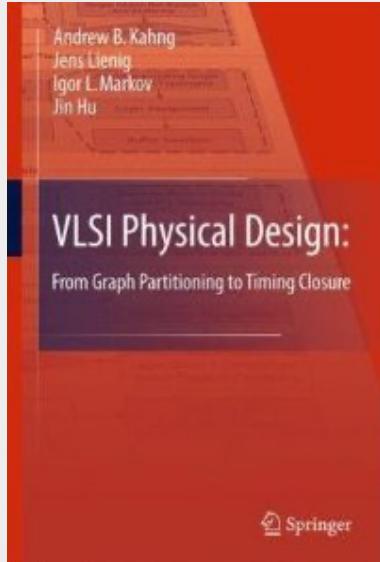




# *CSC688E: Algorithms for VLSI Design Automation*

## **Course Introduction**

# Textbook



Authors:

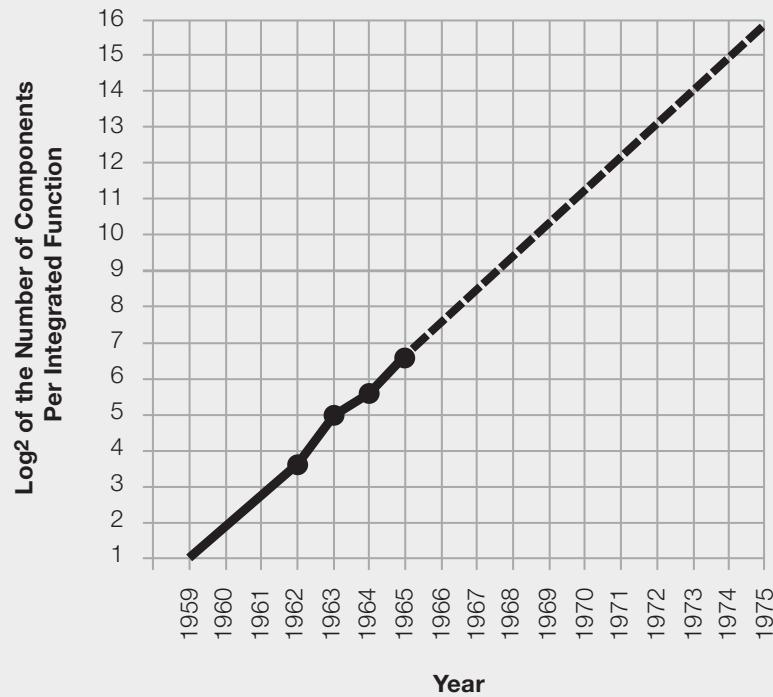
Andrew B. Kahng, Jens Lienig, Igor L. Markov, Jin Hu

# Lecture 1

- 1.1 Electronic Design Automation (EDA)
- 1.2 VLSI Design Flow
- 1.3 VLSI Design Styles
- 1.4 Layout Layers and Design Rules
- 1.5 Physical Design Optimizations
- 1.6 Algorithms and Complexity
- 1.7 Graph Theory Terminology
- 1.8 Common EDA Terminology

## Moore's Law

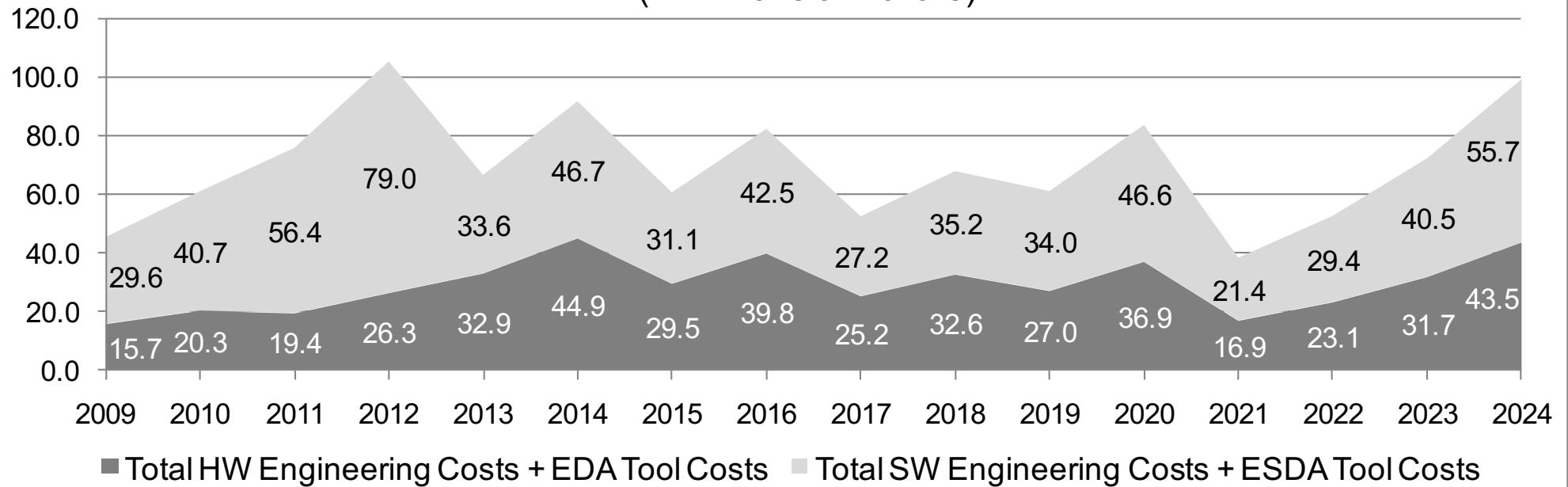
In 1965, Gordon Moore (Fairchild) stated that the number of transistors on an IC would double every year. 10 years later, he revised his statement, asserting that they double every 18 months. Since then, this “rule” has been famously known as Moore’s Law.



Moore: „Cramming more components onto integrated circuits“  
Electronics, Vol. 38, No. 8, 1965

