CS612

Algorithms for Electronic Design Automation

Background and Introduction

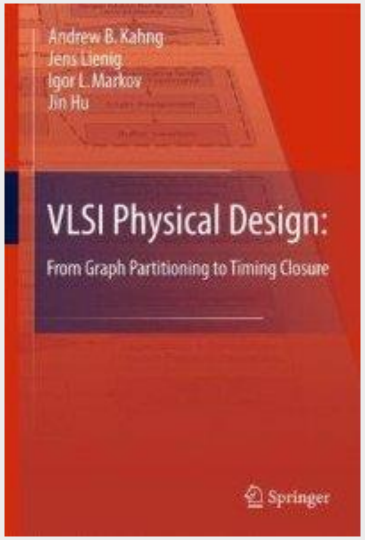
Mustafa Ozdal

SOME SLIDES ARE FROM THE BOOK:

VLSI Physical Design: From Graph Partitioning to Timing Closure

MODIFICATIONS WERE MADE ON THE ORIGINAL SLIDES

Chapter 1 – Introduction



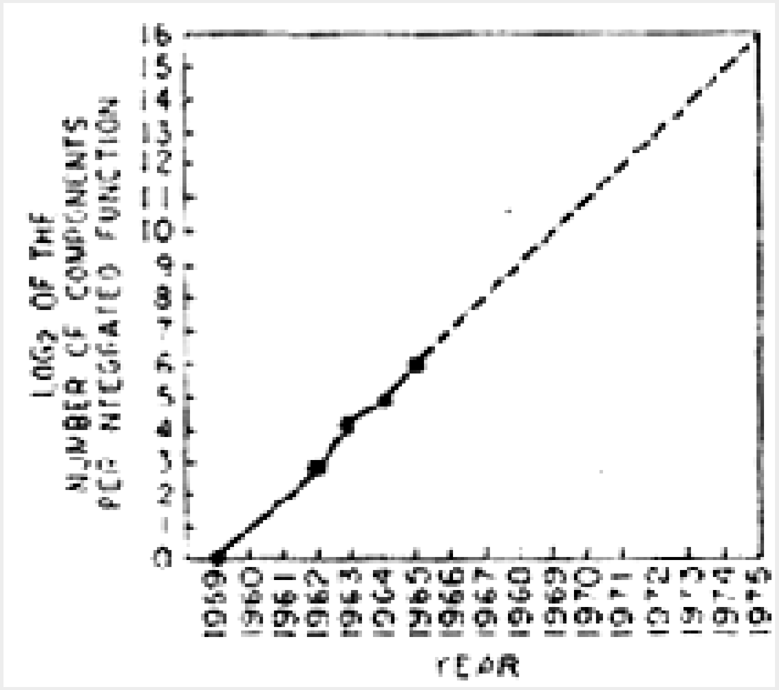
Original Authors:

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

1.1 Electronic Design Automation (EDA)

Moore's Law

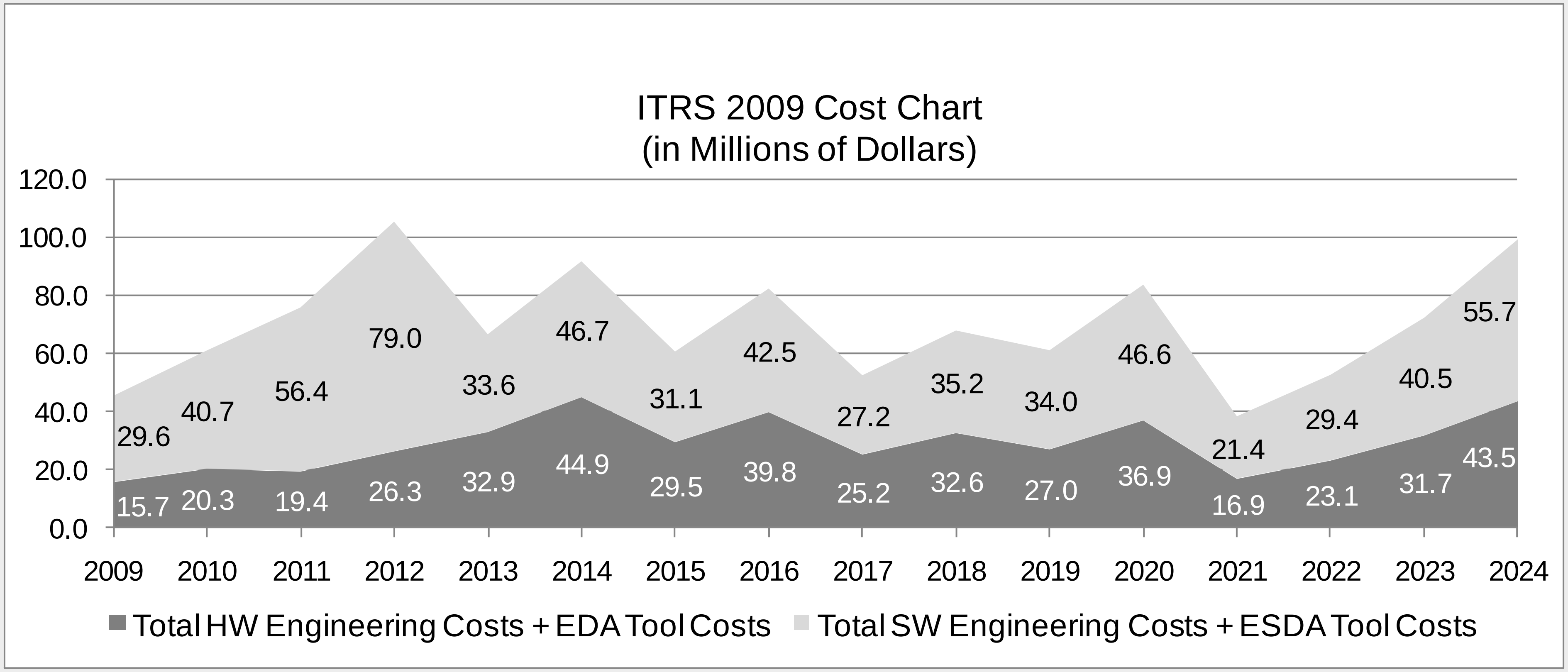
In 1965, Gordon Moore (Fairchild) stated that the number of transistors on an IC would double every year. 10 years later, he revised his statement, asserting that they double every 18 months. Since then, this "rule" has been famously known as Moore's Law.



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

1.1 Electronic Design Automation (EDA)

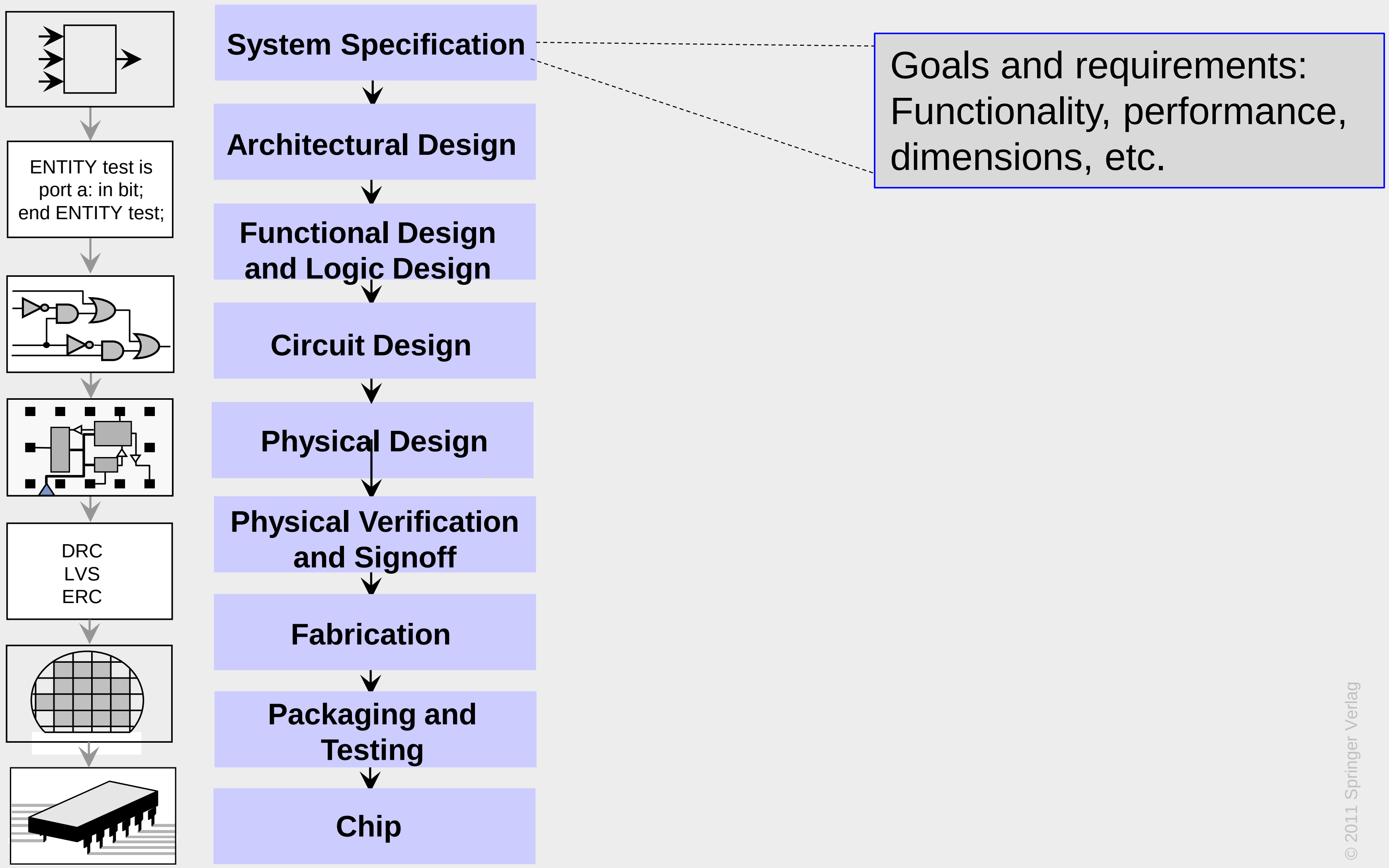
Without the design technology innovations between 1993 and 2007, the design cost of a chip would have been $1800M



