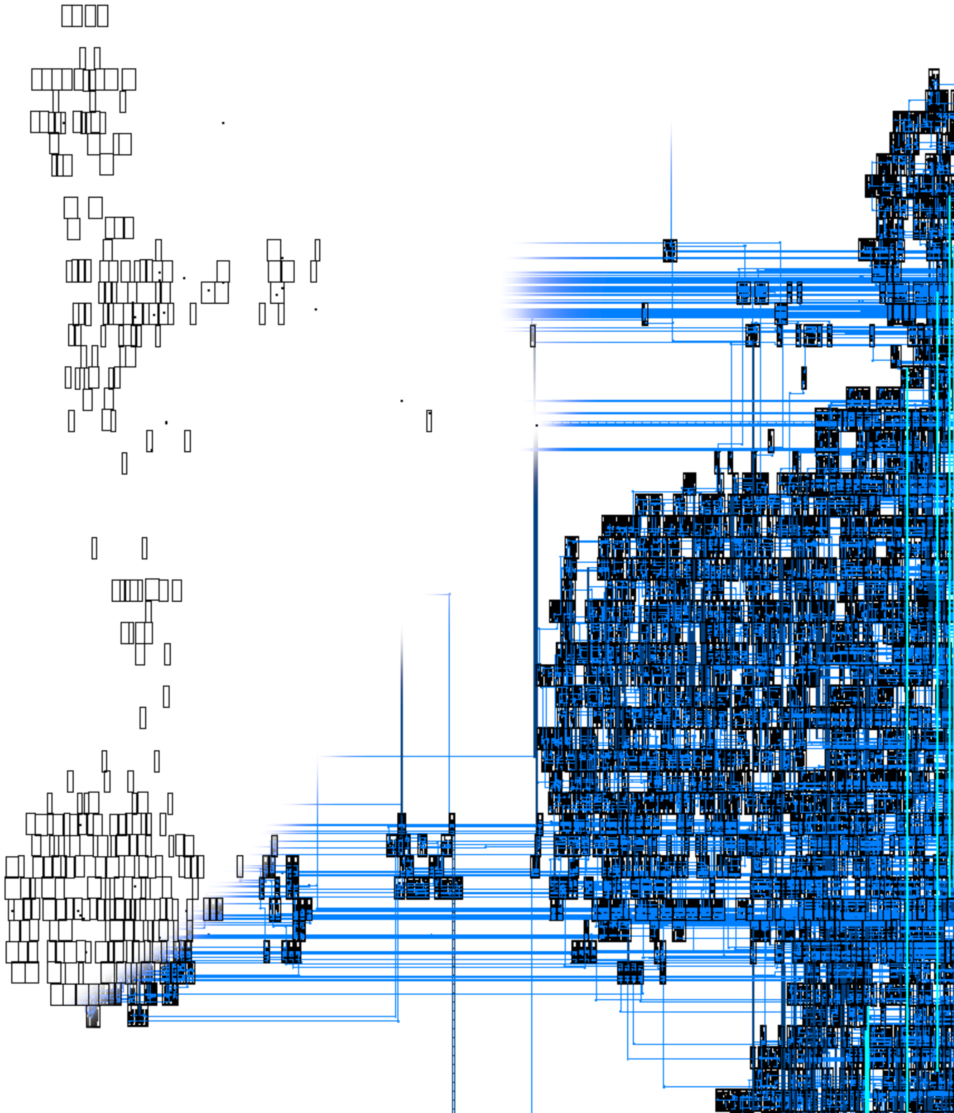
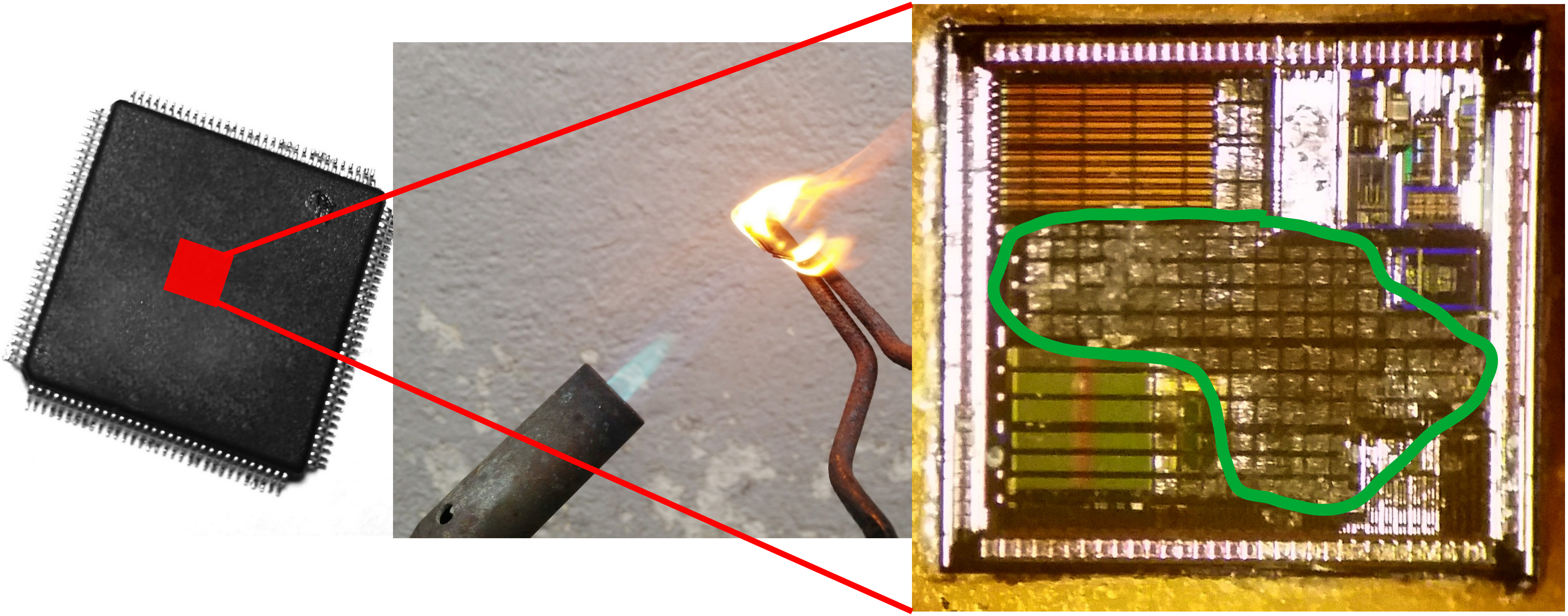
Place & Route on Silicon

a gentle introduction

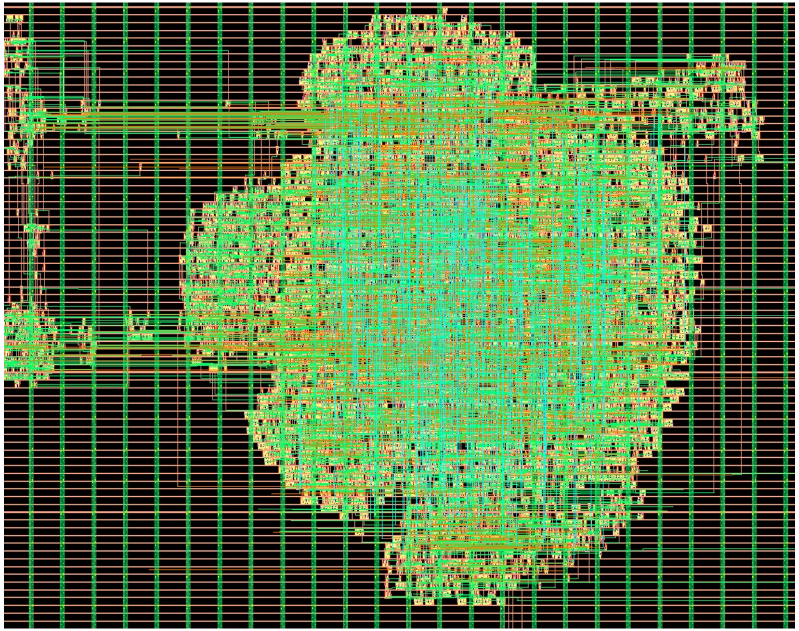
37c3@tkramer.ch





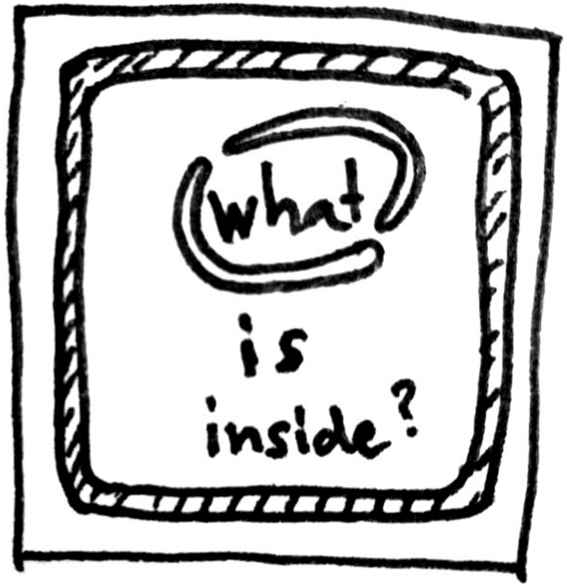
Today

- Place and Route
- How does the “silicon compiler” work
- Not today: How to use it



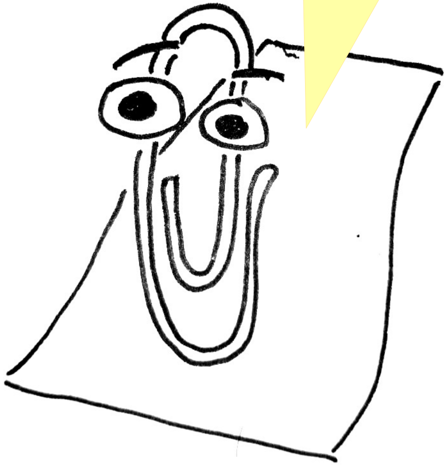
Motivation

- Most chips are proprietary
  - Obscurity
  - Documentation?
  - Firmware?
  - Use, study, share, improve?