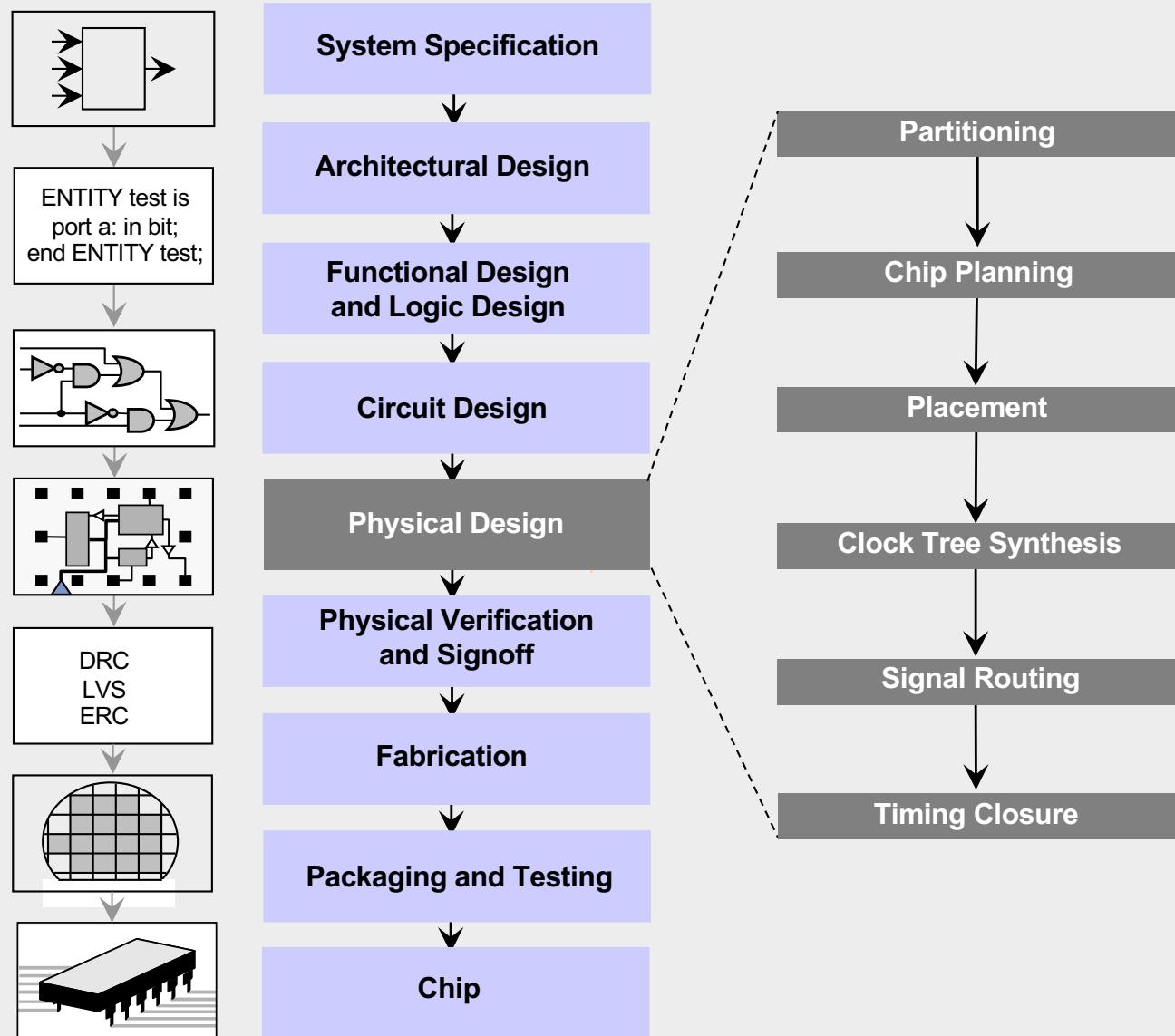
Impact of EDA technologies on overall IC design productivity and IC design cost

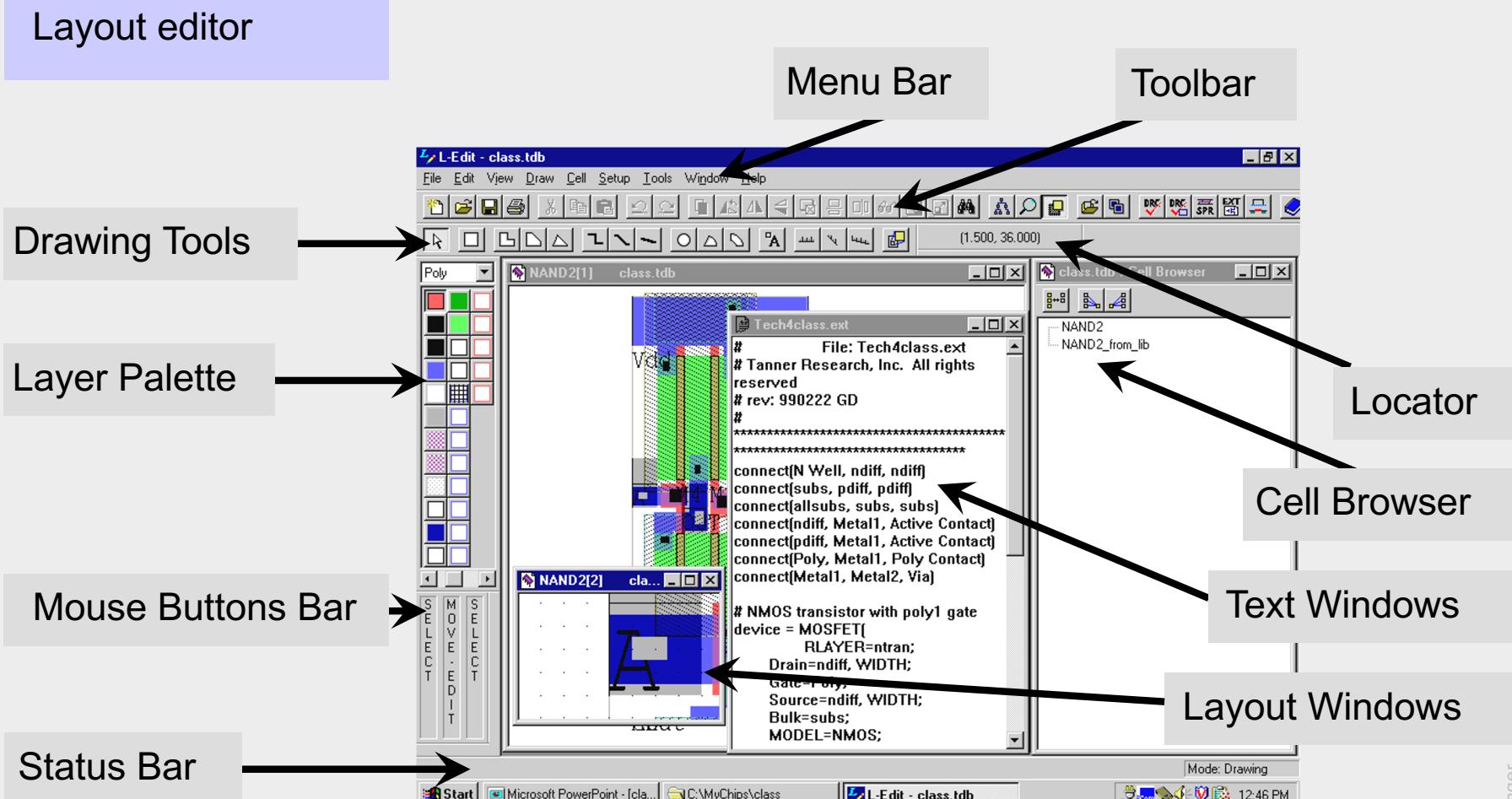
ITRS 2009 Cost Chart  
(in Millions of Dollars)



## 1.1 Electronic Design Automation (EDA)

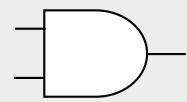
Time Period	Circuit and Physical Design Process Advancements
1950 -1965	Manual design only.
1965 -1975	Layout editors, e.g., place and route tools, first developed for printed circuit boards.
1975 -1985	More advanced tools for ICs and PCBs, with more sophisticated algorithms.
1985 -1990	First performance-driven tools and parallel optimization algorithms for layout; better understanding of underlying theory (graph theory, solution complexity, etc.).
1990 -2000	First over-the-cell routing, first 3D and multilayer placement and routing techniques developed. Automated circuit synthesis and routability-oriented design become dominant. Start of parallelizing workloads. Emergence of physical synthesis.
2000 - now	Design for Manufacturability (DFM), optical proximity correction (OPC), and other techniques emerge at the design-manufacturing interface. Increased reusability of blocks, including intellectual property (IP) blocks.