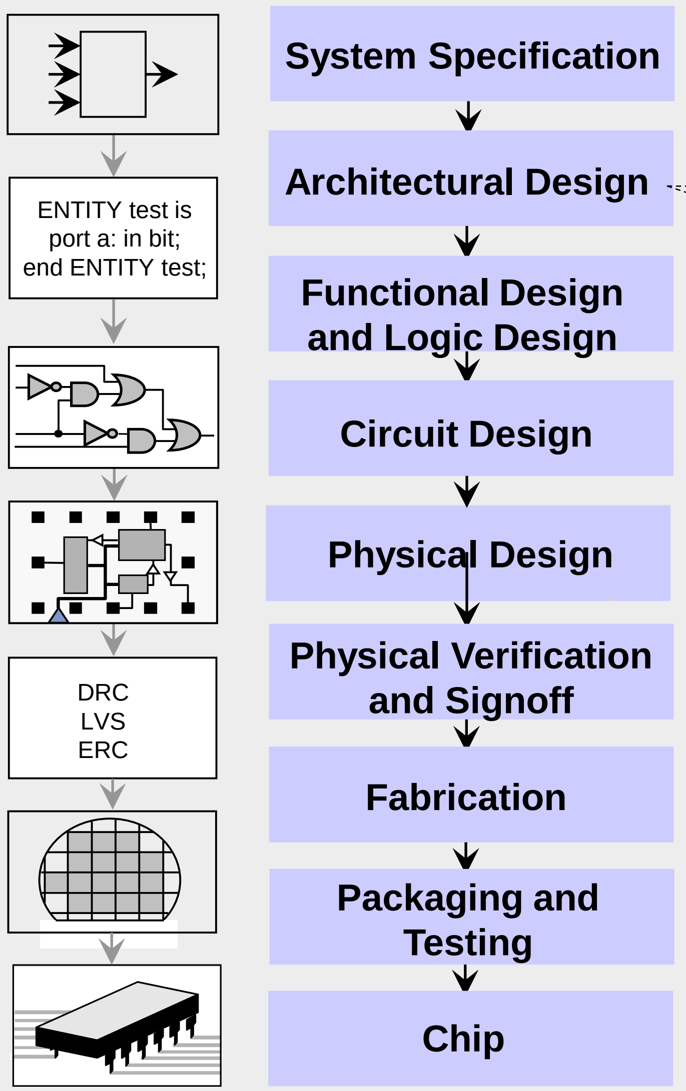
Electronic Design Automation (EDA)

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.

1.2 VLSI Design Flow

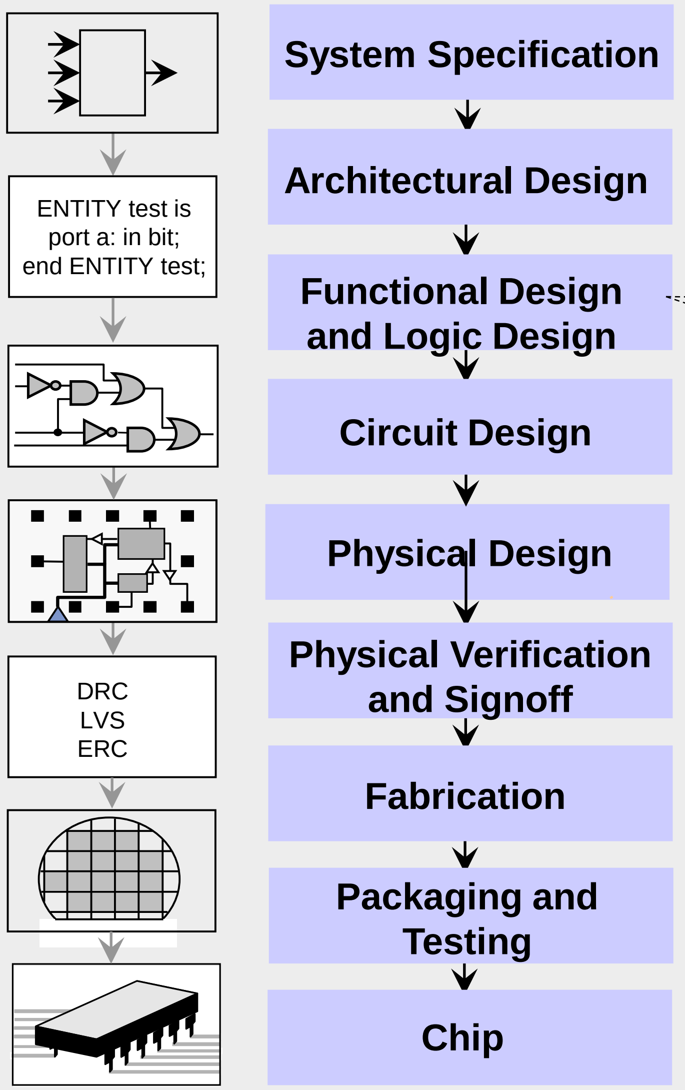

System Specification
Architectural Design
Functional Design and Logic Design
Circuit Design
Physical Design
Physical Verification and Signoff
Fabrication
Packaging and Testing
Chip
ENTITY test is port a: in bit; end ENTITY test;
DRC
LVS
ERC
Goals and requirements: Functionality, performance, dimensions, etc.

1.2 VLSI Design Flow



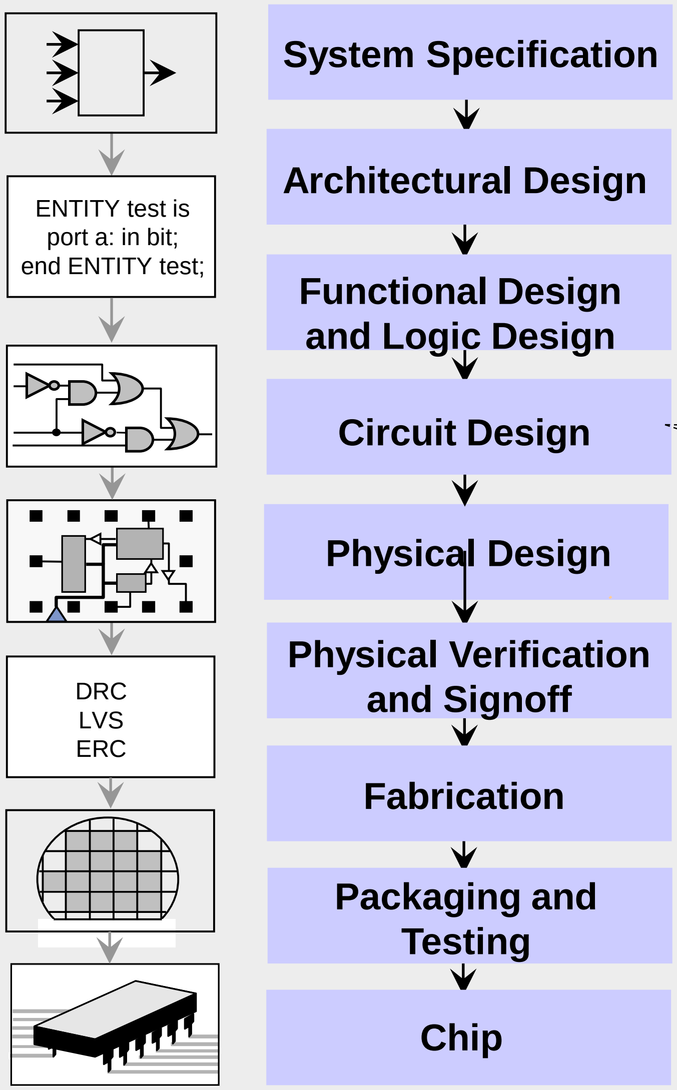
e.g. Instruction set, memory system, number of cores, communication protocols, …

1.2 VLSI Design Flow



Hardware description using RTL. Logic synthesis tools to generate netlists.