Motivation

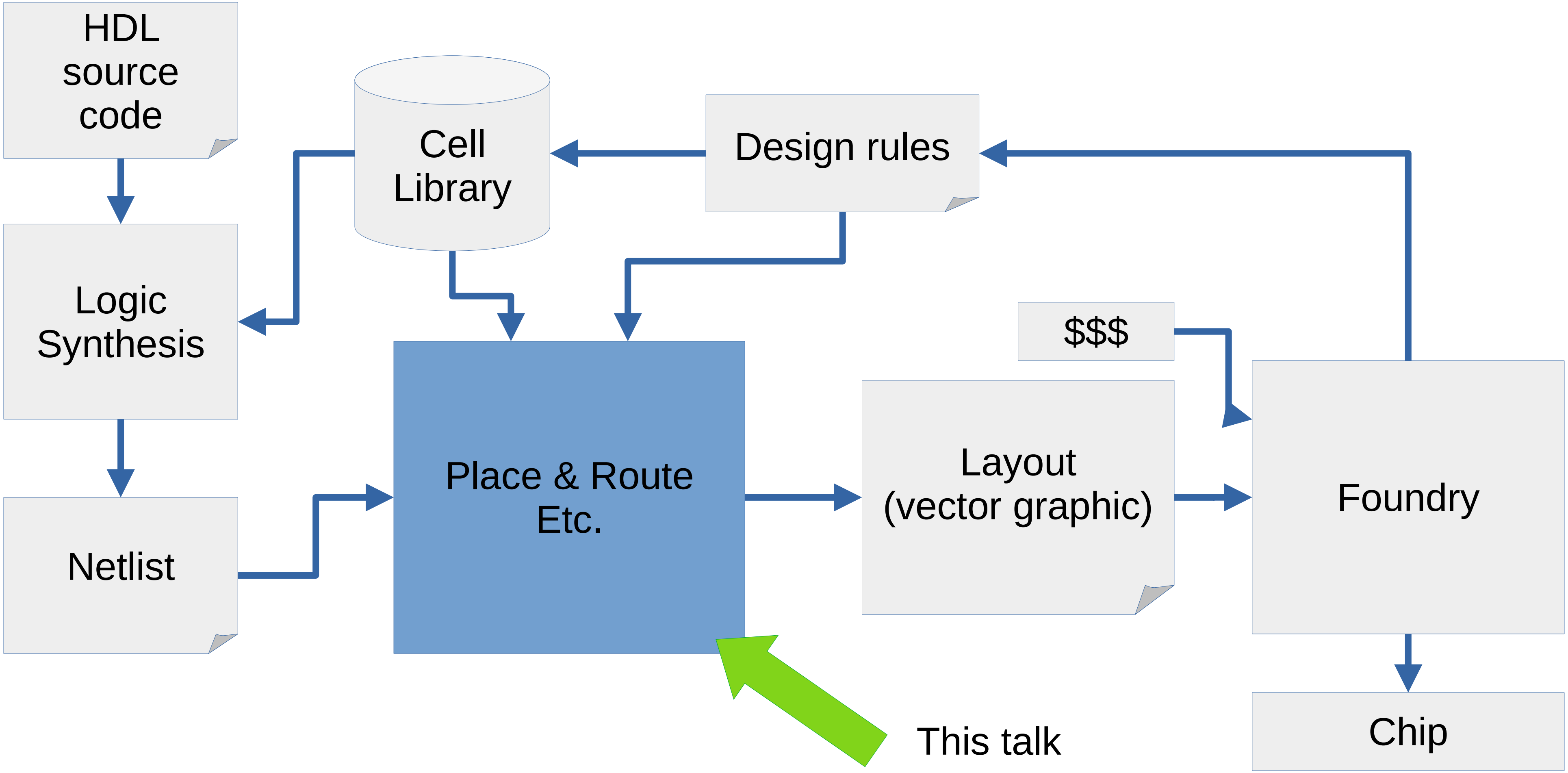
- Create chips, open down to the layout?
- Yes, but
  - Software vendors set rules (NDAs, restrictive licenses)
  - NDAs for process design kits
- FOSS toolchain!
  - Allows to implement novel ideas
  - Accessible for small entities
  - Useful for education



Why now?

- Many things happening recently
  - US: Skywater’s open process design kit (PDK)
  - US: Google, DARPA
  - US: OpenRoad (free software place & route toolchain)
  - Germany: IHP open source process design kit (PDK)
  - EU: ChipsAct
- Tech vendors become more restrictive (personal opinion)
  - Time to have alternatives

Digital Chip Design Flow



HDL source code

Logic Synthesis

Netlist

Cell Library

Design rules

Place & Route Etc.

Layout (vector graphic)

$$$

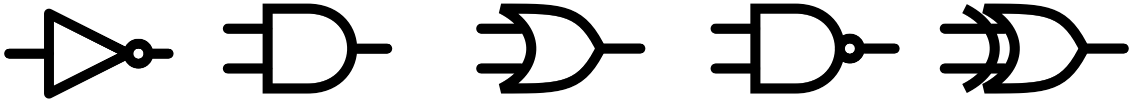
Foundry

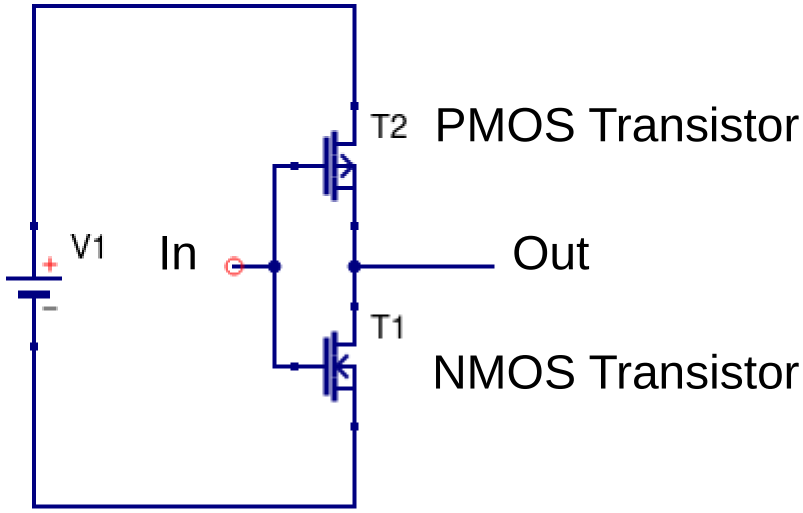
Chip

This talk

Digital Circuits / Logic

- Logic “gates”
  - small boolean functions
  - Inverter, AND, OR, NAND, XOR, ...