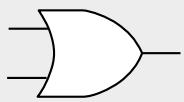


## Common digital cells

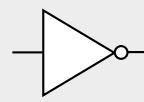
AND



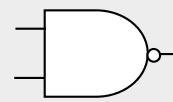
OR



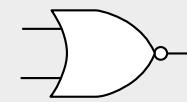
INV



NAND



NOR



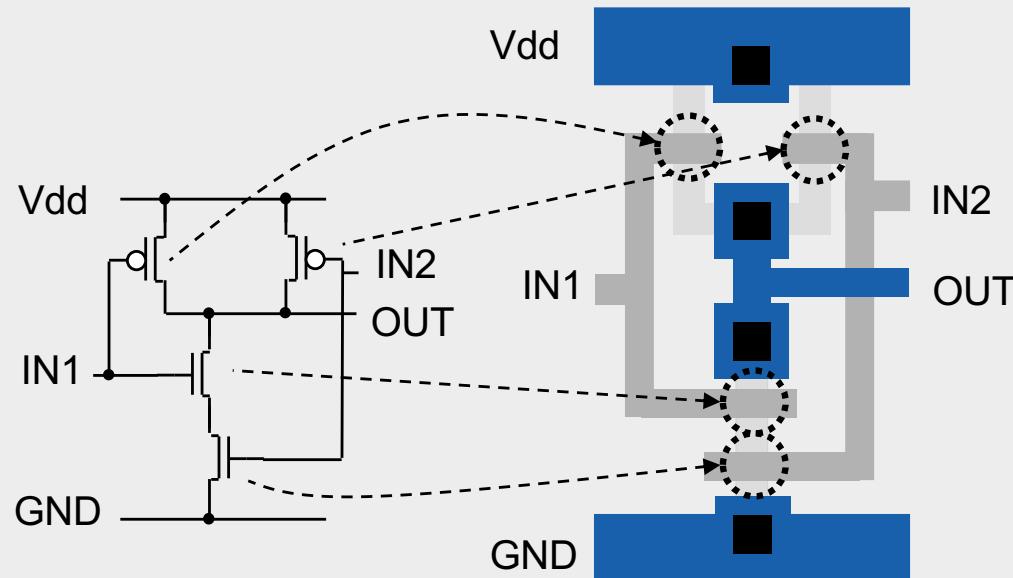
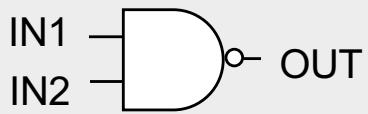
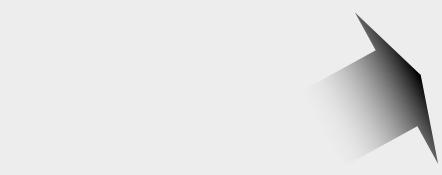
IN1	IN2	OUT
0	0	0
1	0	0
0	1	0
1	1	1