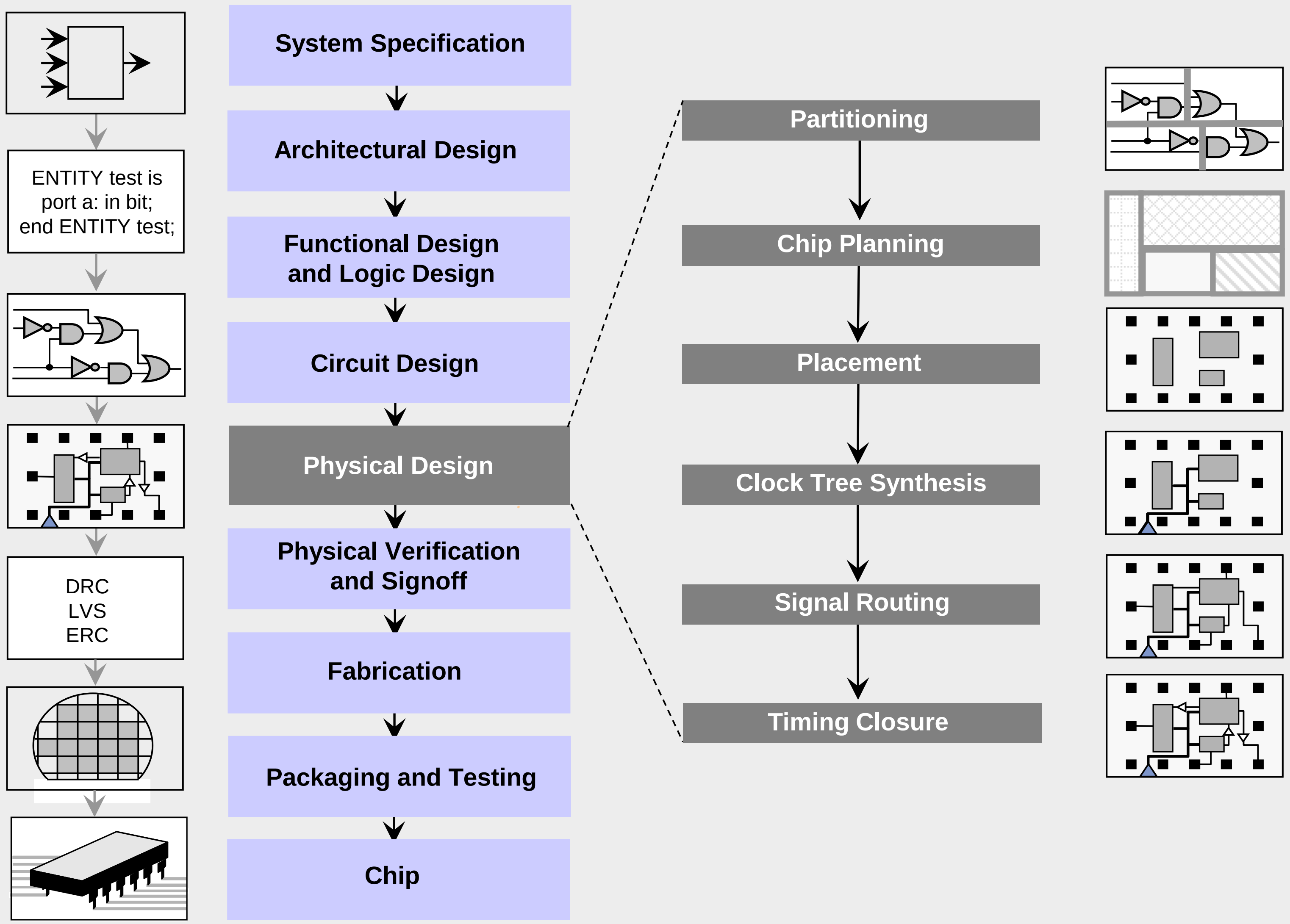
1.2 VLSI Design Flow



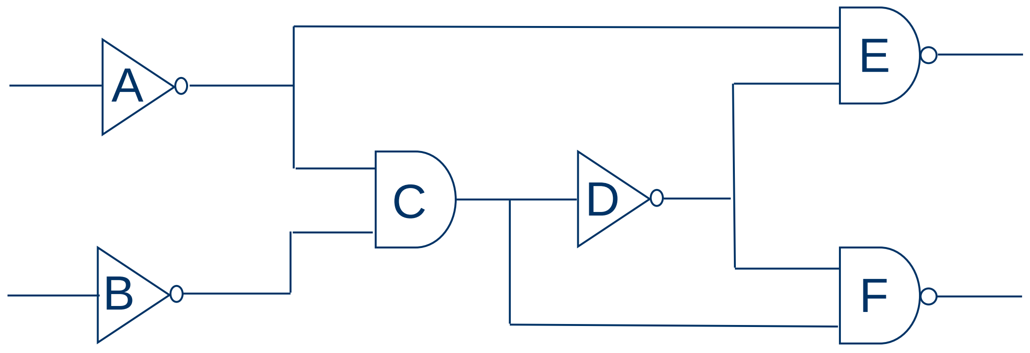
```
ENTITY test is
port a: in bit;
end ENTITY test;
```