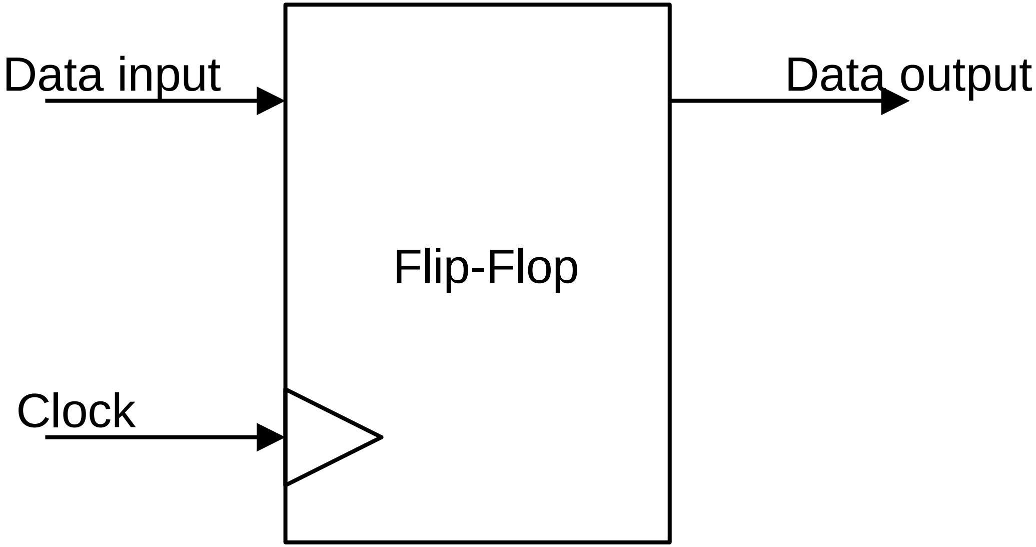




CMOS inverter

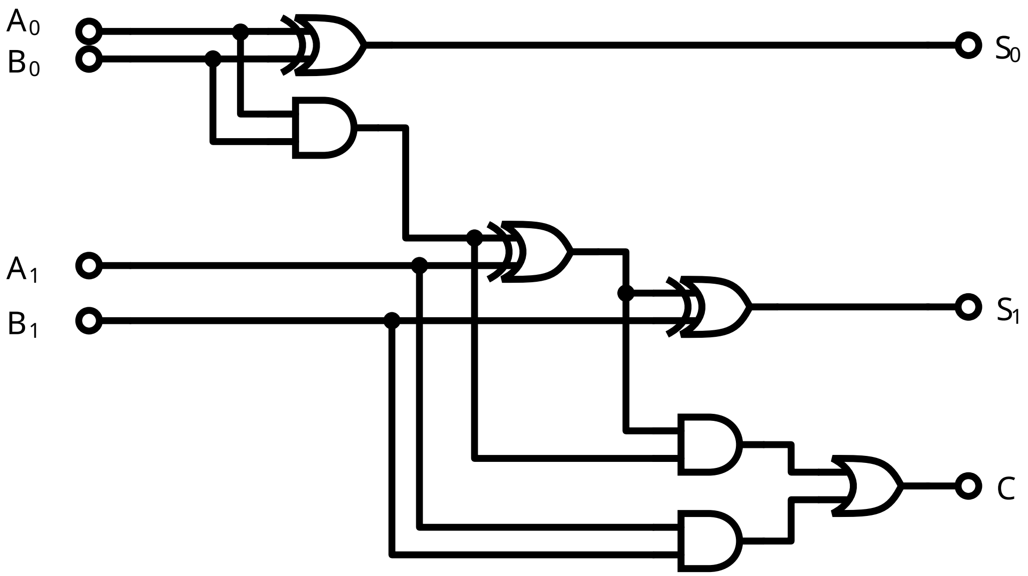
Digital Circuits / Logic

- Storage elements
- Update output on clock edge

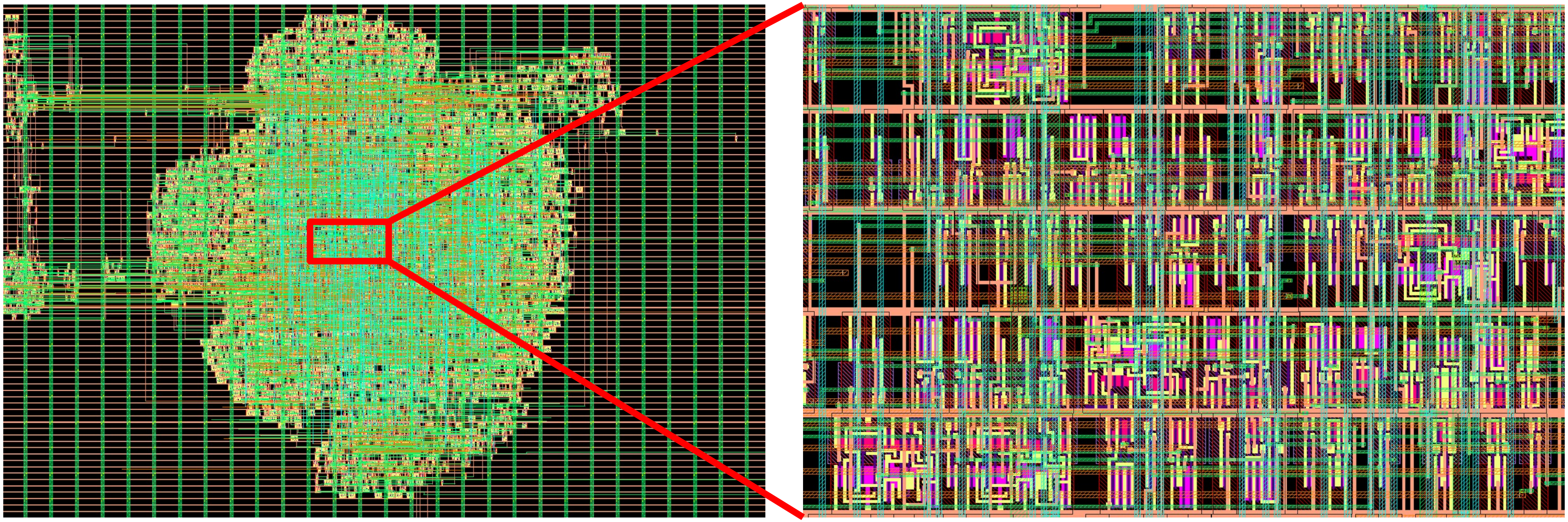


Digital Circuits / Logic

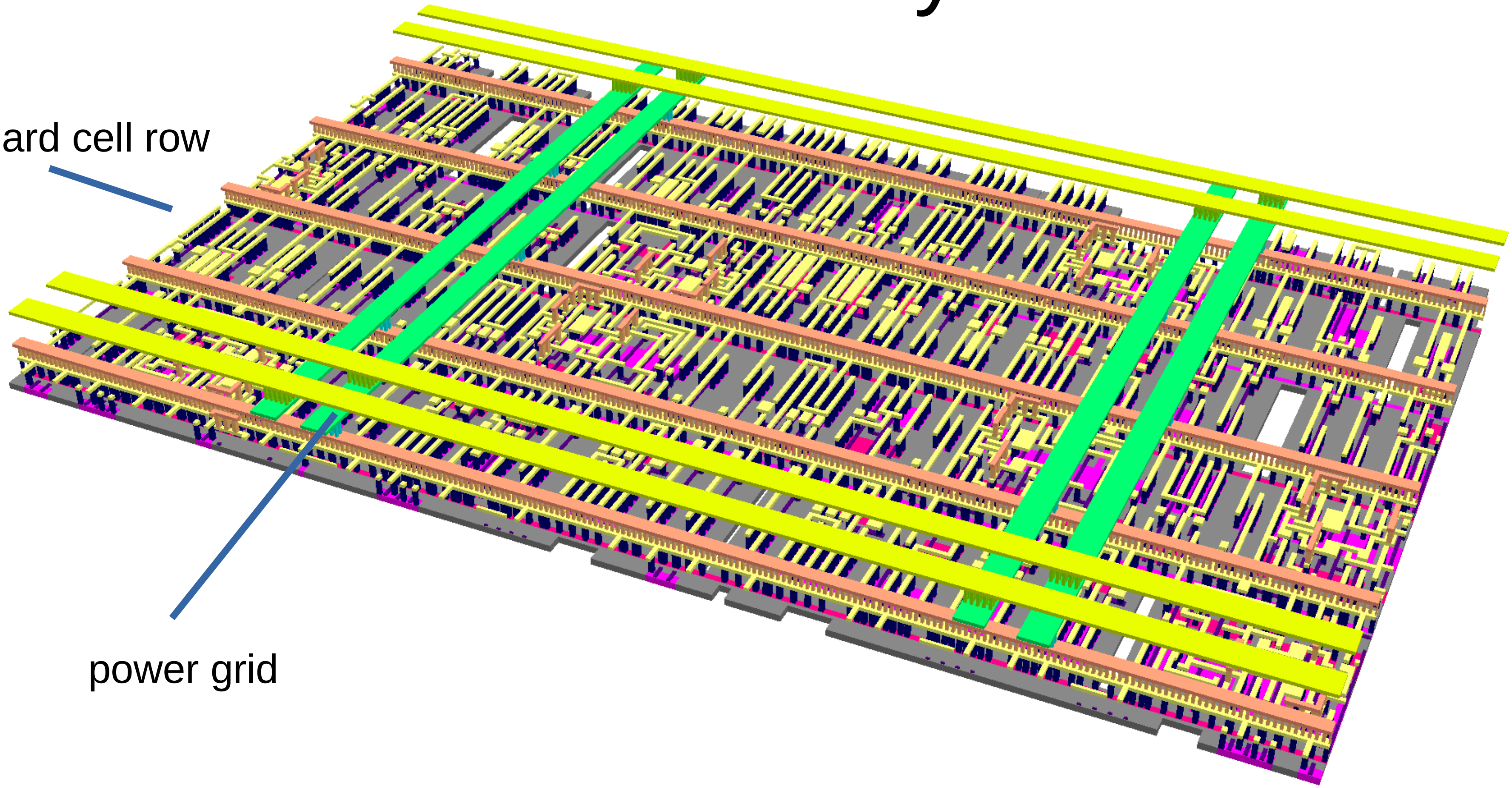
- Assemble gates to large circuits
- Up to millions of gates