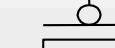
IN1	IN2	OUT
0	0	0
1	0	1
0	1	1
1	1	1

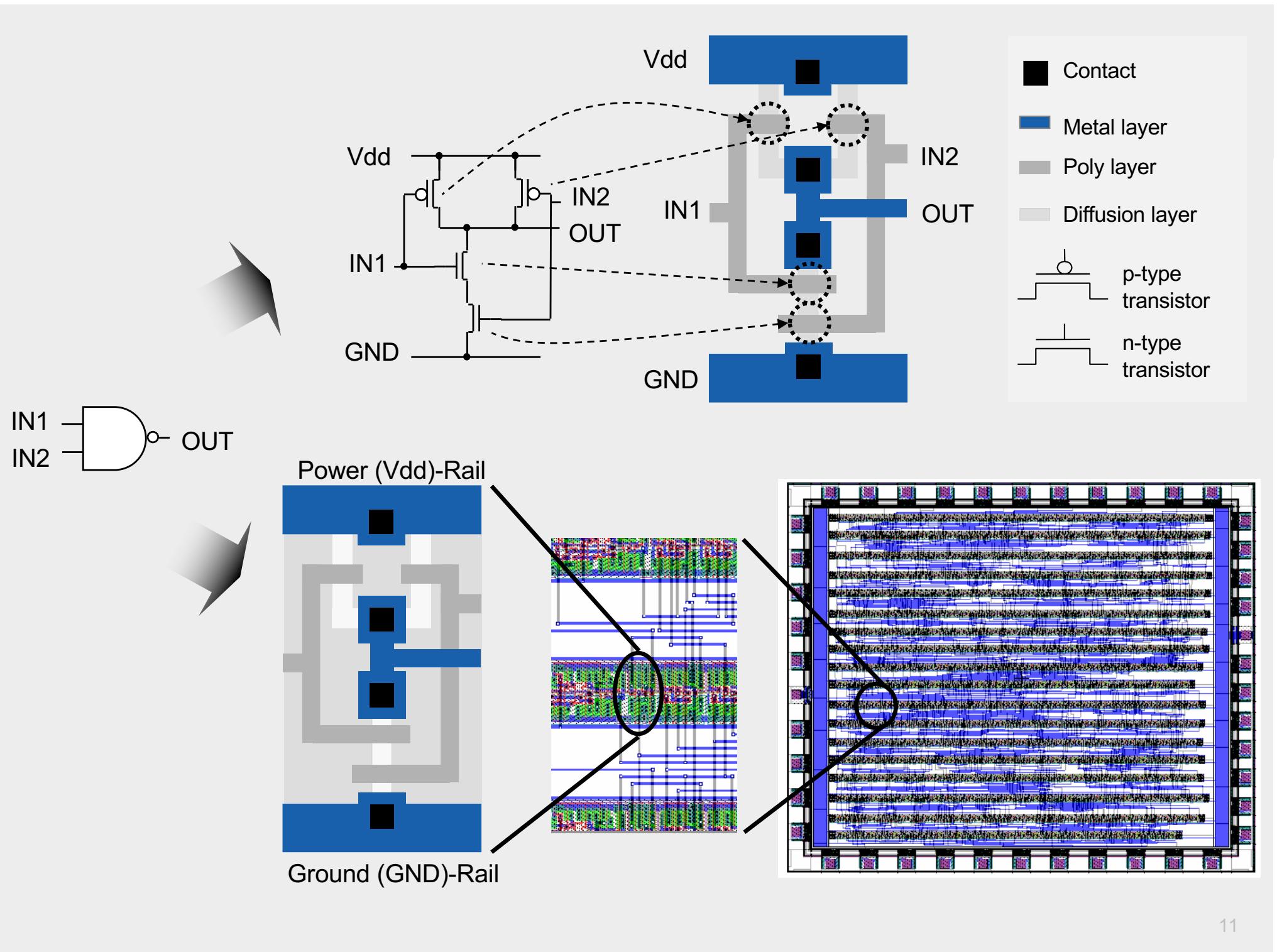
IN	OUT
0	1
1	0

IN1	IN2	OUT
0	0	1
1	0	1
0	1	1
1	1	0

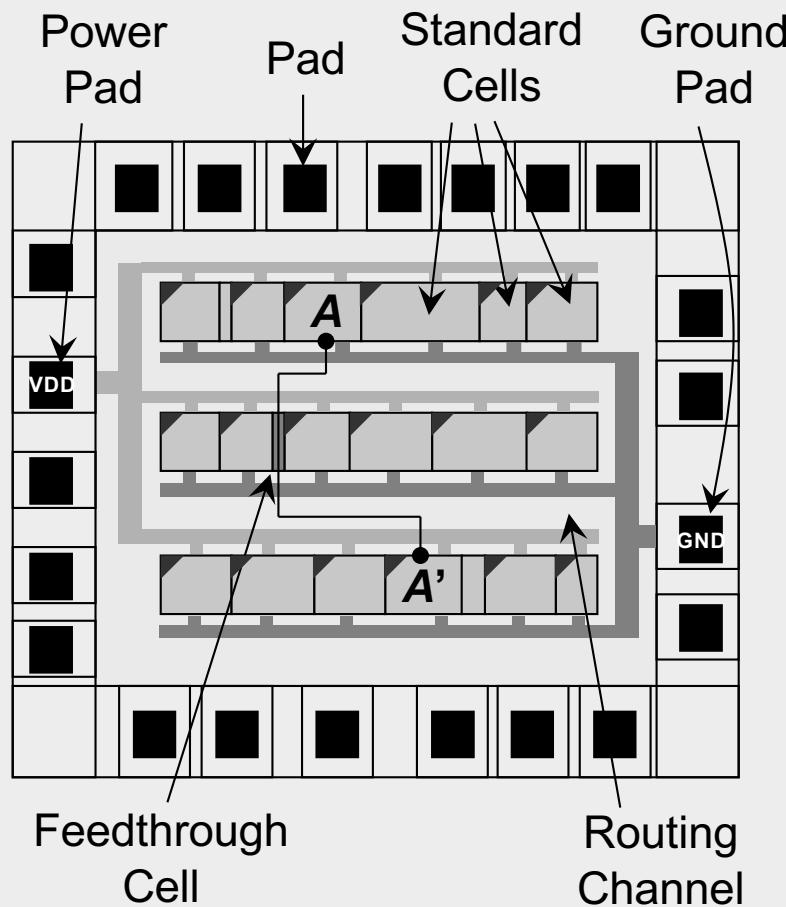
IN1	IN2	OUT
0	0	1
1	0	0
0	1	0
1	1	0



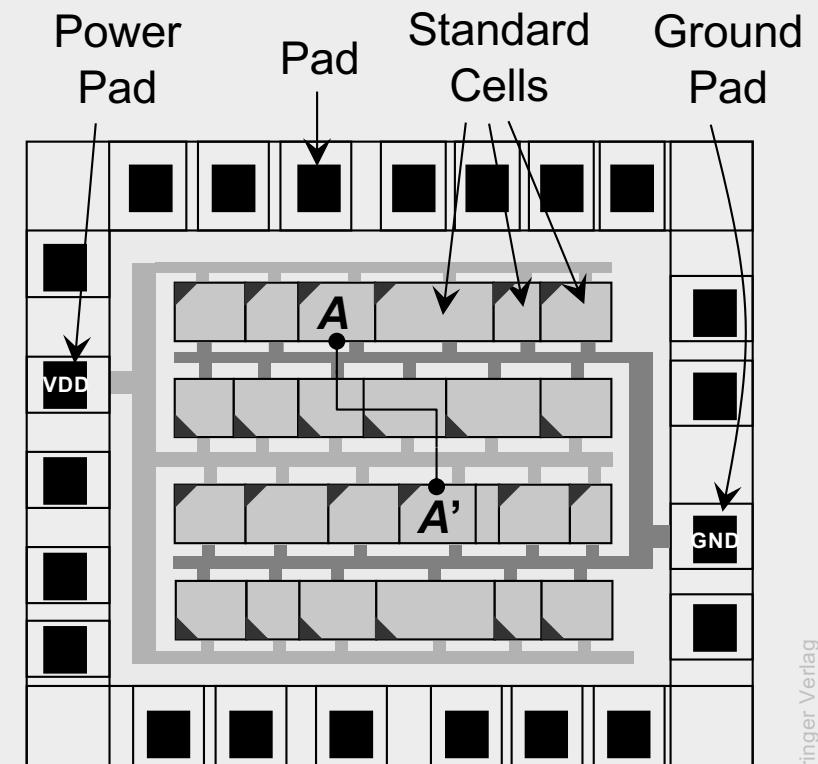
[Black square]	Contact
[Blue rectangle]	Metal layer
[Grey rectangle]	Poly layer
[Light grey rectangle]	Diffusion layer
	p-type transistor
	n-type transistor



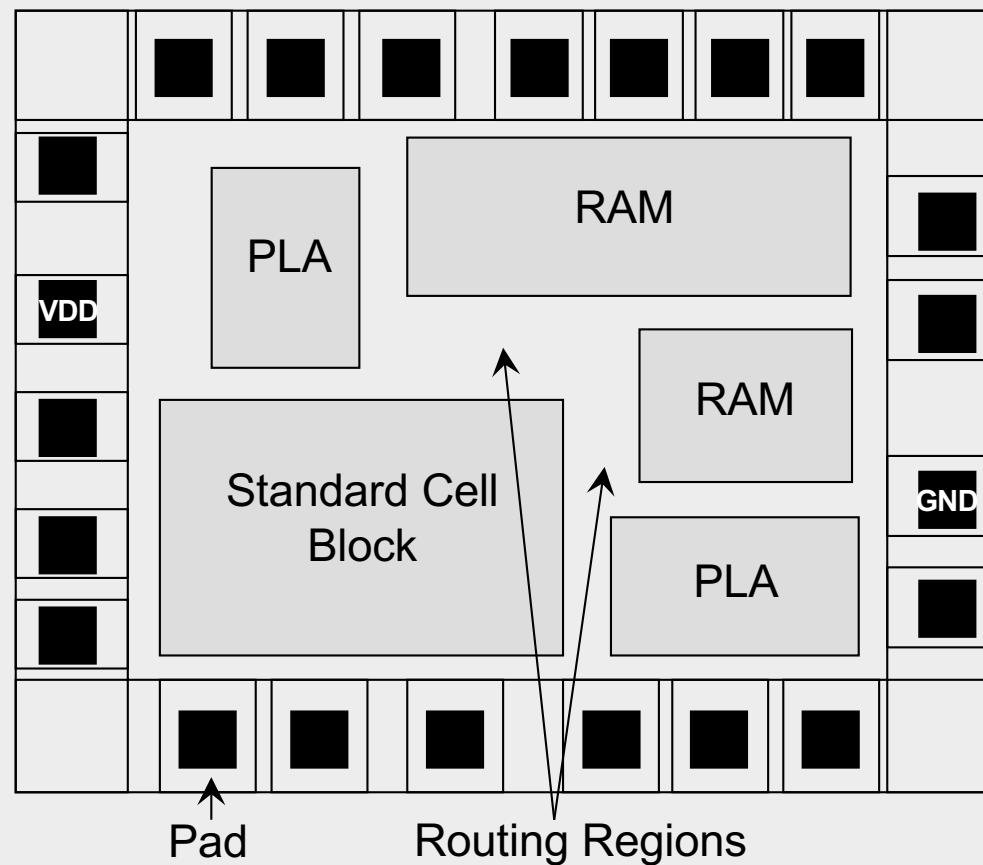
Standard cell layout with  
a feedthrough cell