Transistor-level design of low level elements, e.g. SRAMs, IO, critical blocks,…

1.2 VLSI Design Flow

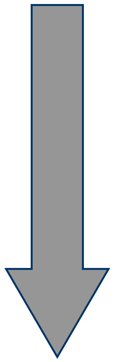


Physical Design

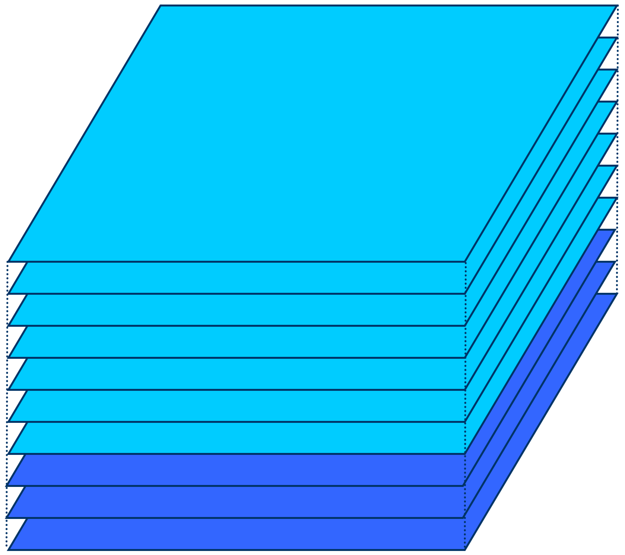


Netlist

Logic gates & connections

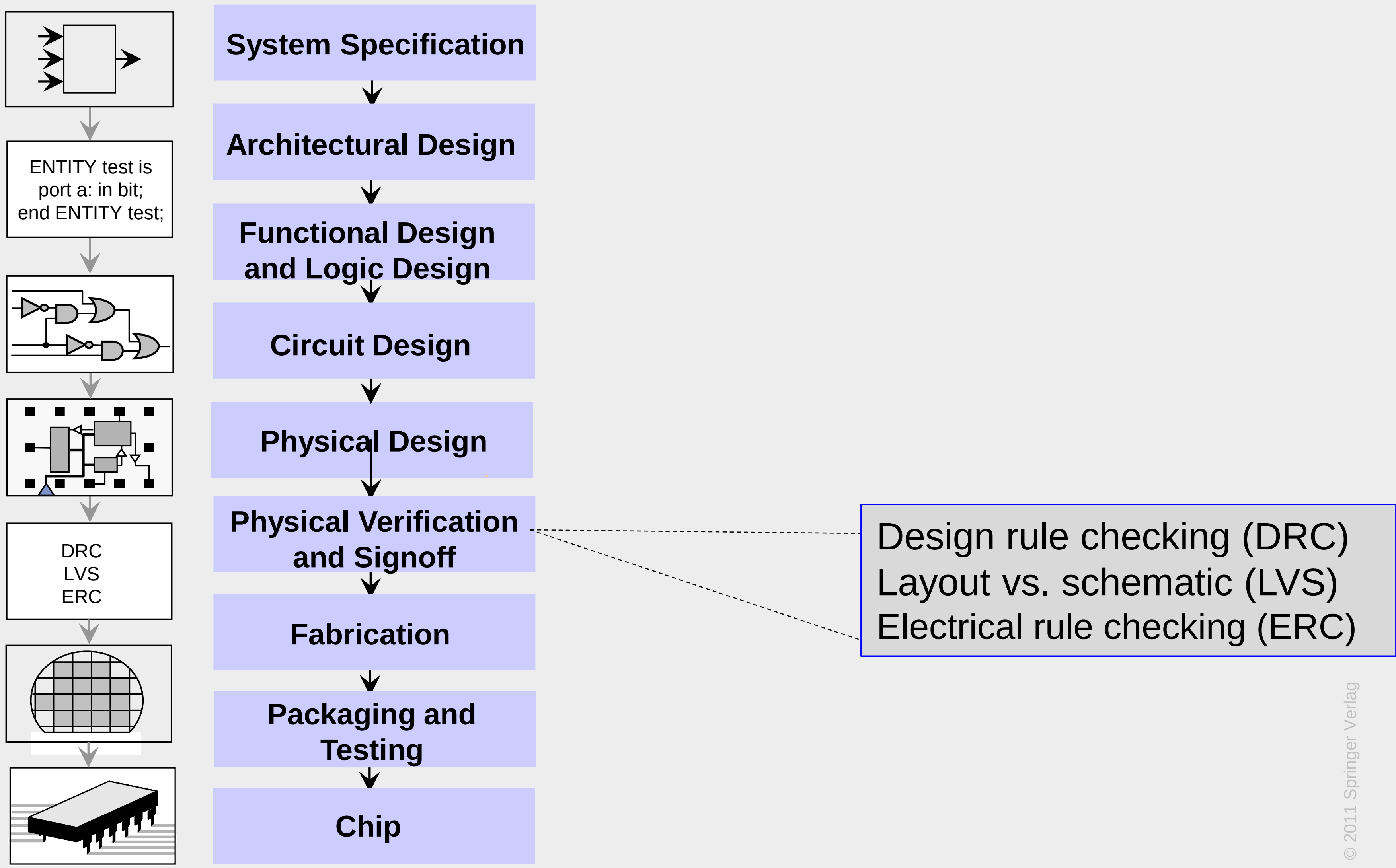


Shapes on chip layers

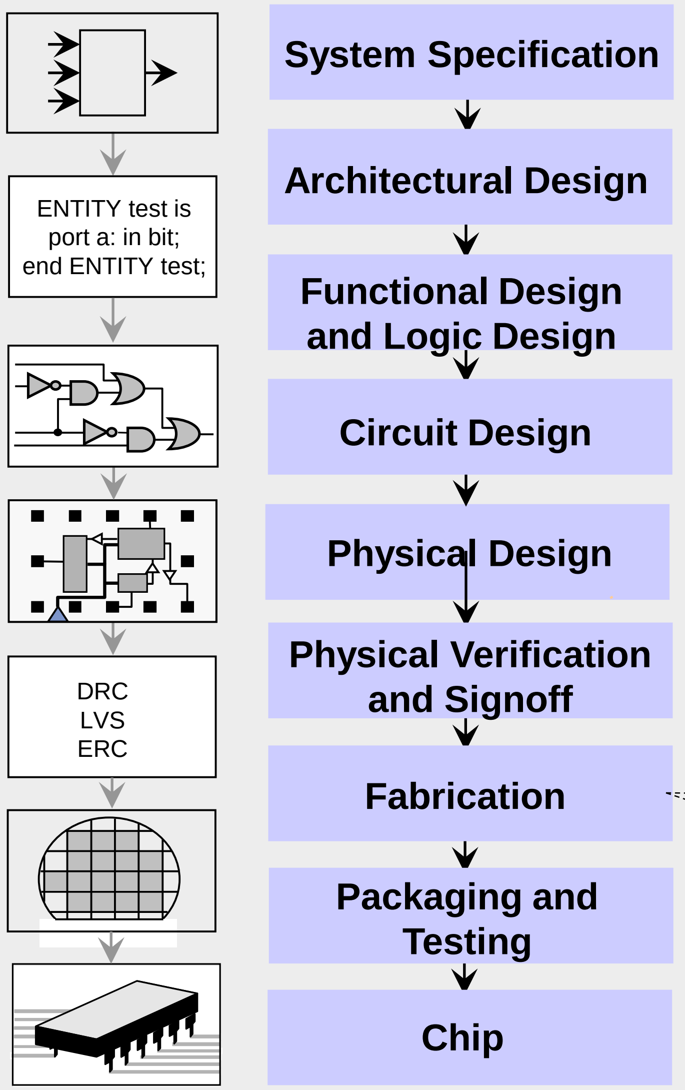


Chip Layers

1.2 VLSI Design Flow

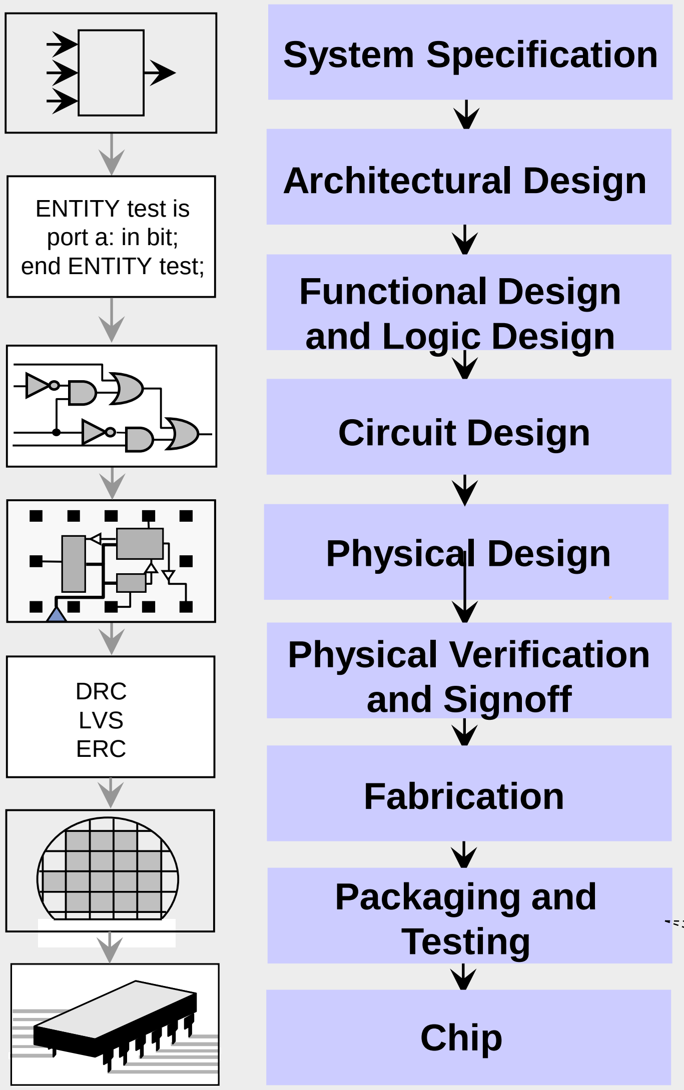


1.2 VLSI Design Flow



Layout sent to *a fab* (tapeout). Photomasks used to print the layout patterns onto layers.

1.2 VLSI Design Flow


```
ENTITY test is
port a: in bit;
end ENTITY test;
```

Die is positioned in a package. Pins connected to the package pins. Multiple chips can be integrated.

VLSI Design Styles

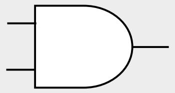
- Full-custom design
  - Direct transistor-level design
  - Max flexibility, few constraints
  - Potential to highly optimize, but high design cost
  - Critical and high-volume parts that will be replicated

- Semi-custom design
  - Cell-based: Standard and macro cells
    - Many pre-designed elements in the libraries
  - Array-based: Configure the pre-fabricated elements
    - e.g. FPGA