Row-Based Layouts



Row-based layouts

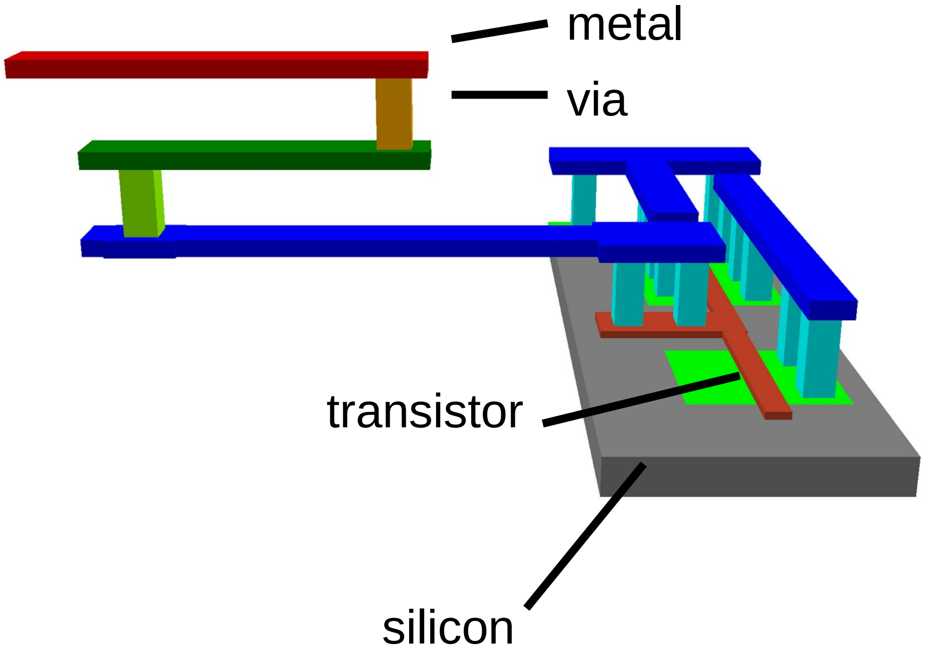


standard cell row

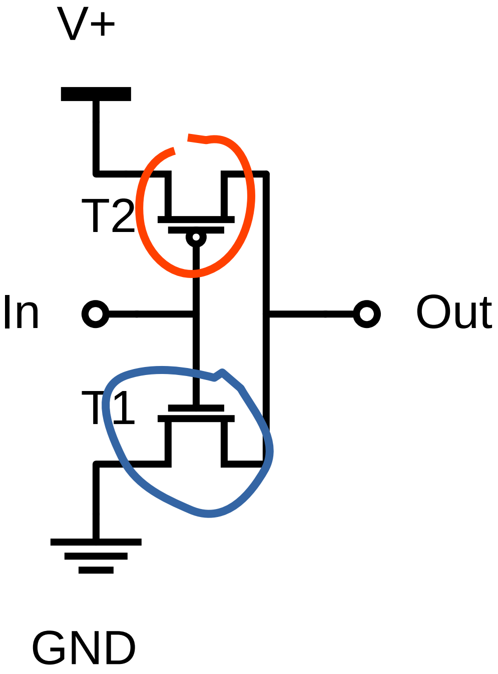
power grid

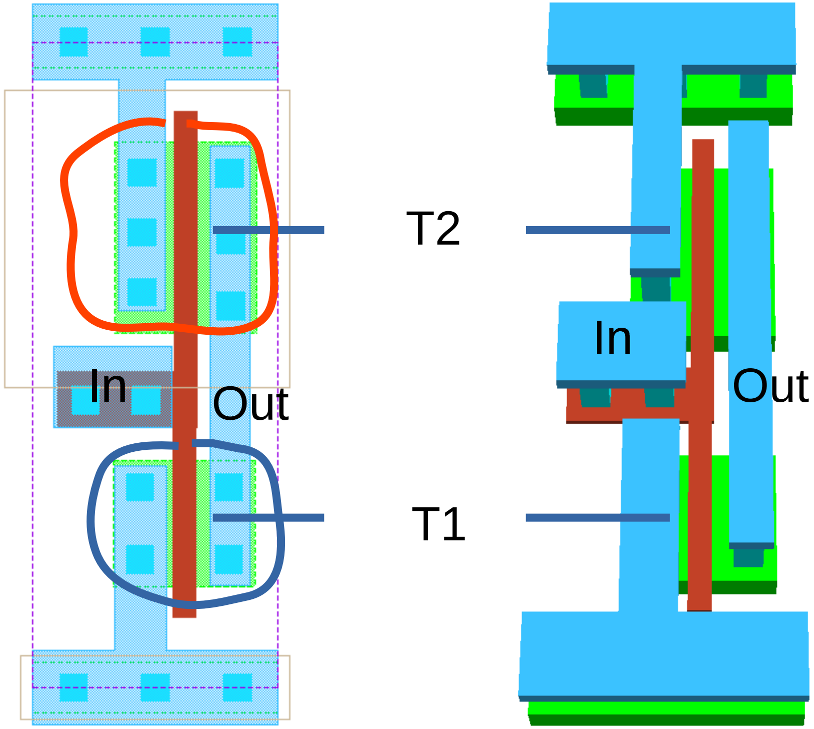
CMOS Layerstack

- Silicon, doping, poly-silicon
  - Build transistors
  - Ignore for place & route
- Metal layers and vias
  - Build wires



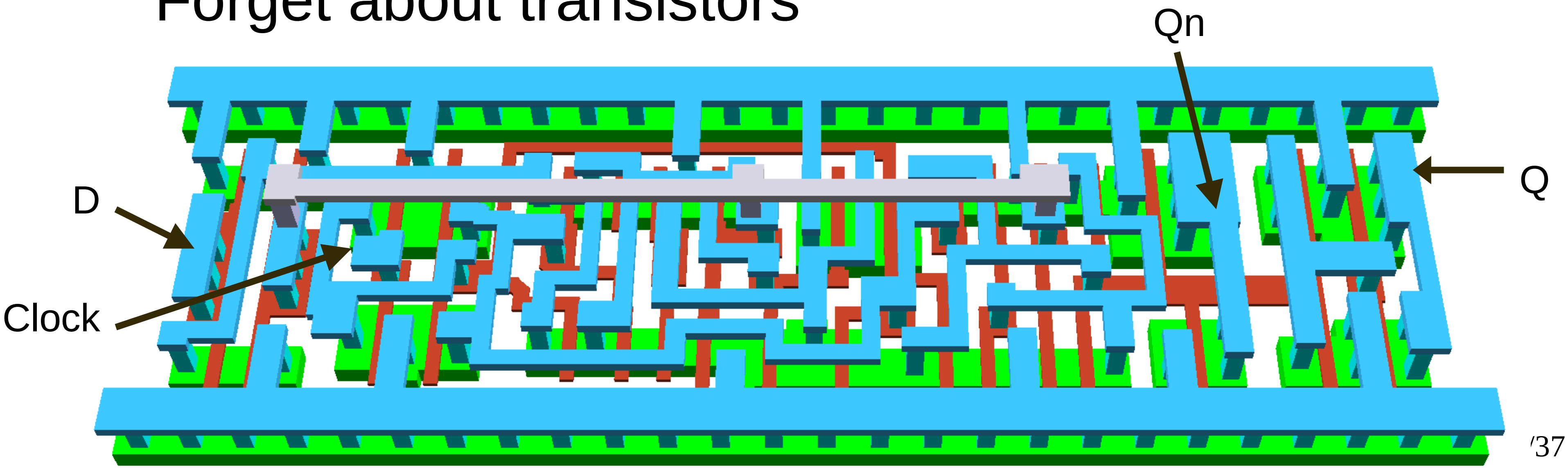
Standard Cells





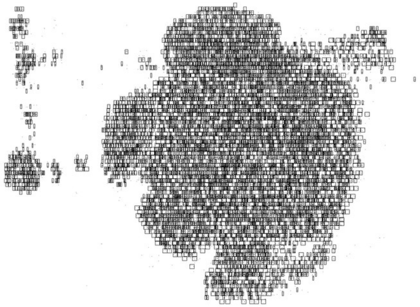
Standard Cells

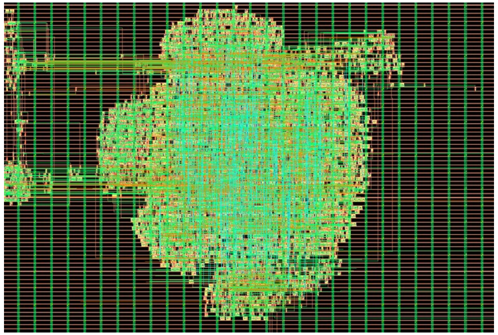
- Simplify place & route
  - Care about metal layers and pins only
  - Forget about transistors



Place and Route

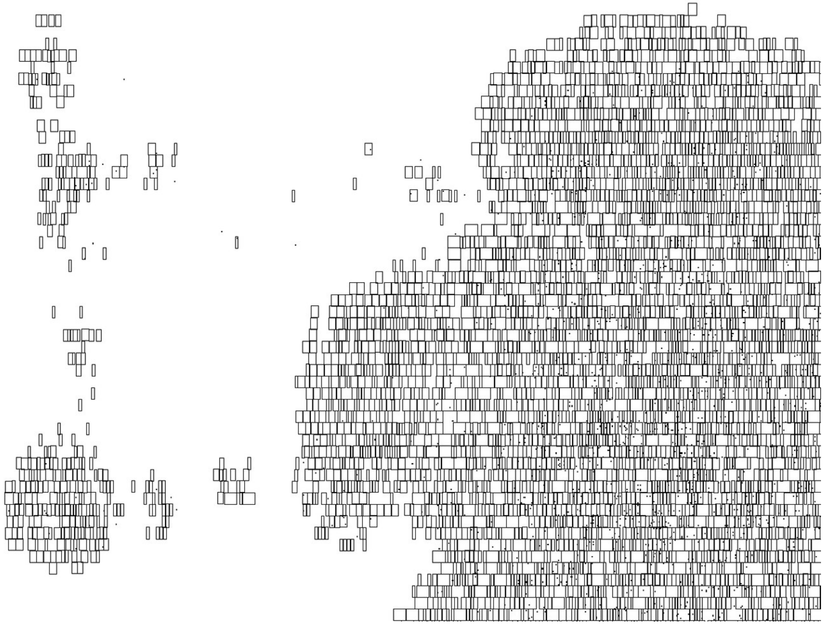
- Place components
- Create clock-tree
- Route regular nets
- Check and fix timing
- Verify
  - Layout versus schematic (LVS)
  - Design rule checks (DRC)





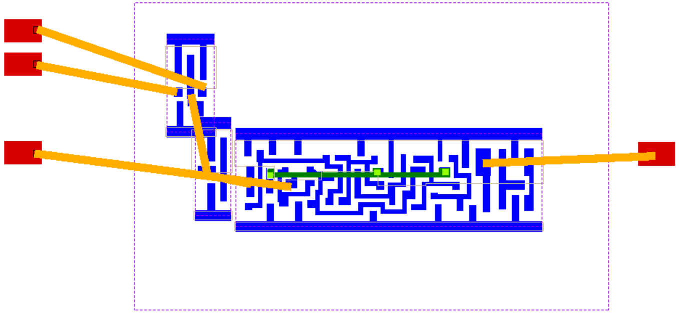
Placement Problem

- Find positions for cells
- Optimize wire-length
- Constraints
  - Components cannot overlap
  - Routing should be possible