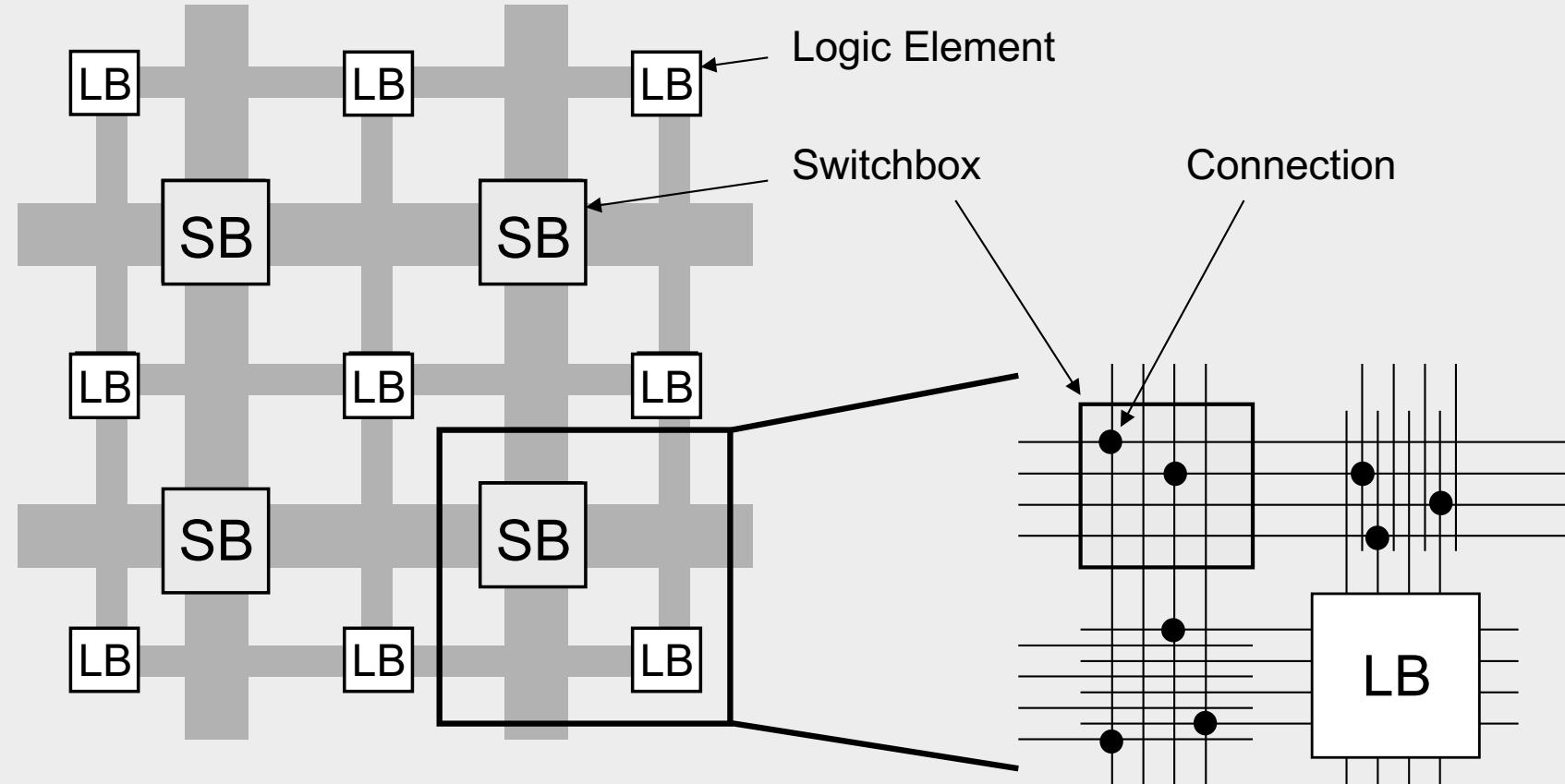
Standard cell layout using  
over-the-cell (OTC) routing



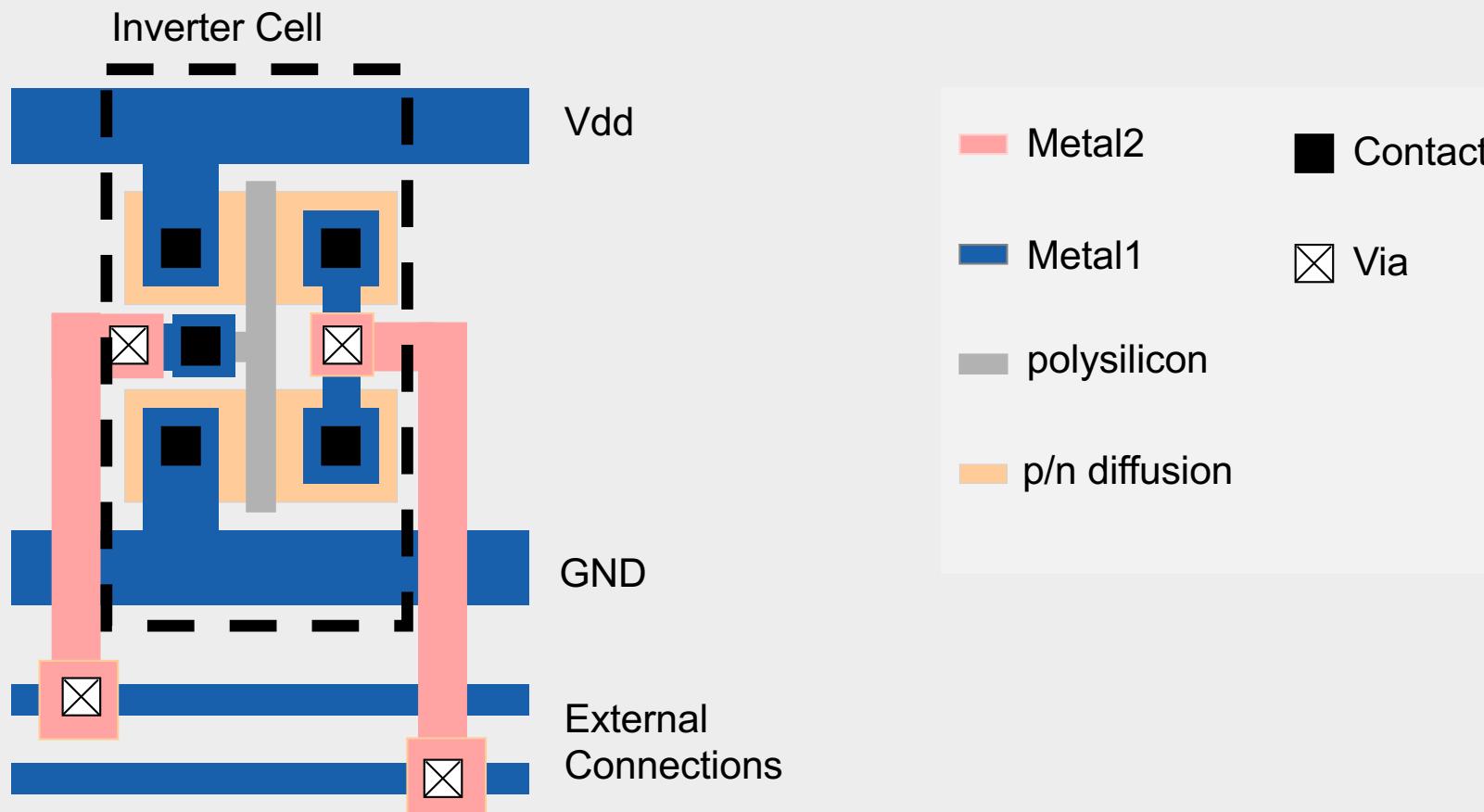
## Layout with macro cells



Field-programmable gate array (FPGA)



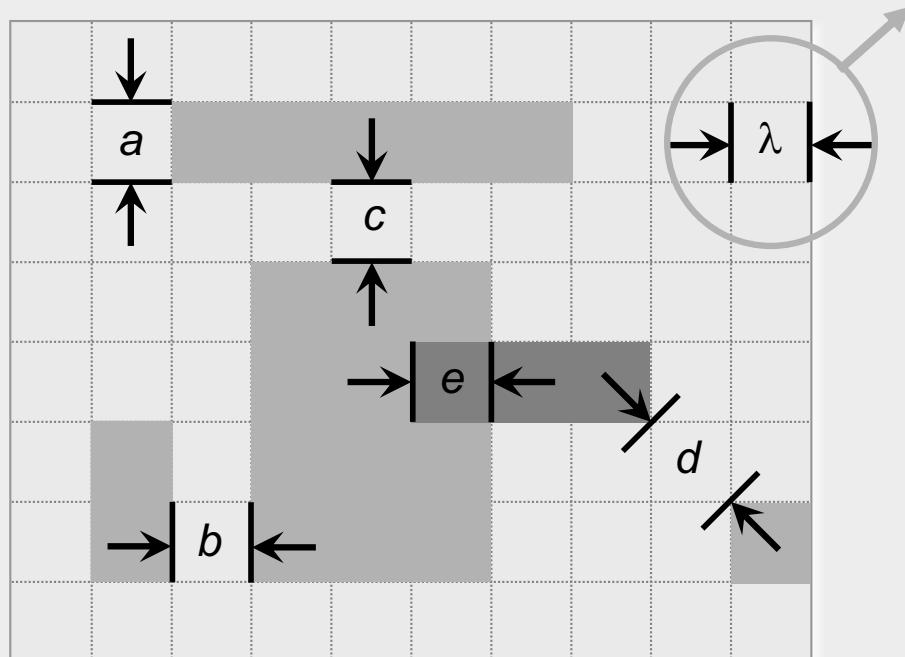
Layout layers of an inverter cell  
with external connections