Cell Based Design

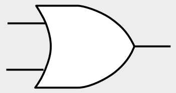
Common digital cells

AND



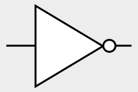
IN1	IN2	OUT
0	0	0
1	0	0
0	1	0
1	1	1

OR



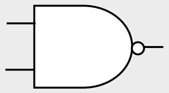
IN1	IN2	OUT
0	0	0
1	0	1
0	1	1
1	1	1

INV



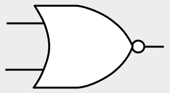
IN	OUT
0	1
1	0

NAND

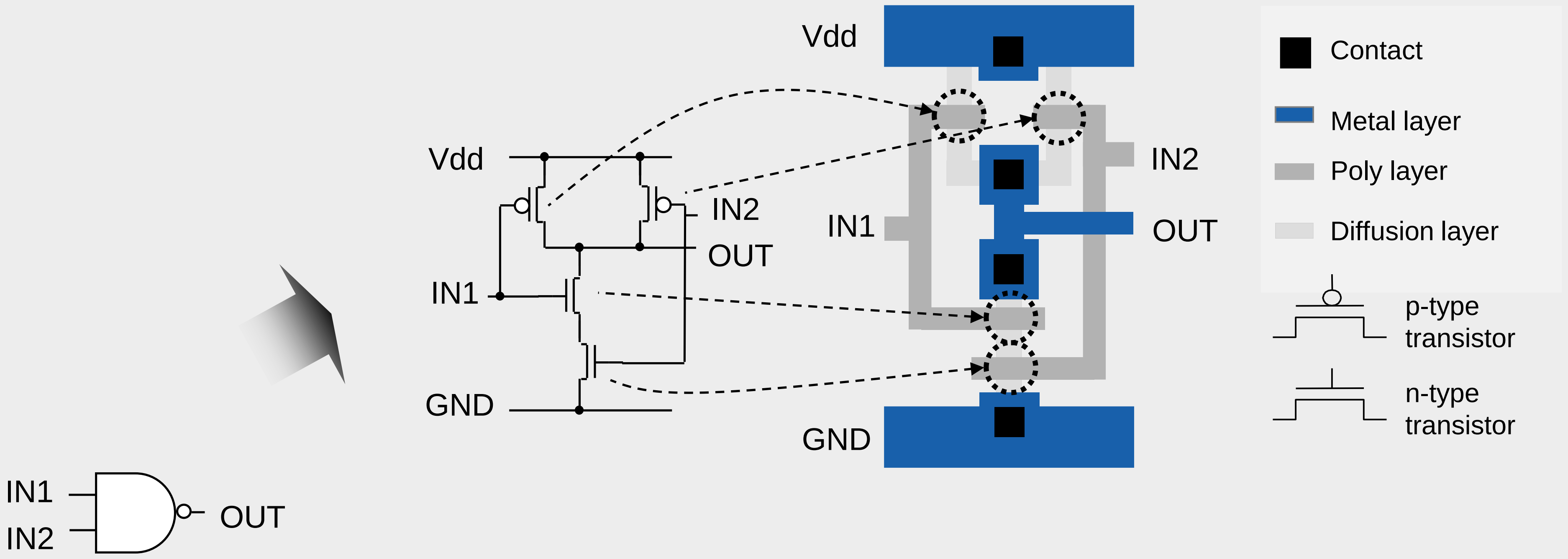


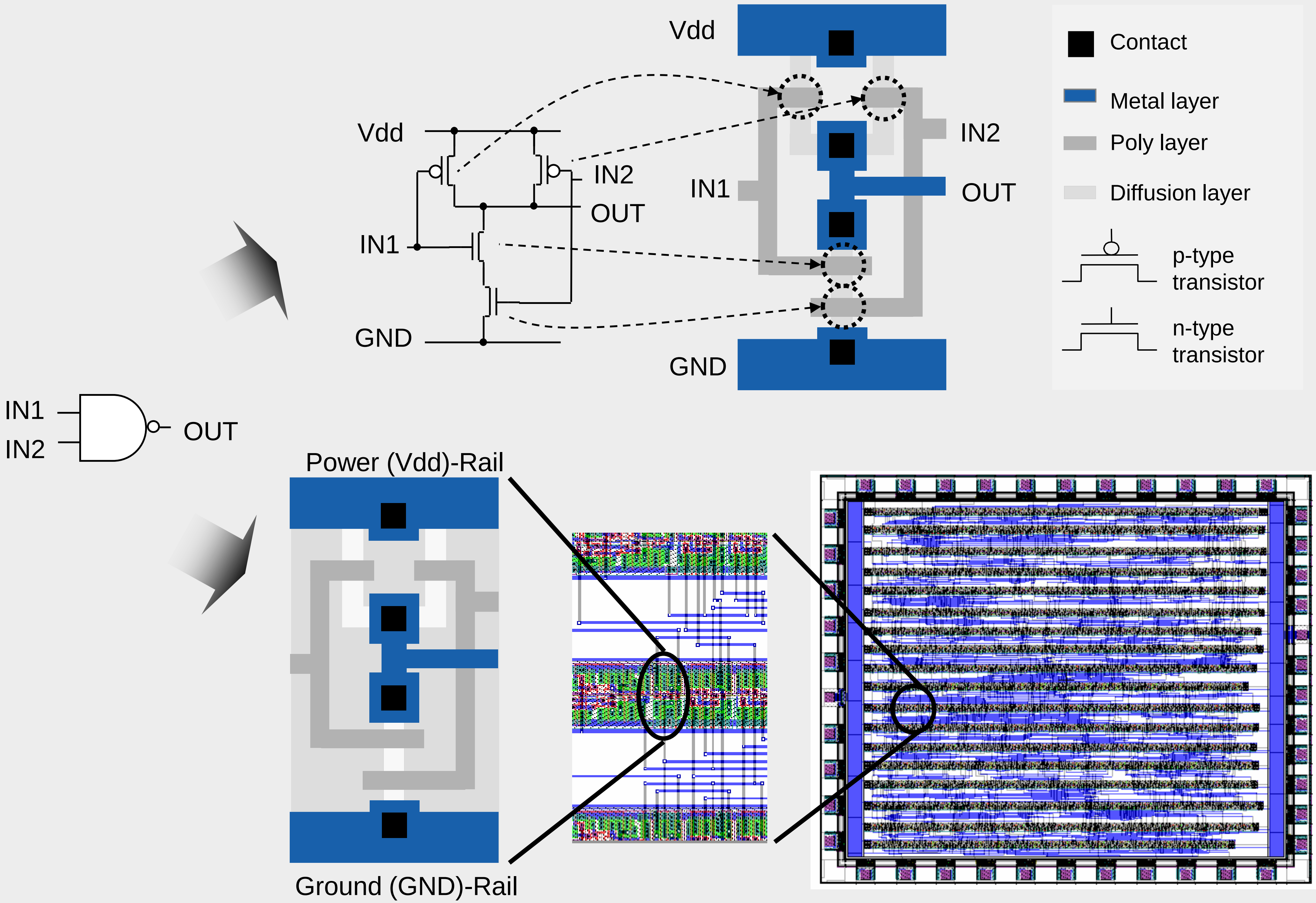
IN1	IN2	OUT
0	0	1
1	0	1
0	1	1
1	1	0

NOR

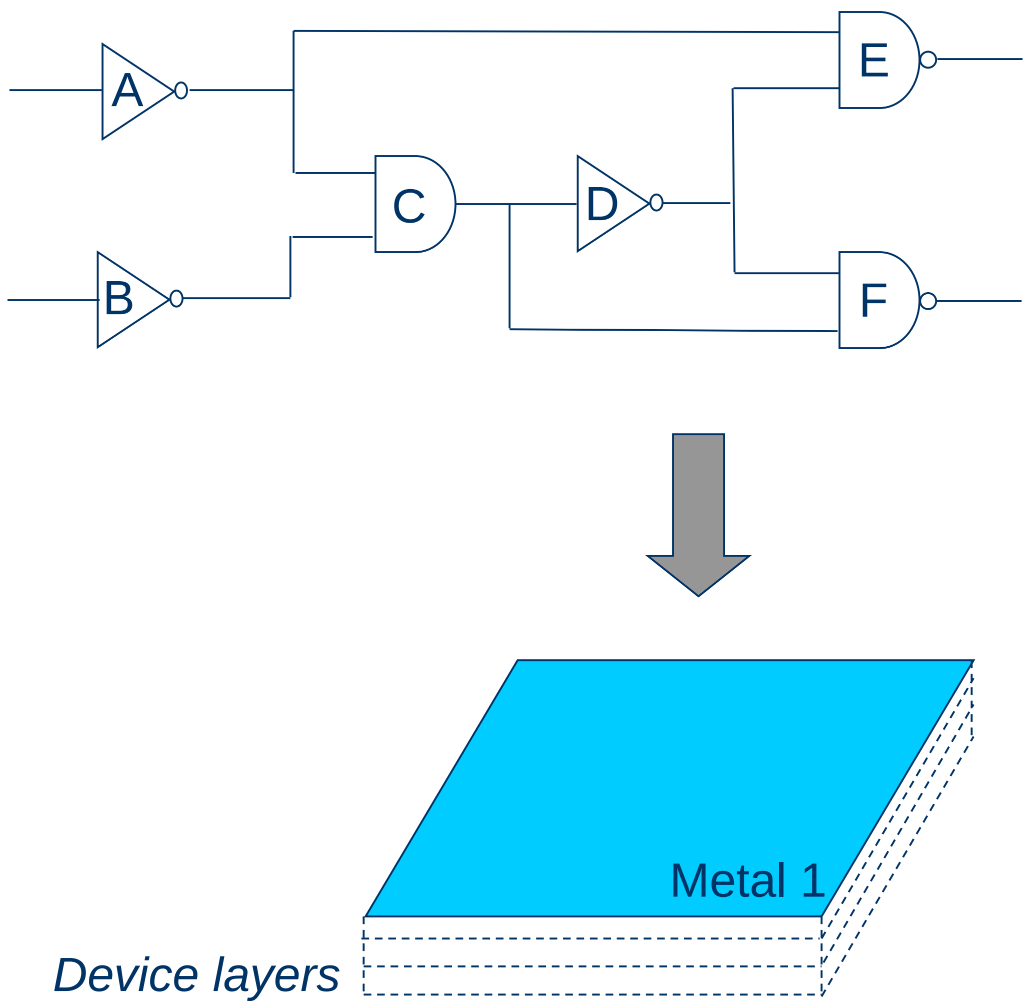


IN1	IN2	OUT
0	0	1
1	0	0
0	1	0
1	1	0


Vdd
IN2
OUT
IN1
GND
Vdd
IN1
IN2
OUT
GND
IN1
IN2
OUT
Contact
Metal layer
Poly layer
Diffusion layer
p-type transistor
n-type transistor


Vdd
IN2
OUT
IN1
GND
Vdd
IN1
IN2
OUT
GND
Contact
Metal layer
Poly layer
Diffusion layer
p-type transistor
n-type transistor
IN1
IN2
OUT
Power (Vdd)-Rail
Ground (GND)-Rail

Cell Placement



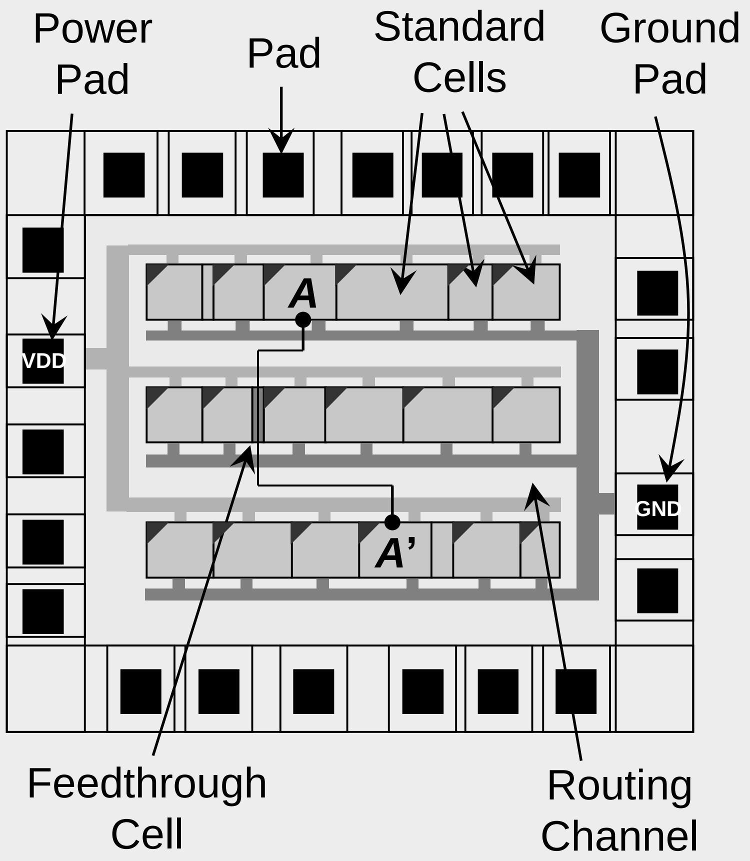
- Implementation on device layers available in cell library

  (e.g. AND, OR, NAND, NOR, INV, etc.)

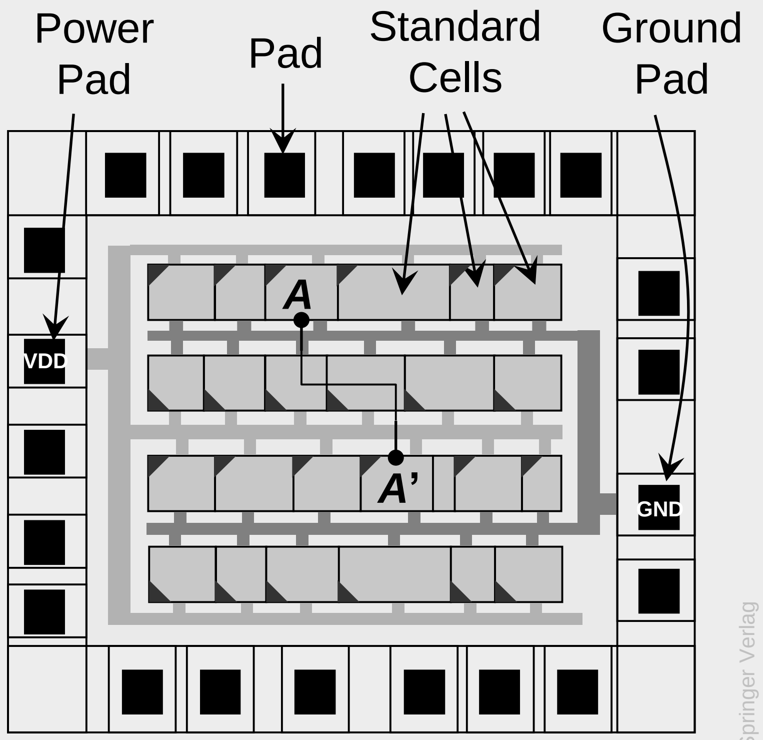
- Cells arranged in rows on metal1

1.3 VLSI Design Styles

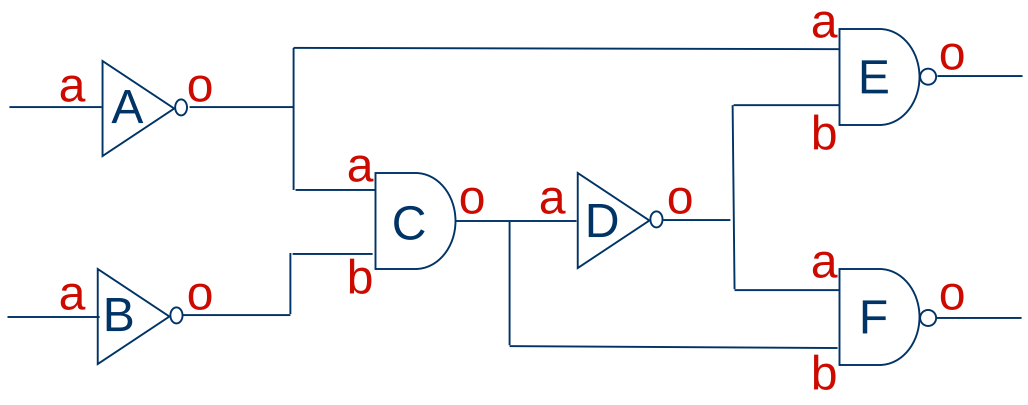
Standard cell layout with a feedthrough cell