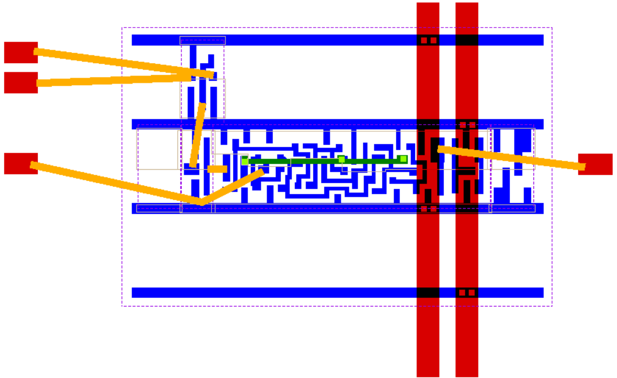


Breaking Down the Problem

- Global placement
  - Find approximate locations for components
  - Allow small overlaps but respect density

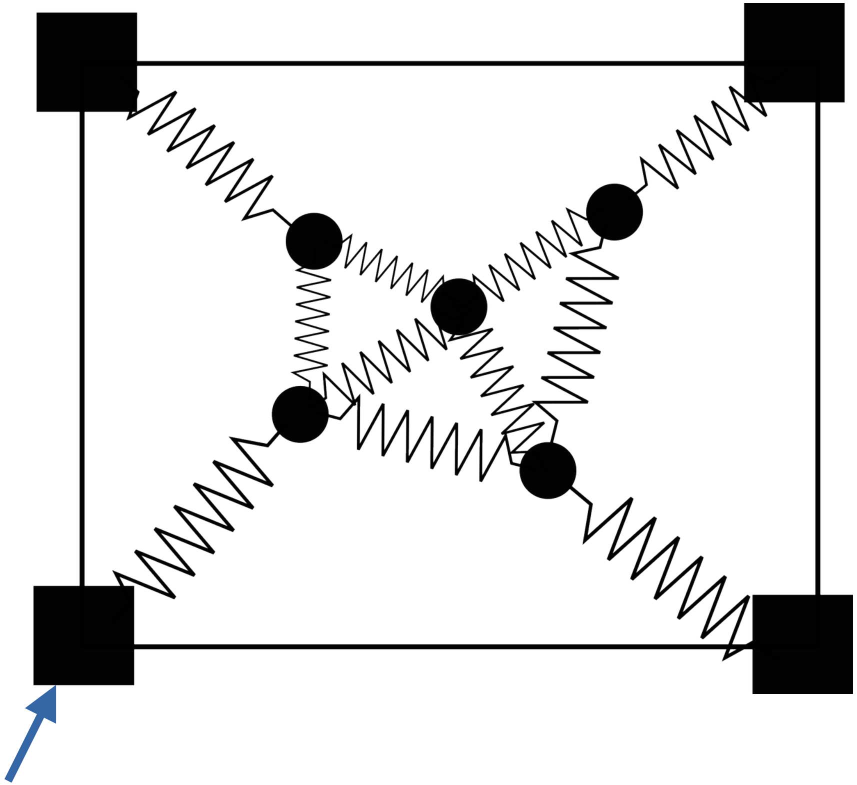


- Legalization
  - Resolve overlaps
  - Snap to grid
- Detail placement
  - Local optimizations



Example: Quadratic Placement

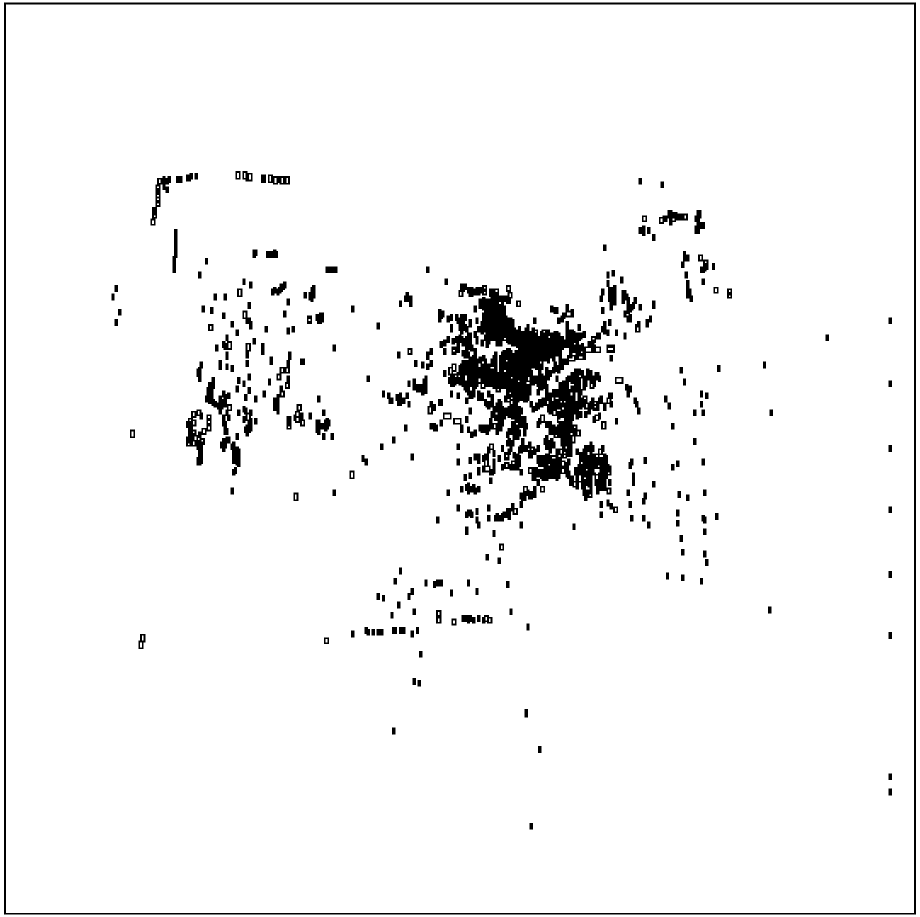
- Idea: connected cells attract each other
  - Simulate a network of springs
  - Ignore overlaps
- Minimize the sum of squared wire-lengths
- “Analytic” placement
  - Minimize a cost function
  - Gradient descent
- Problem: high overlap