### Categories of design rules

- **Size rules**, such as *minimum width*: The dimensions of any component (shape), e.g., length of a boundary edge or area of the shape, cannot be smaller than given minimum values. These values vary across different metal layers.
- **Separation rules**, such as *minimum separation*: Two shapes, either on the same layer or on adjacent layers, must be a minimum (rectilinear or Euclidean diagonal) distance apart.
- **Overlap rules**, such as *minimum overlap*: Two connected shapes on adjacent layers must have a certain amount of overlap due to inaccuracy of mask alignment to previously-made patterns on the wafer.

## Categories of design rules



$\lambda$ : smallest meaningful technology-dependent unit of length

Minimum Width: **a**

Minimum Separation: **b**, **c**, **d**

Minimum Overlap: **e**

## 1.5 Physical Design Optimizations

### Types of constraints

- **Technology constraints** enable fabrication for a specific technology node and are derived from technology restrictions. Examples include minimum layout widths and spacing values between layout shapes.
- **Electrical constraints** ensure the desired electrical behavior of the design. Examples include meeting maximum timing constraints for signal delay and staying below maximum coupling capacitances.
- **Geometry (design methodology) constraints** are introduced to reduce the overall complexity of the design process. Examples include the use of preferred wiring directions during routing, and the placement of standard cells in rows.

### Runtime complexity

- **Runtime complexity:** the time required by the algorithm to complete as a function of some natural measure of the problem size, allows comparing the scalability of various algorithms
- Complexity is represented in an asymptotic sense, with respect to the input size  $n$ , using **big-Oh notation** or  $O(\dots)$
- Runtime  $t(n)$  is order  $f(n)$ , written as  $t(n) = O(f(n))$  when  $\lim_{n \rightarrow \infty} \left| \frac{t(n)}{f(n)} \right| = k$  where  $k$  is a real number
- Example:  $t(n) = 7n! + n^2 + 100$ , then  $t(n) = O(n!)$  because  $n!$  is the fastest growing term as  $n \rightarrow \infty$ .

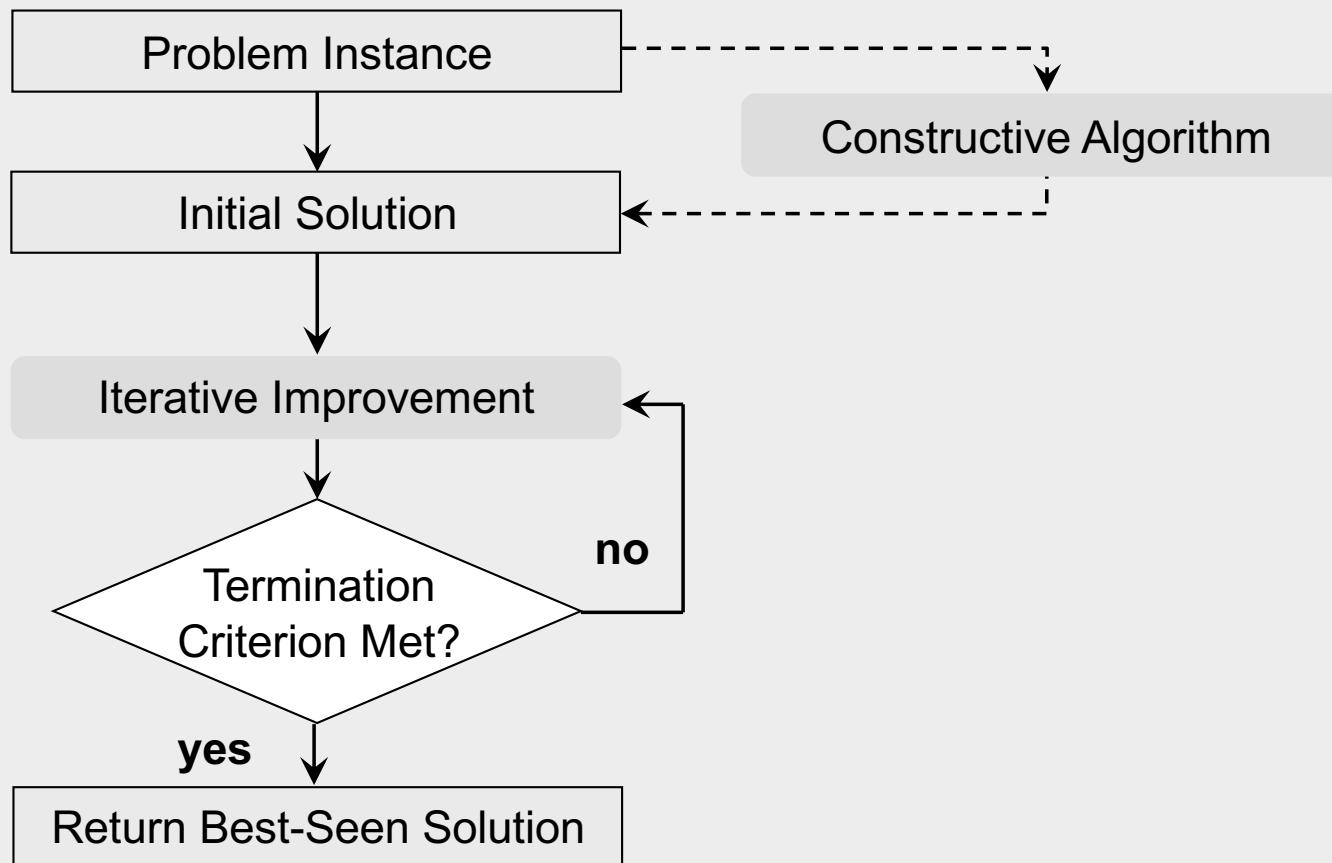
### Runtime complexity

- Example: Exhaustively Enumerating All Placement Possibilities
    - Given:  $n$  cells
    - Task: find a single-row placement of  $n$  cells with minimum total wirelength by using exhaustive enumeration.
    - Solution: The solution space consists of  $n!$  placement options. If generating and evaluating the wirelength of each possible placement solution takes 1  $\mu\text{s}$  and  $n = 20$ , the total time needed to find an optimal solution would be 77,147 years!
  - A number of physical design problems have best-known algorithm complexities that grow exponentially with  $n$ , e.g.,  $O(n!)$ ,  $O(n^n)$ , and  $O(2^n)$ .
  - Many of these problems are **NP-hard** (NP: non-deterministic polynomial time)
    - No known algorithms can ensure, in a time-efficient manner, globally optimal solution
- ⇒ **Heuristic algorithms** are used to find near-optimal solutions

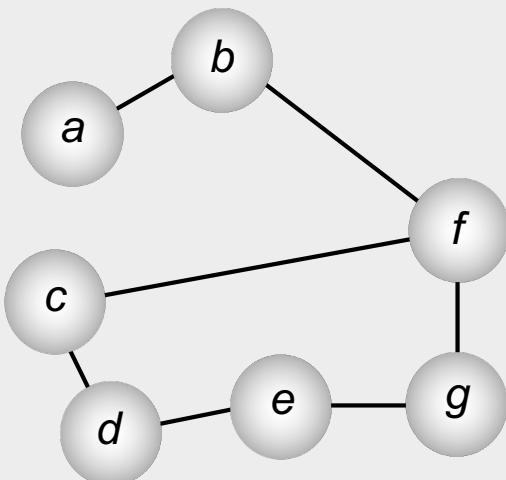
### Heuristic algorithms

- **Deterministic:** All decisions made by the algorithm are repeatable, i.e., not random. One example of a deterministic heuristic is Dijkstra's shortest path algorithm.
- **Stochastic:** Some decisions made by the algorithm are made randomly, e.g., using a pseudo-random number generator. Thus, two independent runs of the algorithm will produce two different solutions with high probability. One example of a stochastic algorithm is simulated annealing.
- In terms of structure, a heuristic algorithm can be
  - **Constructive:** The heuristic starts with an initial, incomplete (partial) solution and adds components until a complete solution is obtained.
  - **Iterative:** The heuristic starts with a complete solution and repeatedly improves the current solution until a preset termination criterion is reached.