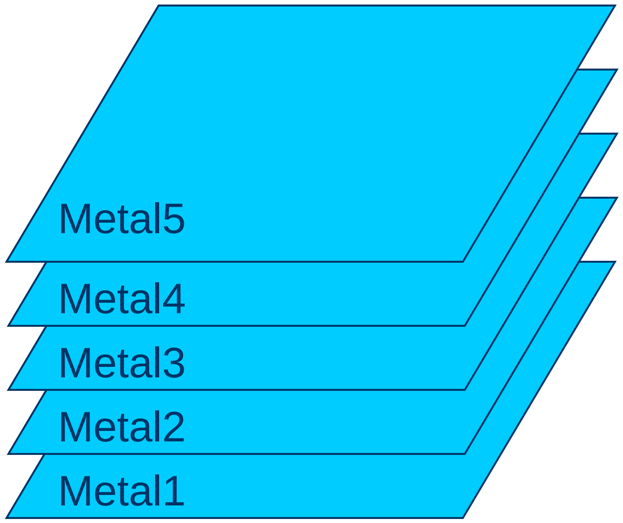


Standard cell layout using over-the-cell (OTC routing



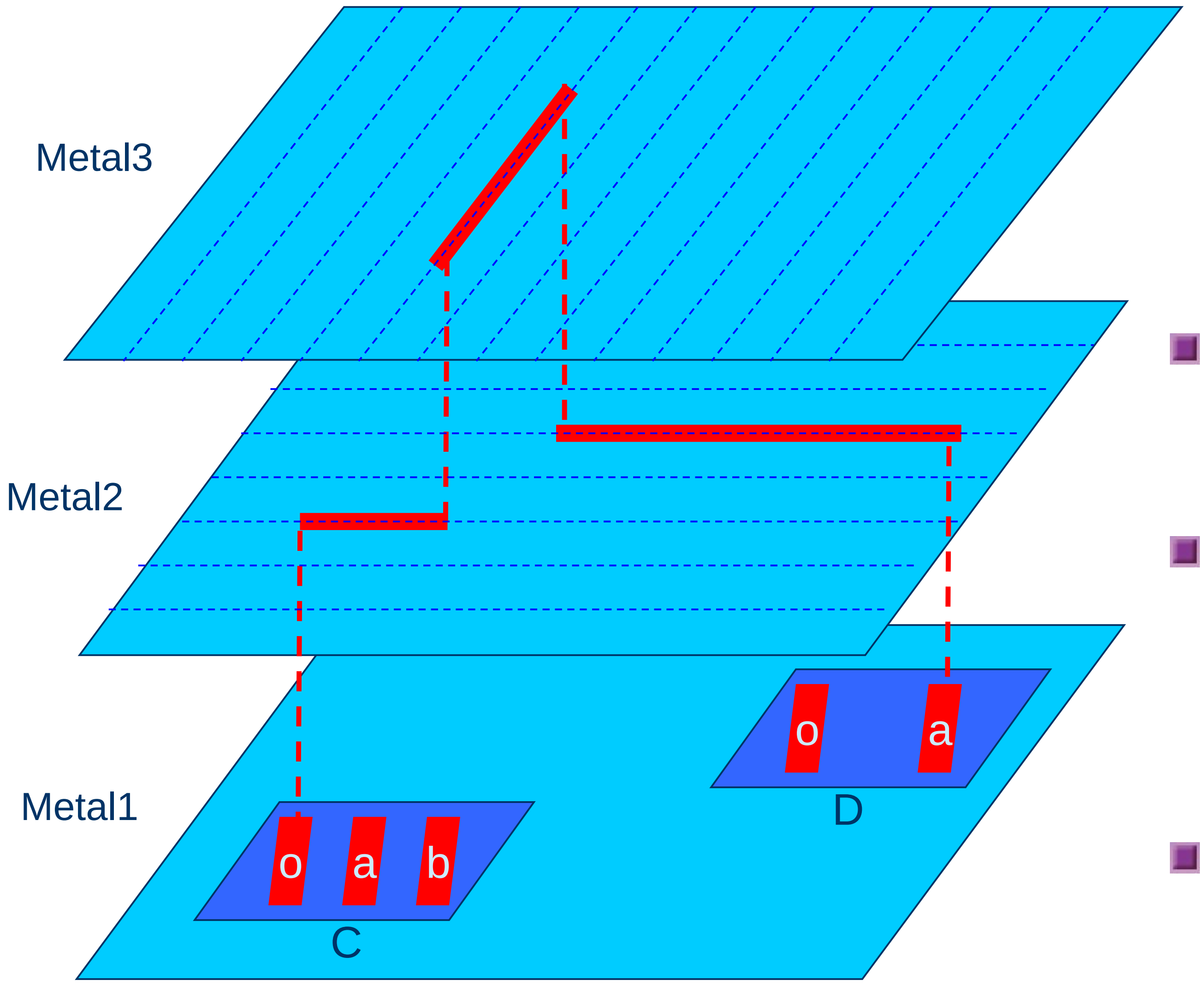
Over-the-cell routing

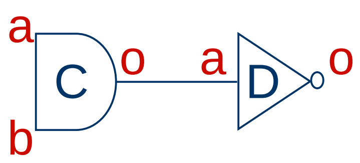




- All terminals on metal1
- Connect terminals using metal layers
- Each layer is either horizontal or vertical

Routing Between Two Pins





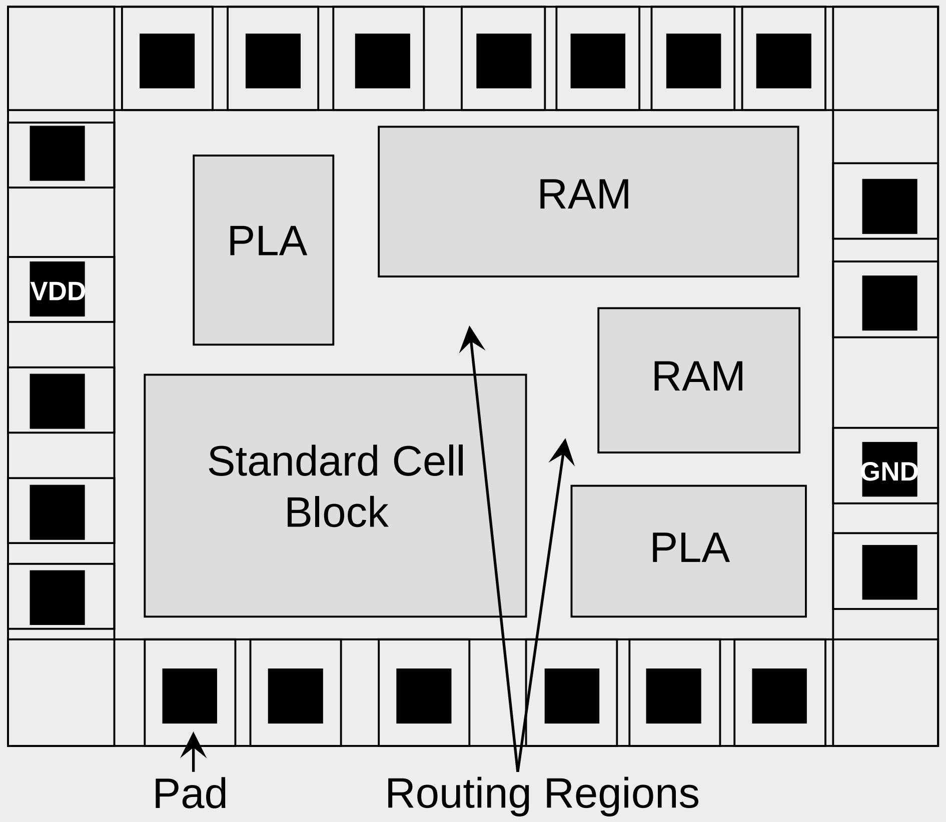
- Route from C/o to D/a
- Metal3: vertical

  Metal2: horizontal

  Metal1: escape-only
- Interlayer connection: “via”

Cell Based Design

Layout with macro cells



Why Cell Based Design?

- Easier to design, optimize and verify
  - Reuse of components
  - Performance of each cell pre-characterized
  - Transistor-level design constraints already handled
  - Complexities abstracted in the cell definition
  - Optimization easier

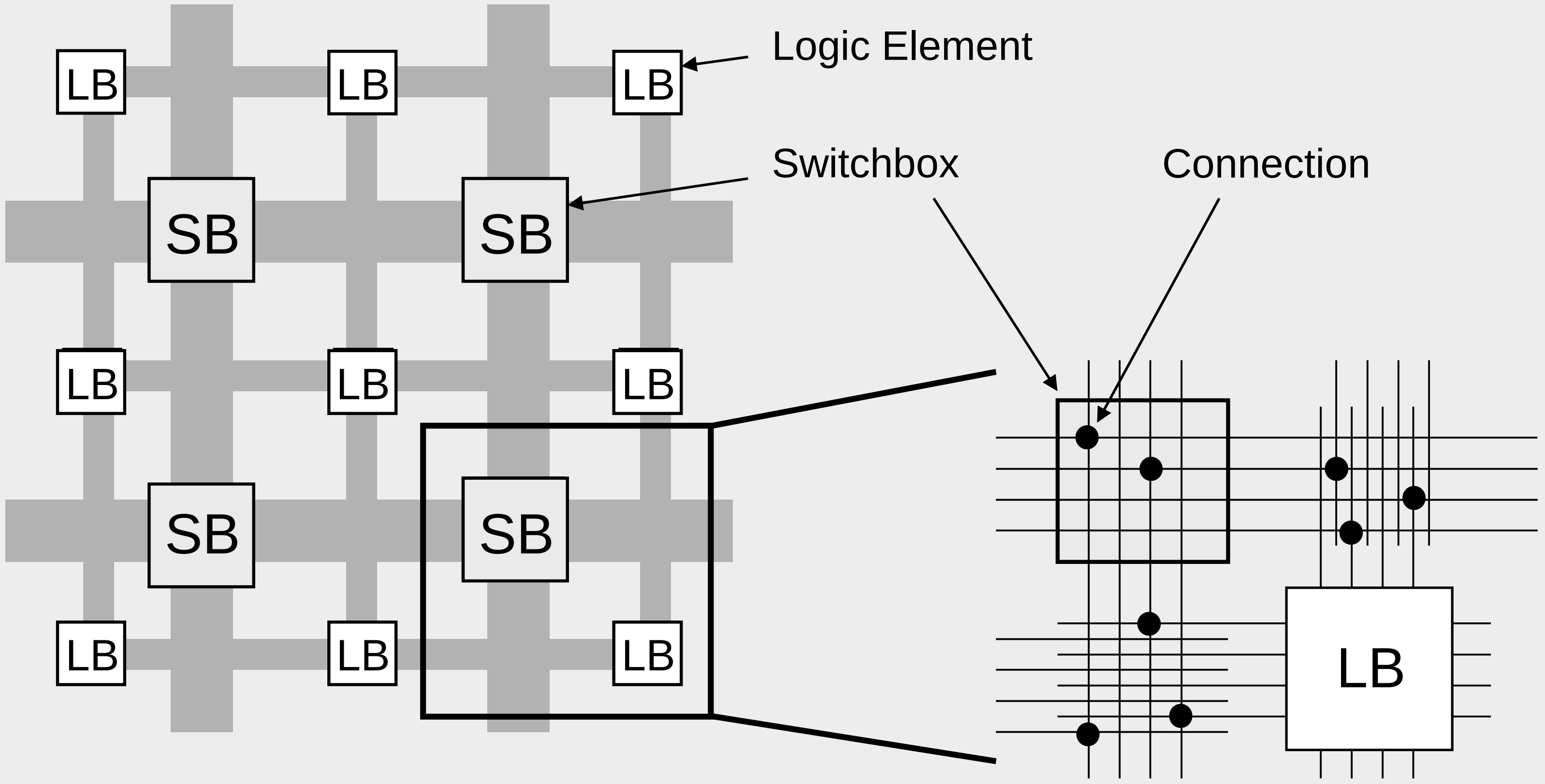
- Disadvantage: Less room for optimization
  - Cannot have a cell for every single complex logic function