Fixed

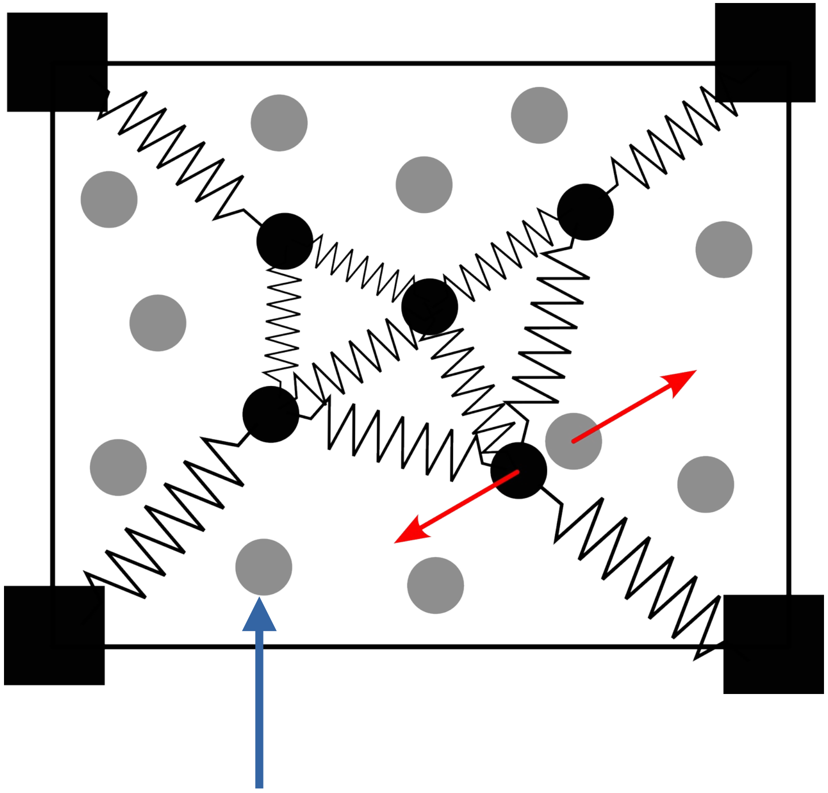
Example: Quadratic Placement

- Problem: high overlap
- Idea: add repulsive force



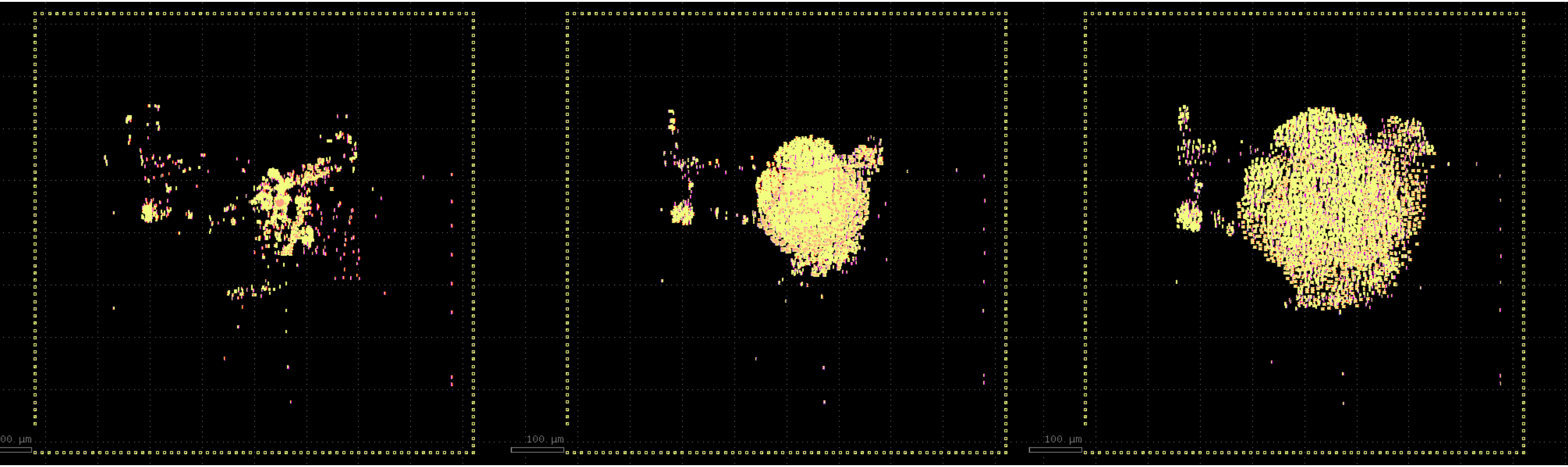
Example: solve density constraint

- Idea: simulate charged particles
- Repulsion reduces overlap
- Used by “ePlace”



filler cells

Effect of Electrostatic Repulsion

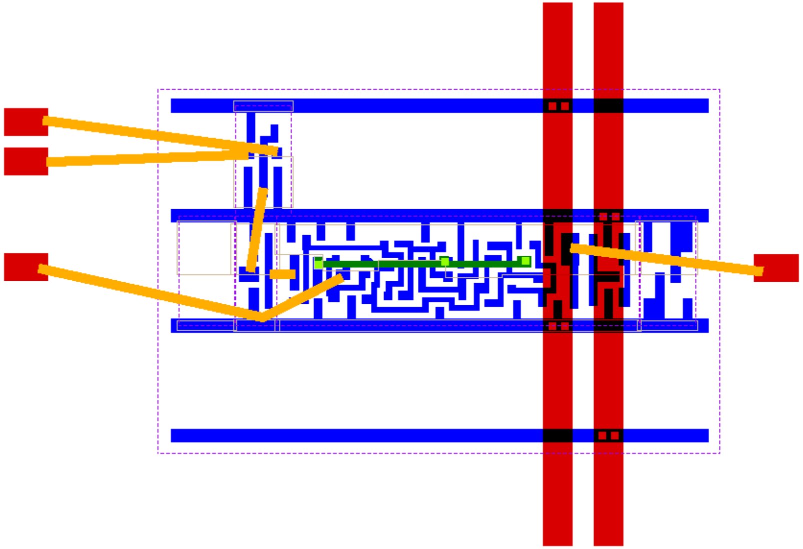


minimized quadratic wire-length

solved density constraint

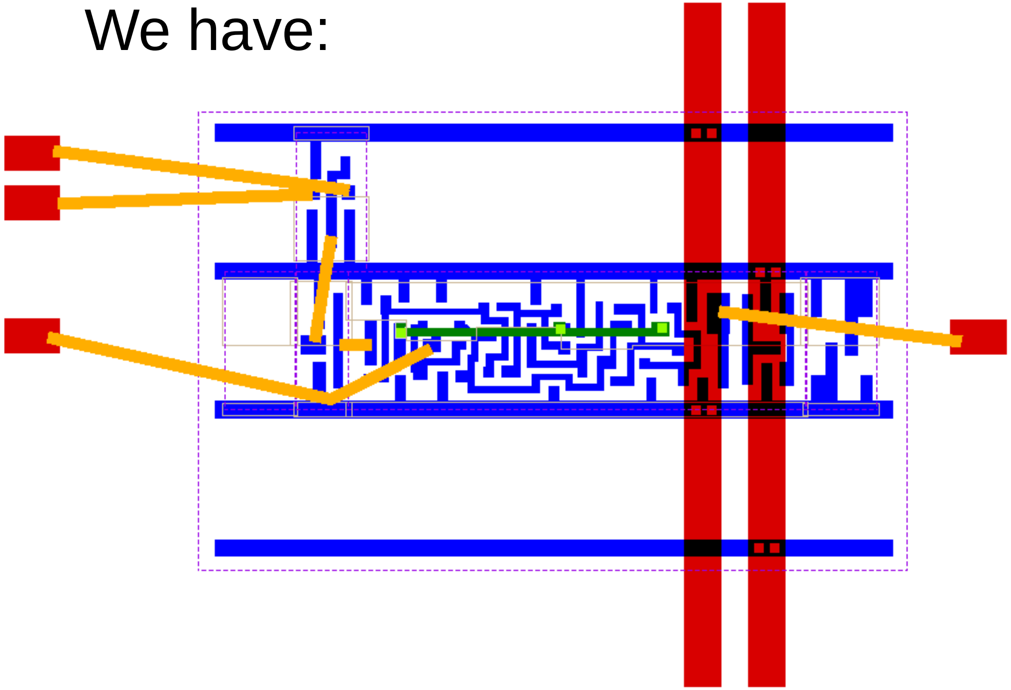
Routing Problem

- Goal: Correct electrical connections
- Constraint: layers, design rules
  - Minimum spacing
  - Minimum width
  - Others (antenna rules, ...)
- Route special nets separately
  - Supply voltages → power grid
  - Clock → clock-tree synthesis

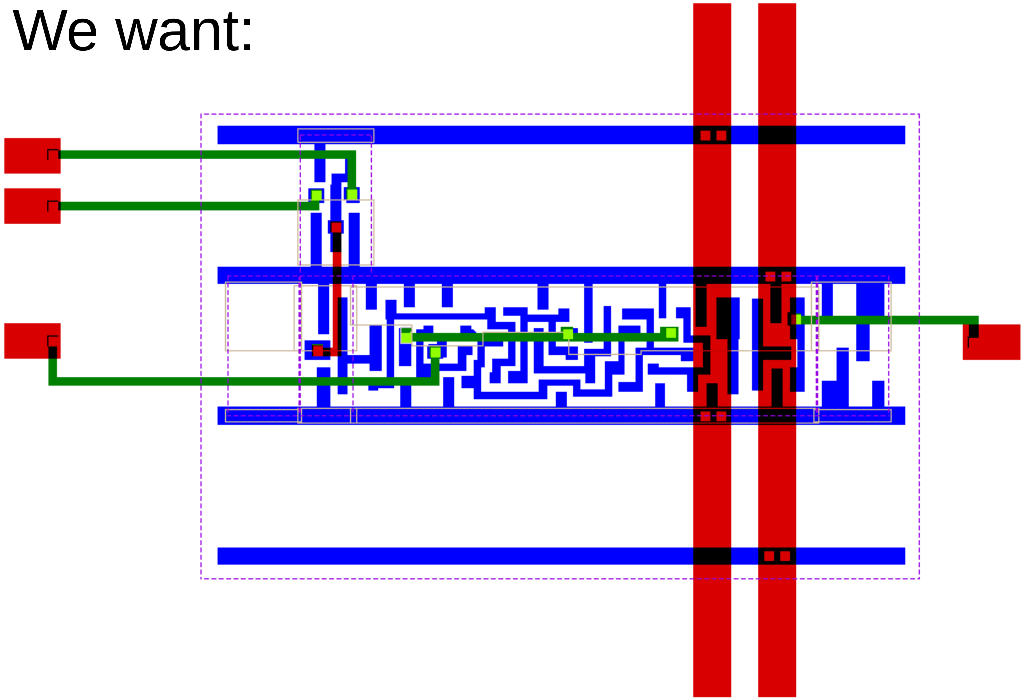


Routing Problem

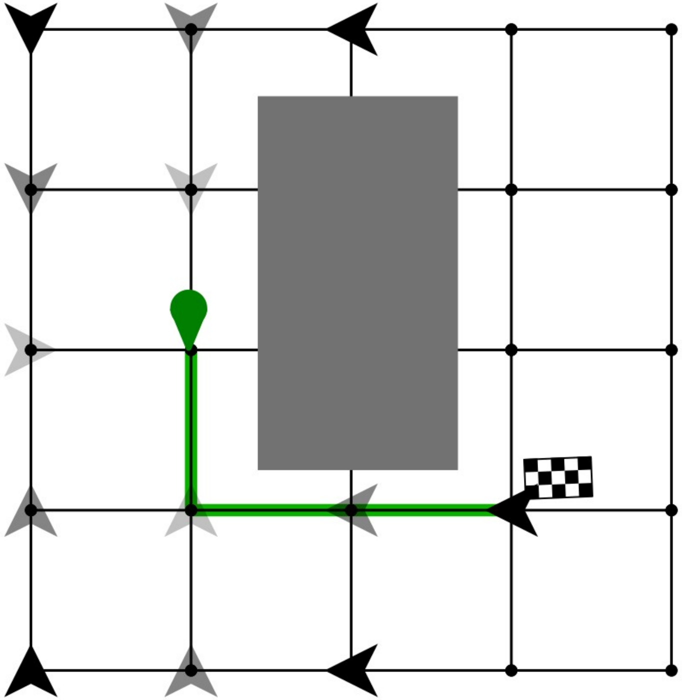
We have:



