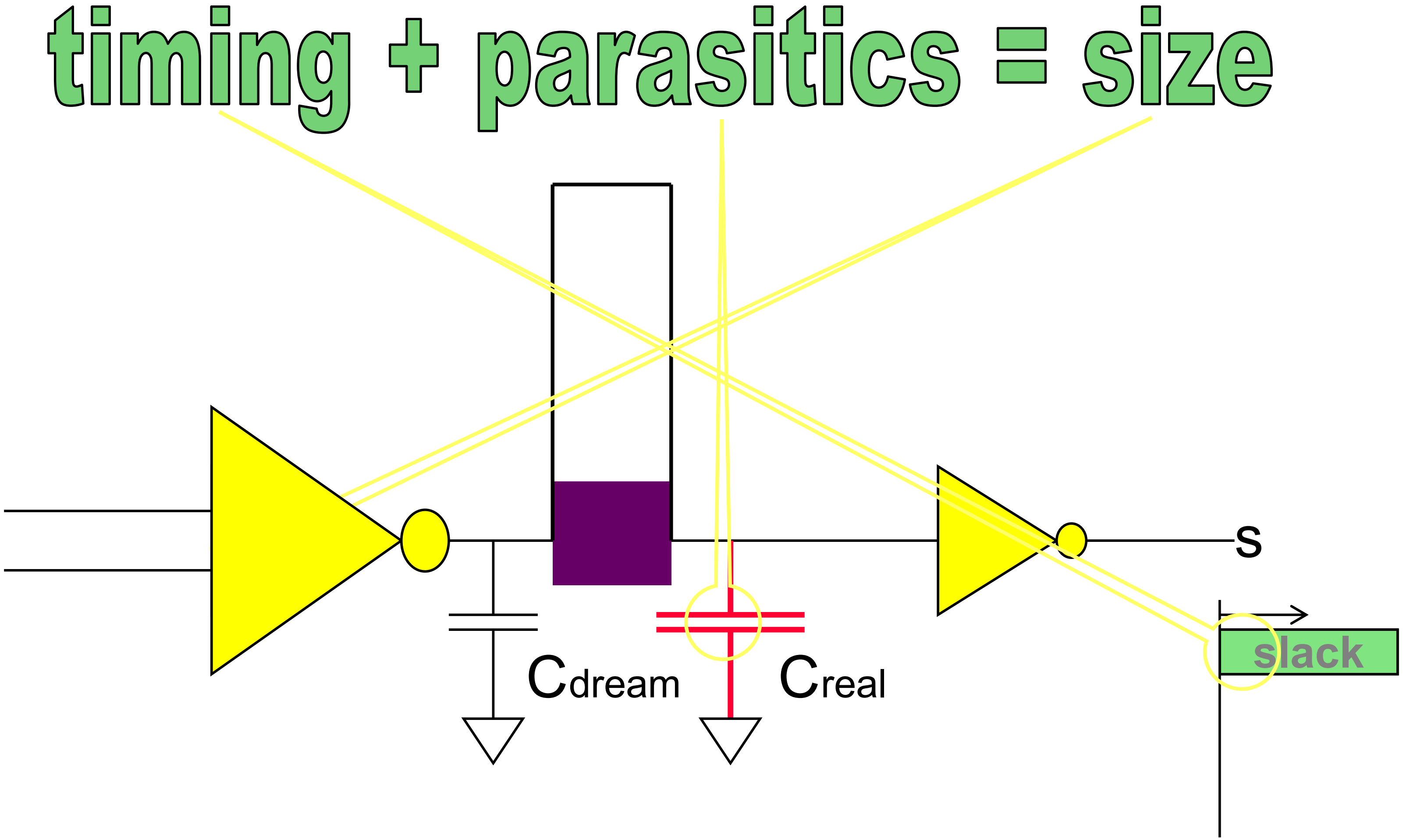
- Synthesis does not accurately model interconnect
- Cell sizes fixed before placement.
- Place & route unable to meet timing goal
- Signal integrity effects handled too late

Iterate to make ends meet!