Field Programmable Gate Array (FPGA)

After fabrication:

Each logic element can be configured to implement different functions

Each switchbox can be configured to change the connectivity



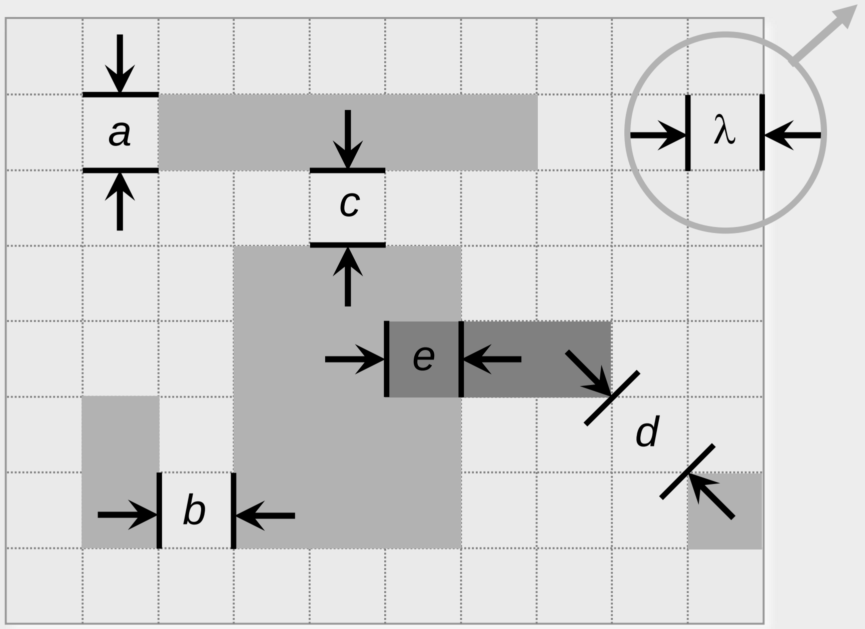
1.4. Design Rules

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.

1.4. Design Rules

Categories of design rules



λ: 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(…)
- Runtime $t(n)$ is order $f(n)$, written as $t(n) = O(f(n))$

  where k is a real number

$$\lim_{n\to\infty}\left|\frac{t(n)}{f(n)}\right| = k$$

- Example: $t(n) = 7n! + n^2 + 100$, then $t(n) = O(n!)$ because $n!$ is the fastest growing term as $n \to \infty$.

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 μs 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

$\Rightarrow$ Heuristic algorithms are used to find near-optimal solutions

1.6 Algorithms and Complexity

Heuristic algorithms

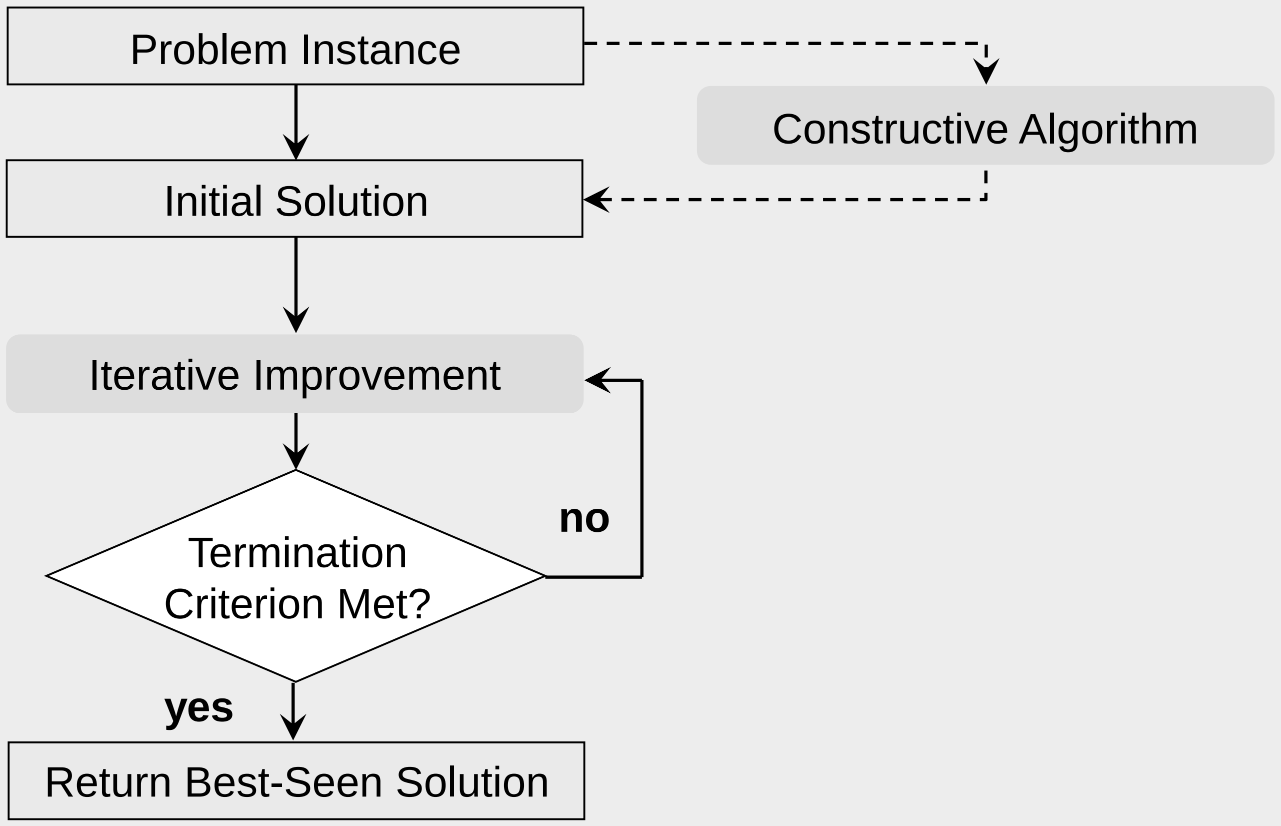
- **Deterministic**: All decisions made by the algorithm are repeatable, i.e., not random. One example of a deterministic heuristic is *A* shortest path algorithm*.

- **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.