We want:



“Maze” Routing



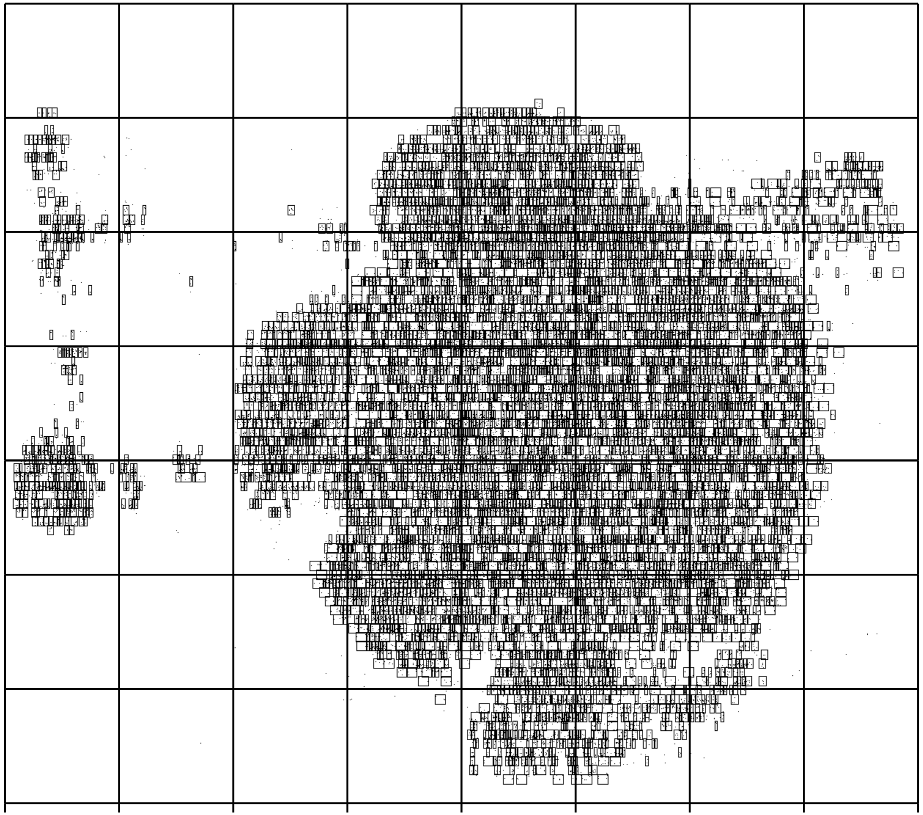
Breaking Down the Problem

- Global routing
  - Find approximate routes
  - Relax design rules
  - Resolve congestion

- Detail routing
  - Final wiring
  - Refine global routes
  - Respect design rules

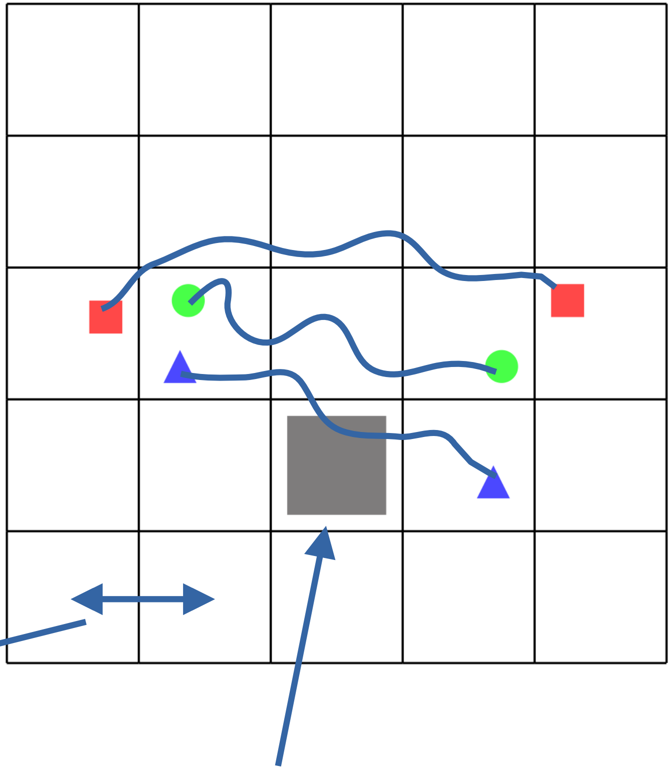
Global Routing

- Find routes on coarse grid



Global Routing

- Example: Connect the three pairs of pins
- Max. two wires can cross between tiles

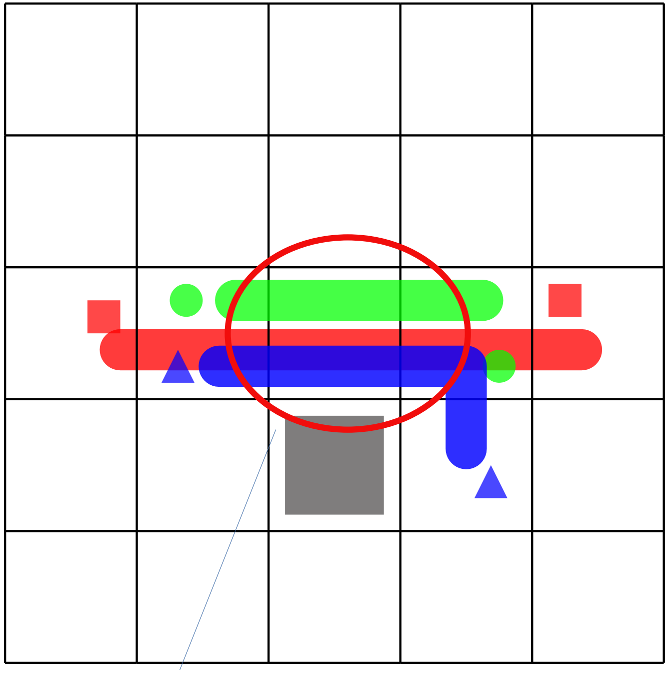


capacity = 2
cost = 1

obstacle, capacity = 0

Global Routing

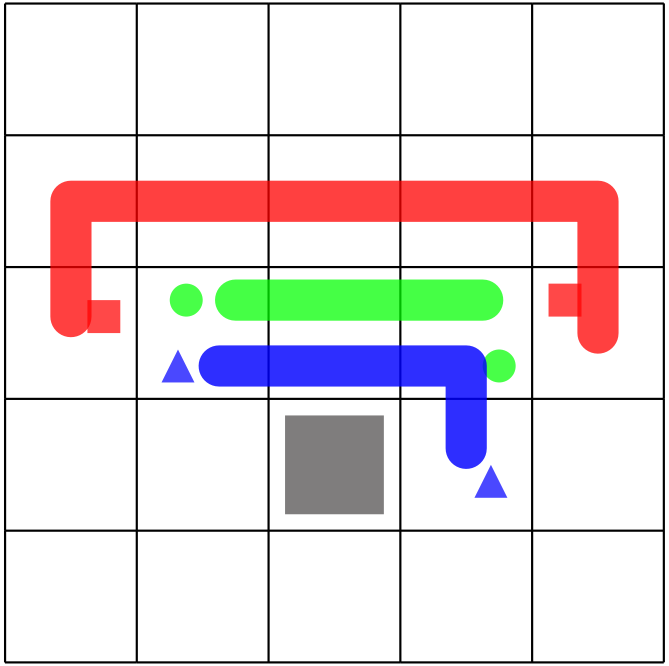
- “Pathfinder” algorithm
  - Route all nets
  - Increase costs for congested edges
  - Iterate