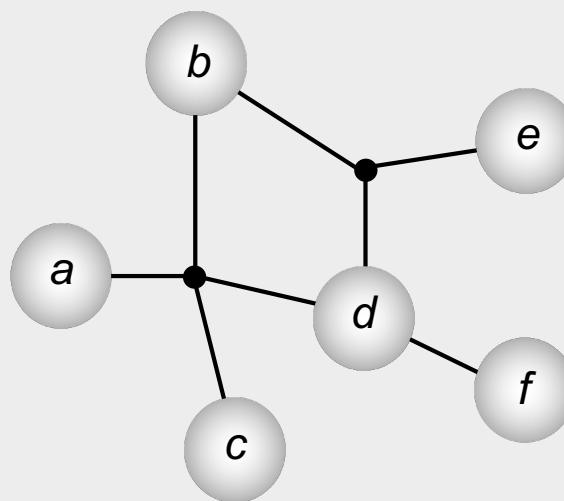
## Heuristic algorithms



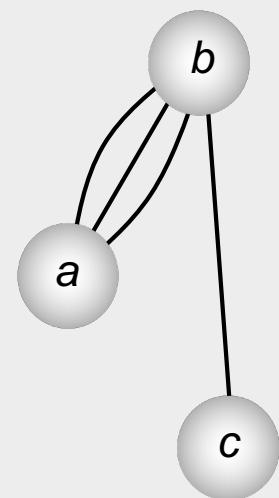
Graph



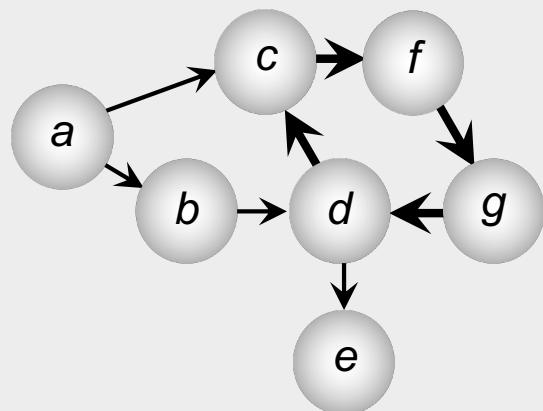
Hypergraph



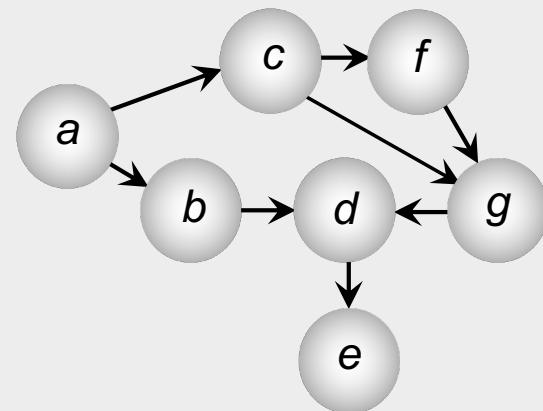
Multigraph



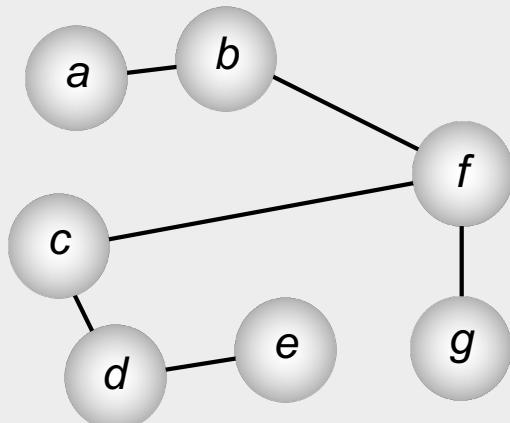
Directed graphs with cycles



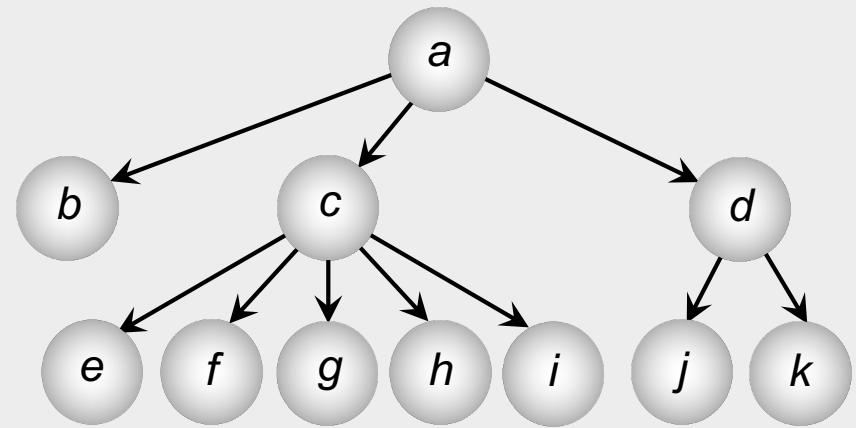
Directed acyclic graph



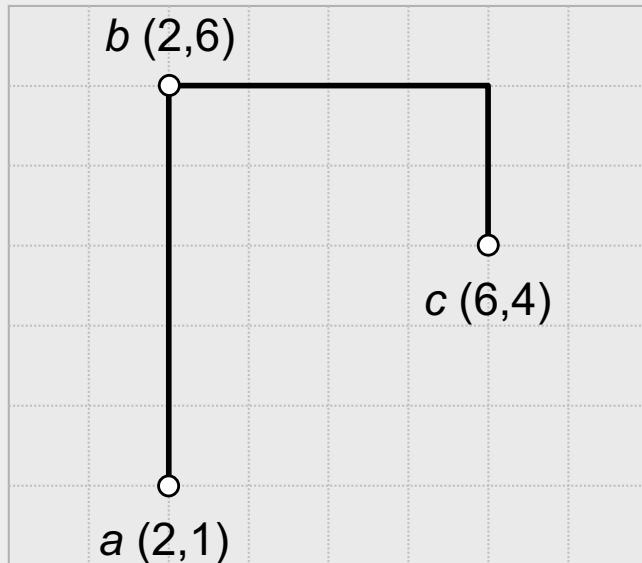
Undirected graph with maximum node degree 3