1.6 Algorithms and Complexity

Heuristic algorithms

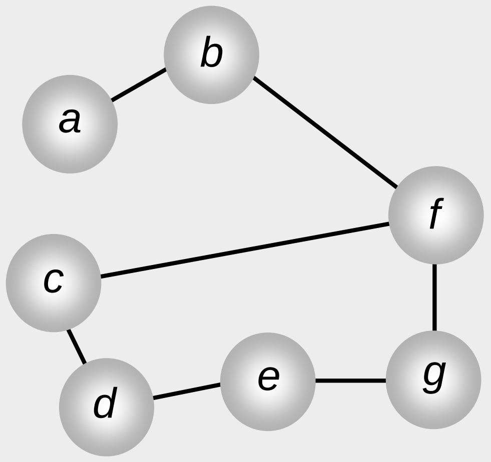


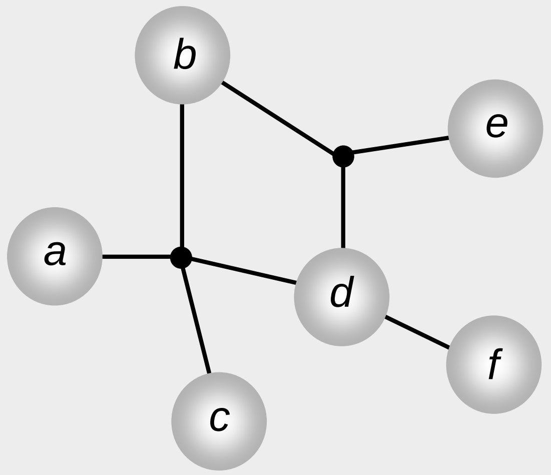
1.7 Graph Theory Terminology

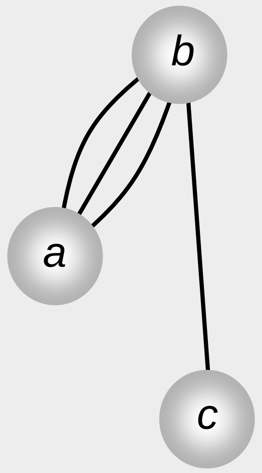
Graph

Hypergraph

Multigraph



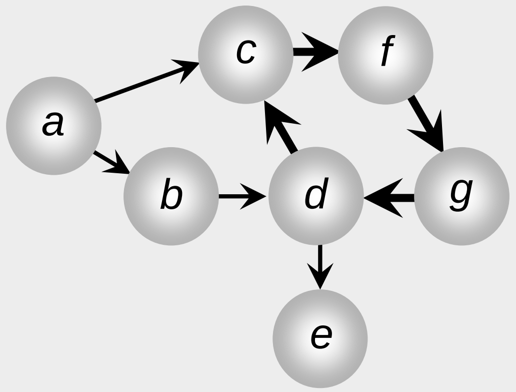


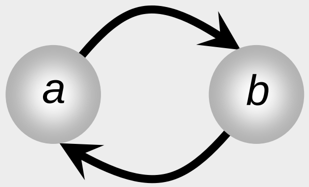


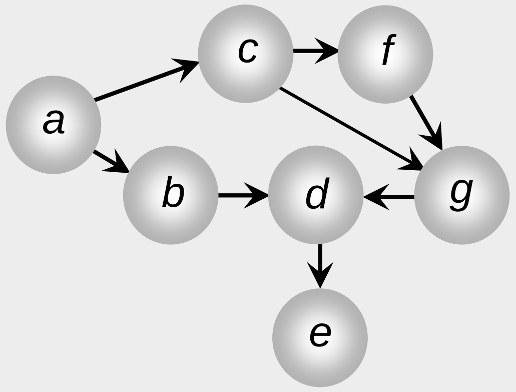
1.7 Graph Theory Terminology

Directed graphs with cycles

Directed acyclic graph



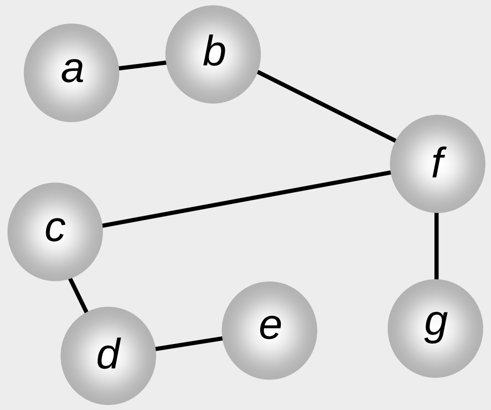


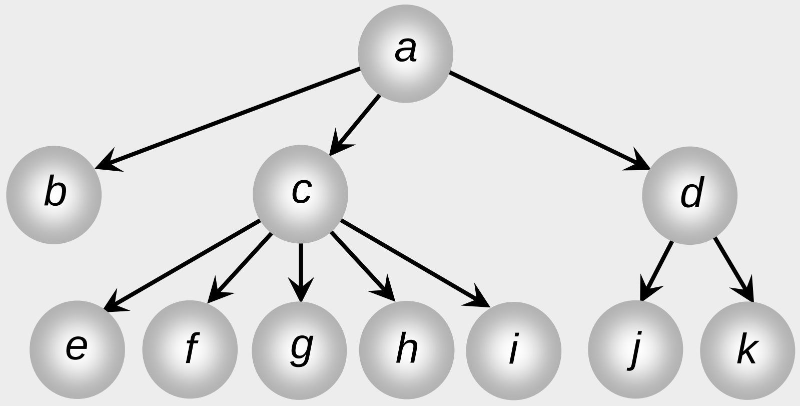


1.7 Graph Theory Terminology

Undirected graph with maximum node degree 3

Directed tree

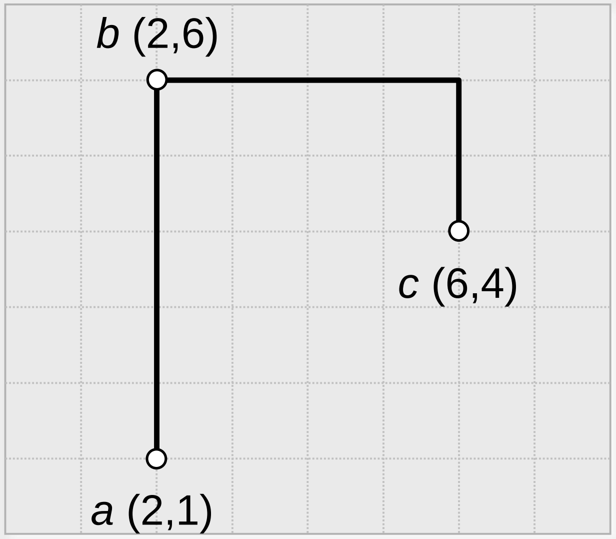


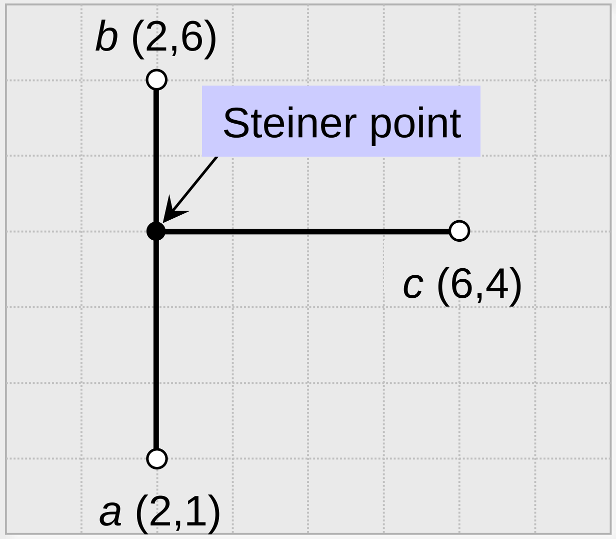


1.7 Graph Theory Terminology

Rectilinear minimum spanning tree (RMST)

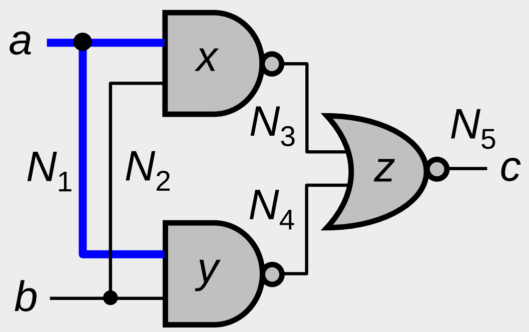
Rectilinear Steiner minimum tree (RSMT)





1.8 Common EDA Terminology

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)

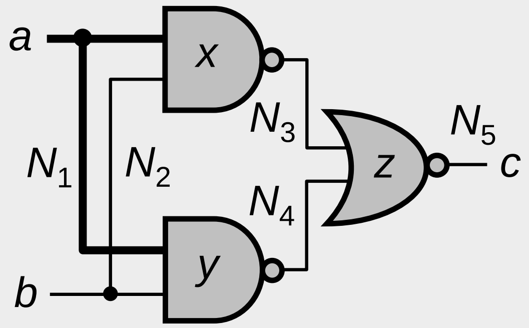
Pin-Oriented Netlist

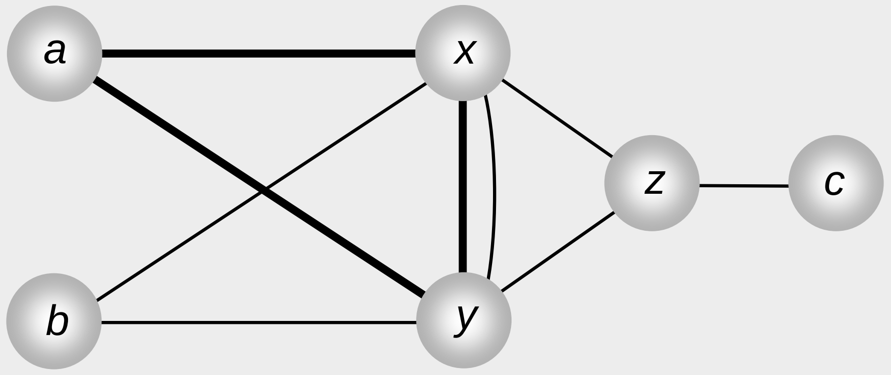
(N_1: a, $x.IN1$, $y.IN1$)
(N_2: b, $x.IN2$, $y.IN2$)
(N_3: $x.OUT$, $z.IN1$)
(N_4: $y.OUT$, $z.IN2$)
(N_5: $z.OUT$, c)

Net-Oriented Netlist

Common EDA Terminology

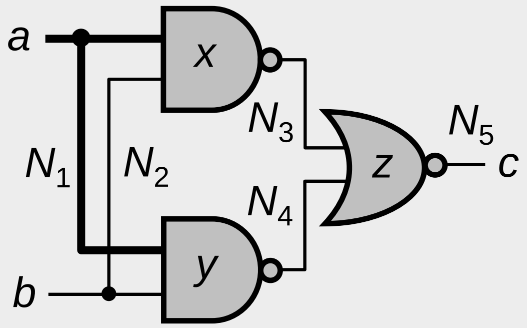
Connectivity graph





1.8 Common EDA Terminology

Connectivity matrix



$$
\begin{array}{c|cccccc}
 & a & b & x & y & z & c \\
\hline
a & 0 & 0 & \mathbf{1} & \mathbf{1} & 0 & 0 \\
b & 0 & 0 & 1 & 1 & 0 & 0 \\
x & \mathbf{1} & 1 & 0 & \mathbf{2} & 1 & 0 \\
y & \mathbf{1} & 1 & \mathbf{2} & 0 & 1 & 0 \\
z & 0 & 0 & 1 & 1 & 0 & 1 \\
c & 0 & 0 & 0 & 0 & 1 & 0
\end{array}
$$

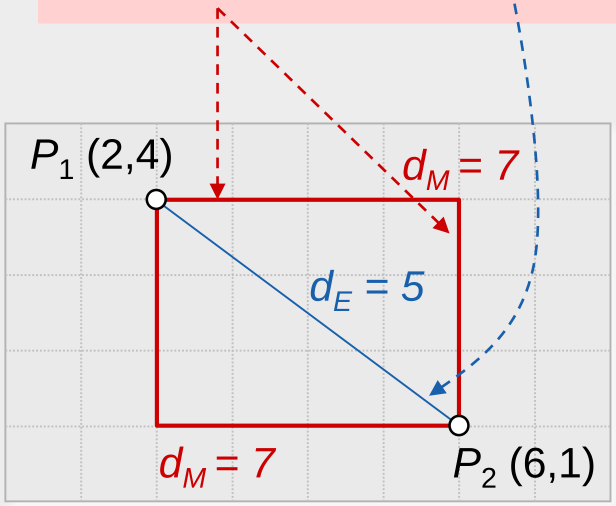
1.8 Common EDA Terminology

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



Next Lecture: Netlist and System Partitioning