conflict
capacity = 2
usage = 3

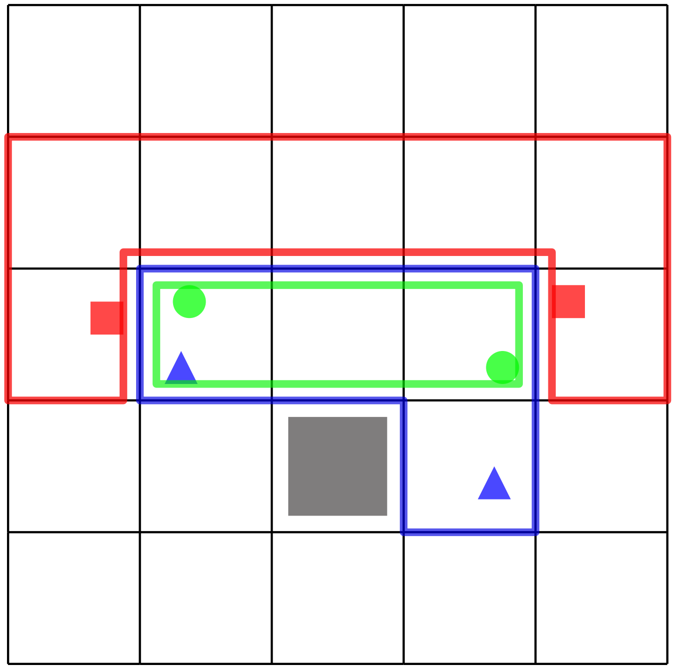
Global Routing

- Push routes away from congested regions
- Use result for detail routing



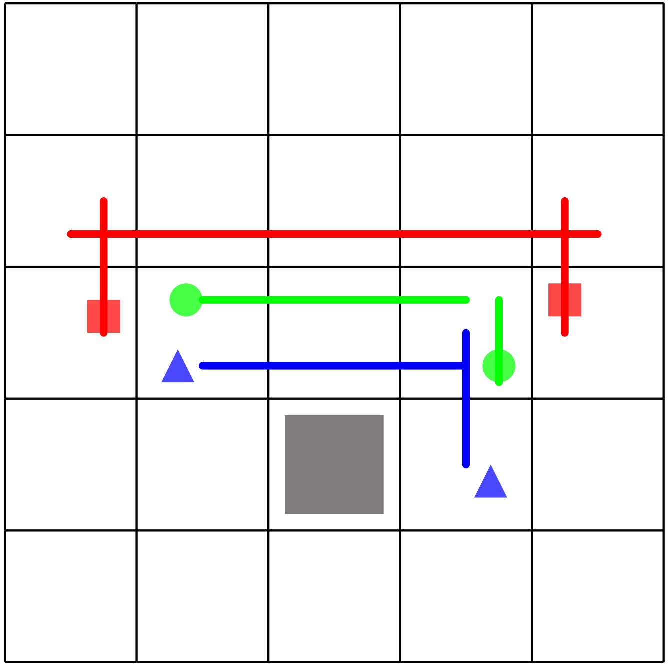
Detail routing

- Find final wiring
- Refine global routes



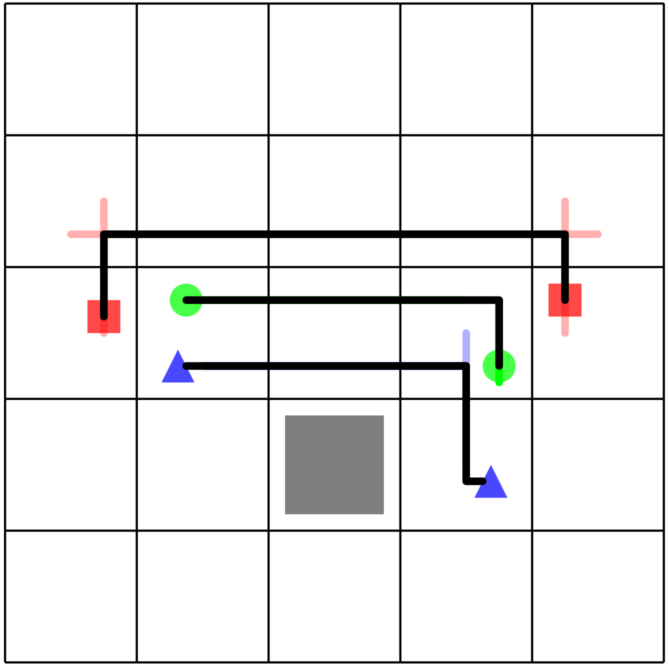
Detail routing

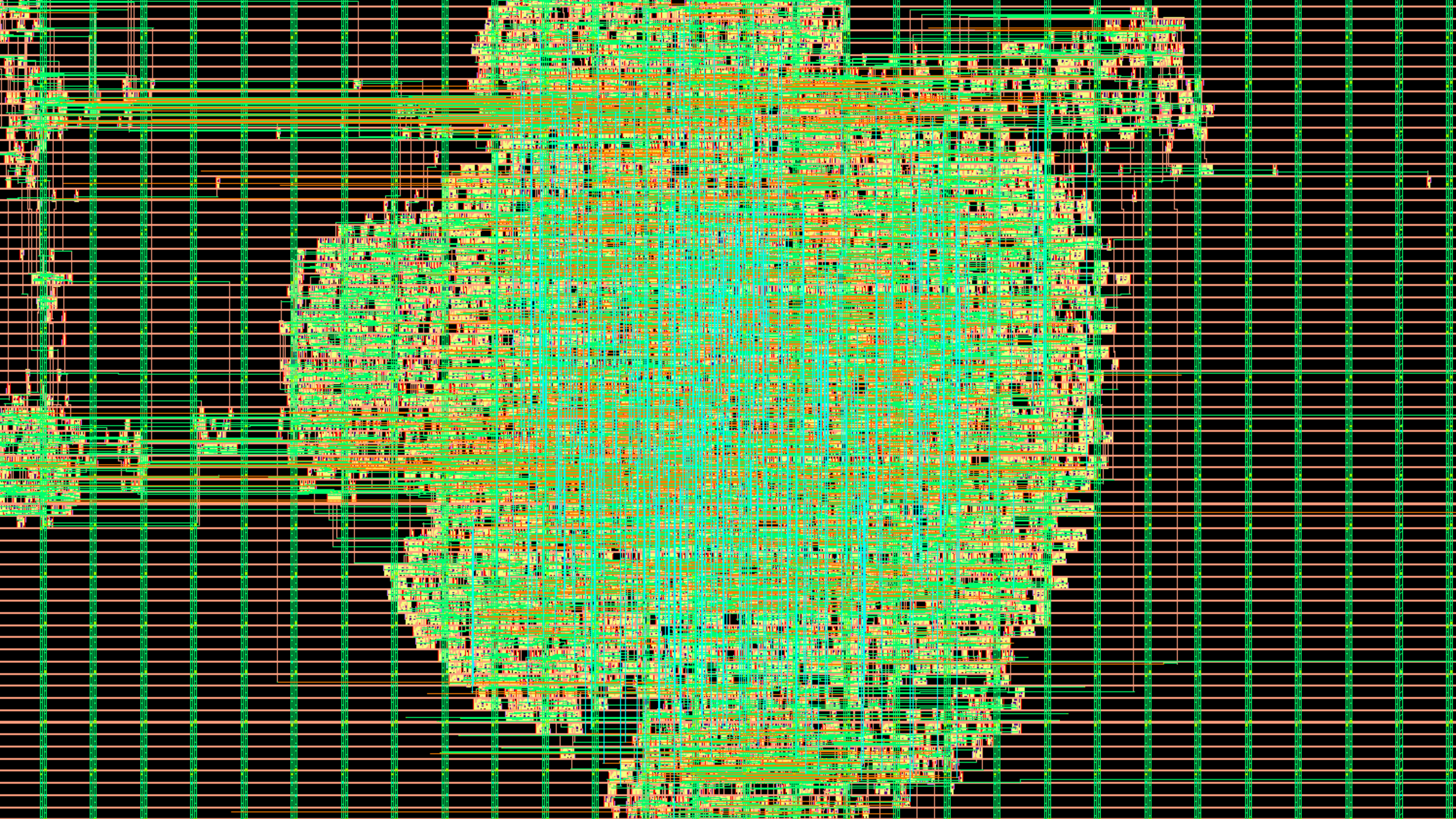
- Convert global routes to narrow lines
- Assign lines to tracks
- Minimize conflicts



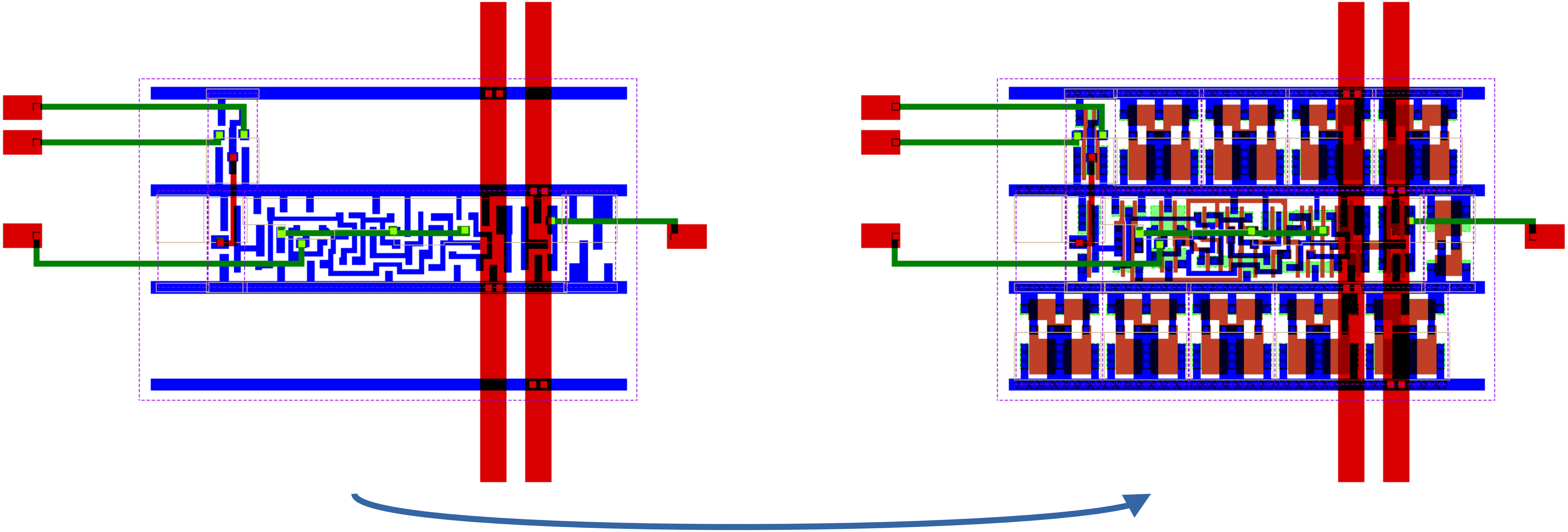
Detail routing

- Fill gaps by maze routing
- Check rule violations
- Fix violations





Make Use of Empty Space



put capacitors

More...

- “VLSI Physical Design: From Graph Partitioning to Timing Closure”, Andrew B. Kahng
- Free Silicon Conference, Paris (f-si.org)
- OpenRoad (theopenroadproject.org)
- Coriolis (coriolis.lip6.fr)
- LunaPnR (nlnet.nl/project/Luna/)
- LibrEDA.org

Thank you!

37c3@tkramer.ch