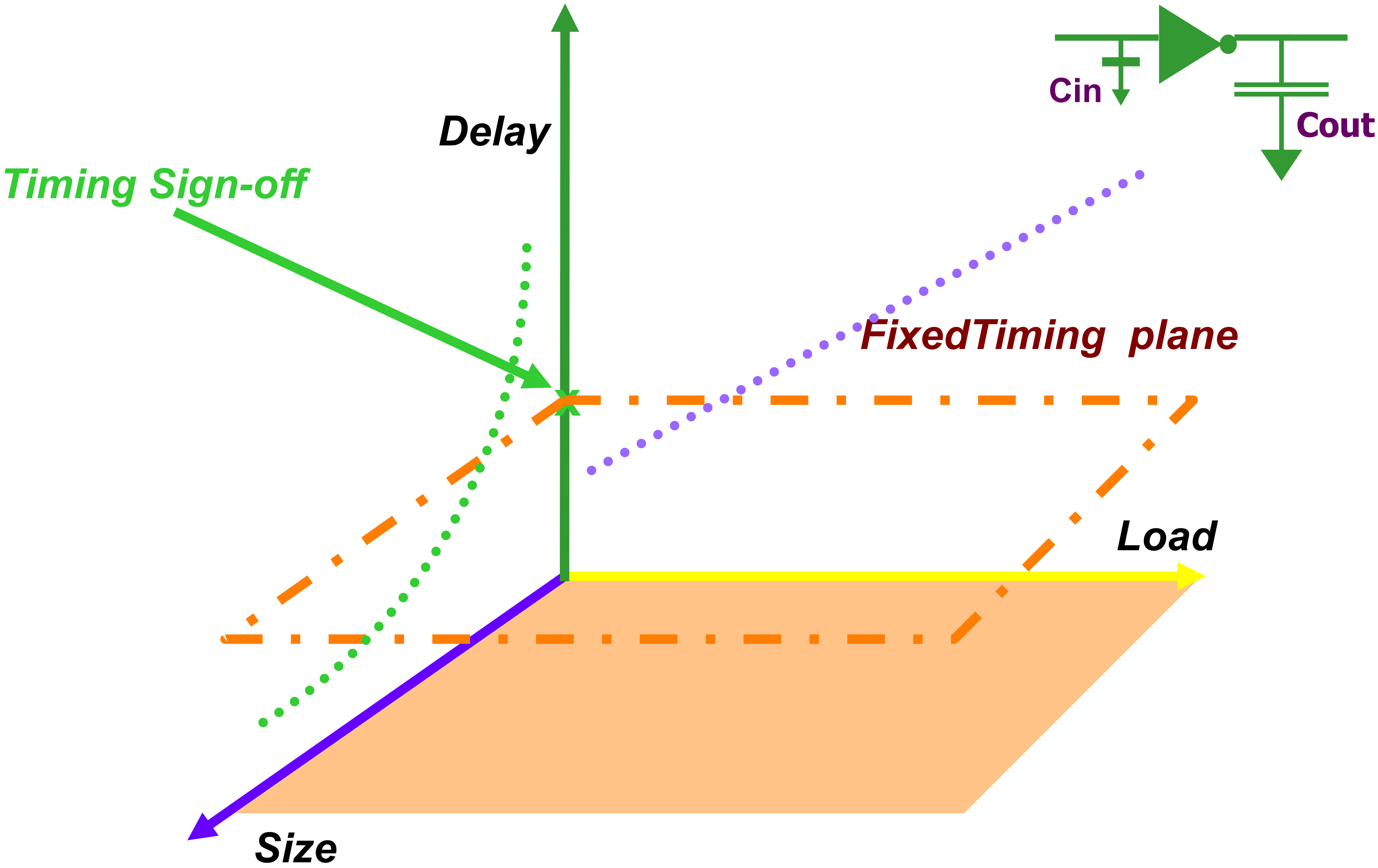
Conventional layout synthesis



Idea: keep timing fixed



Delay, Load and Size



The concept in a nutshell

- Goal:
  - Correct by construction (eliminate iterations)
- Pick the delays up-front
- Keep that delay throughout placement and routing
- Keep delay constant by cell sizing and other techniques

Comparison

Fixed area versus fixed timing

- Cell Area fixed
- Delay is a gamble
- Worst case delay determines timing (max)
- Iterate to make ends meet.
- After timing finally closes, many gates will be too big:
  - waste of area
  - waste of power

- Delay fixed
- Cell Area unknown
- Sum of areas determines chip size. (Additive)
- No iterations required
- Each gate has exactly the right drive strength
  - Not too little (fanout violation, timing fails)
  - Not too much (waste of area)

Summary

- In physical design it is hard to define 'optimal'
  - Modeling at various levels of abstraction is very inaccurate
  - Modeling of intricate wiring constraints is hard
- Most problems are NP-hard
- Rely on heuristics and 'black magic'
  - Difficult to control algorithms accurately
- Physical design spans many levels of abstraction:
  - From logic down to deepest mask level