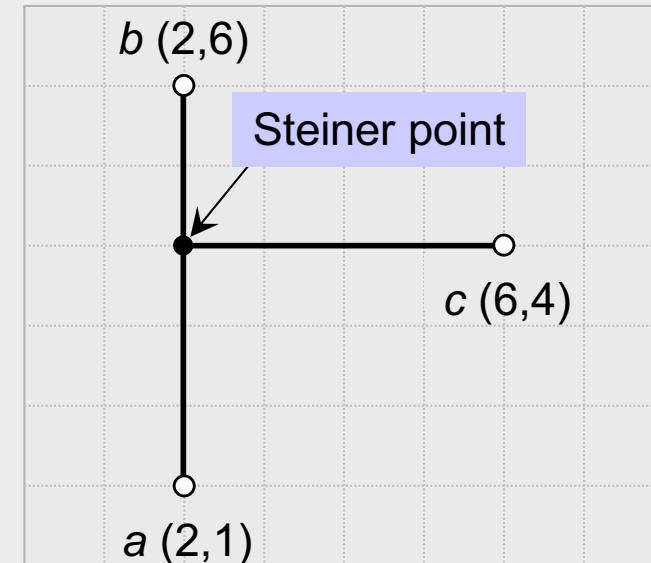
Directed tree



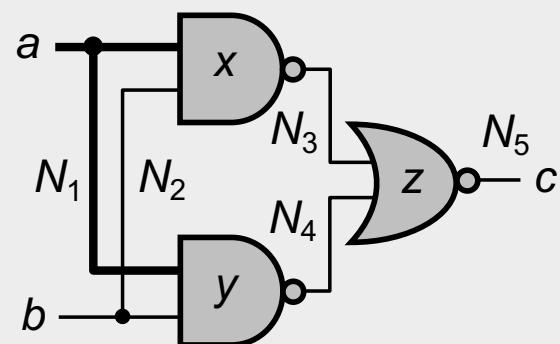
Rectilinear minimum spanning tree (RMST)



Rectilinear Steiner minimum tree (RSMT)



## Netlist



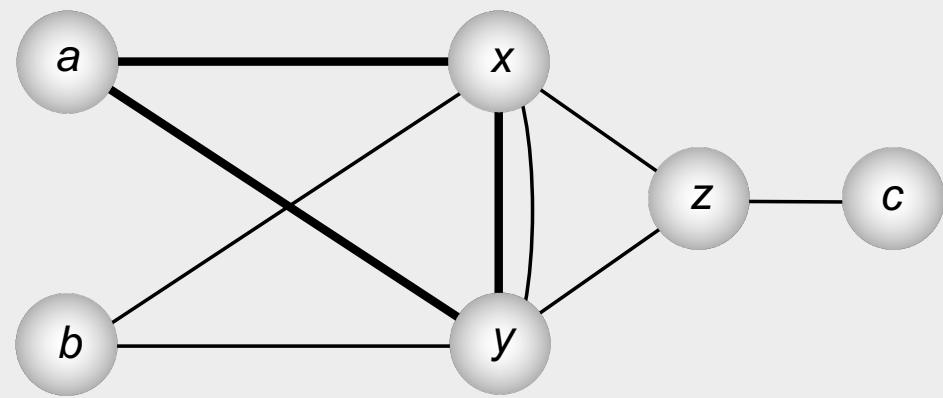
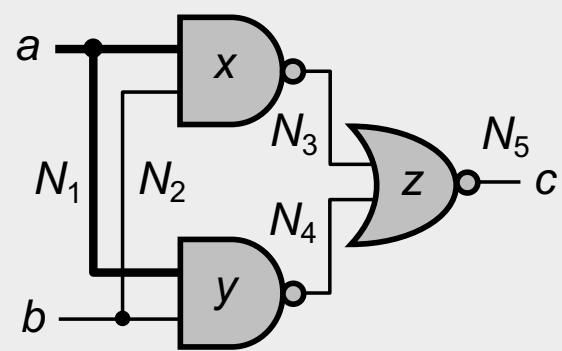
(**a**:  $N_1$ )  
(**b**:  $N_2$ )  
(**c**:  $N_5$ )  
(**x**:  $IN1\ N_1, IN2\ N_2, OUT\ N_3$ )  
(**y**:  $IN1\ N_1, IN2\ N_2, OUT\ N_4$ )  
(**z**:  $IN1\ N_3, IN2\ N_4, OUT\ N_5$ )

( $N_1$ : **a**,  $x.IN1$ ,  $y.IN1$ )  
( $N_2$ : **b**,  $x.IN2$ ,  $y.IN2$ )  
( $N_3$ :  $x.OUT$ ,  $z.IN1$ )  
( $N_4$ :  $y.OUT$ ,  $z.IN2$ )  
( $N_5$ :  $z.OUT$ , **c**)

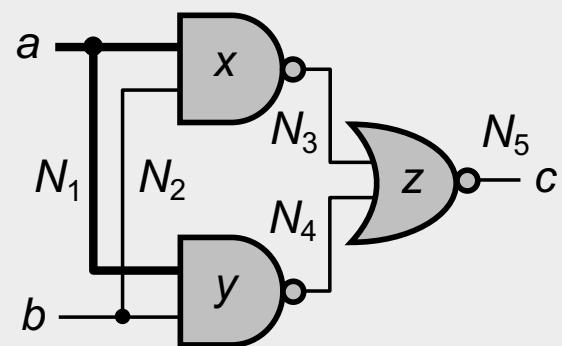
## Pin-Oriented Netlist

## Net-Oriented Netlist

## Connectivity graph



## Connectivity matrix



	$a$	$b$	$x$	$y$	$z$	$c$
$a$	0	0	1	1	0	0
$b$	0	0	1	1	0	0
$x$	1	1	0	2	1	0
$y$	1	1	2	0	1	0
$z$	0	0	1	1	0	1
$c$	0	0	0	0	1	0

Distance metric between two points  $P_1(x_1, y_1)$  and  $P_2(x_2, y_2)$

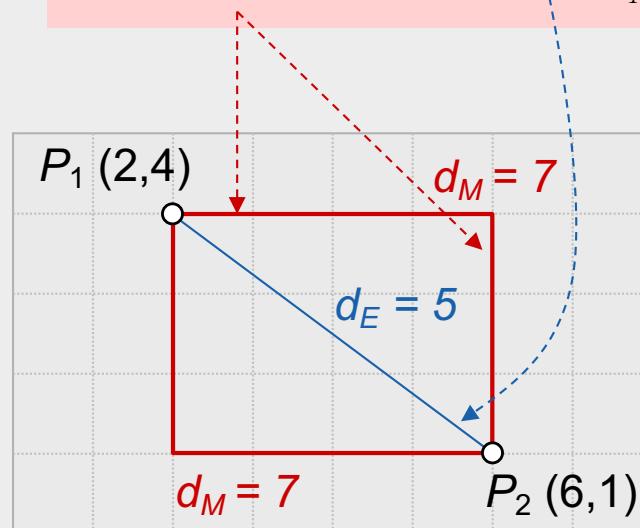
$$d = \sqrt[n]{|x_2 - x_1|^n + |y_2 - y_1|^n}$$

with  $n = 2$ : Euclidean distance

$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$n = 1$ : Manhattan distance

$$d_M(P_1, P_2) = |x_2 - x_1| + |y_2 - y_1|$$



## Summary of Lecture 1

- IC production experienced huge growth since the 1960s
  - Exponential decrease in transistor size, cost per transistor, power per transistor, etc
- IC design is impossible without simplification and automation
  - Row-based standard-cell layout with design rules
  - Traditionally, each step in the VLSI design flow has been automated separately by software (CAD) tools
- Software tools use sophisticated algorithms
  - Many problems in physical design are NP-hard – solved by heuristic algorithms that find near-optimal solutions
  - Deterministic versus stochastic algorithms
  - Constructive algorithms versus iterative improvement
  - Graph algorithms – deal with circuit connectivity
  - Computational geometry – deal with circuit layout