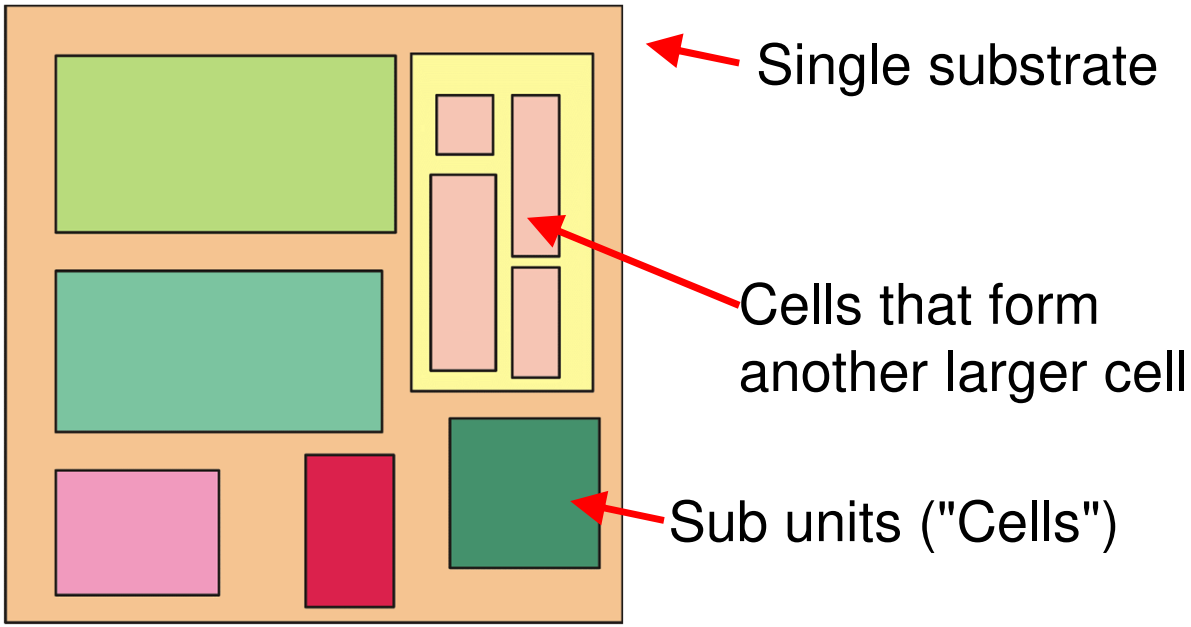
Design Flow of an Integrated Circuit

Integrated Circuit or "Chip" or "Die"



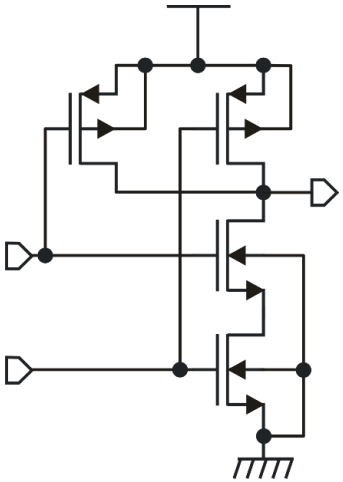
The cell is the most important object in an integrated circuit

Even the **chip** is a cell: it is the larger cell in the project and is often called **Top-Cell**

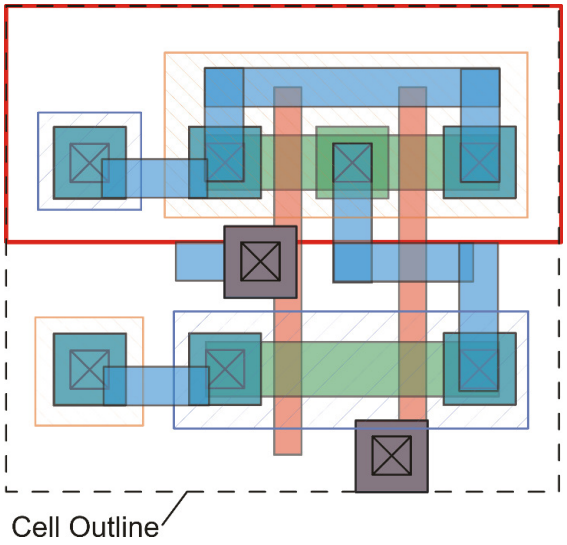
Cells and views

Cell: NAND GATE

Views



"Schematic view"



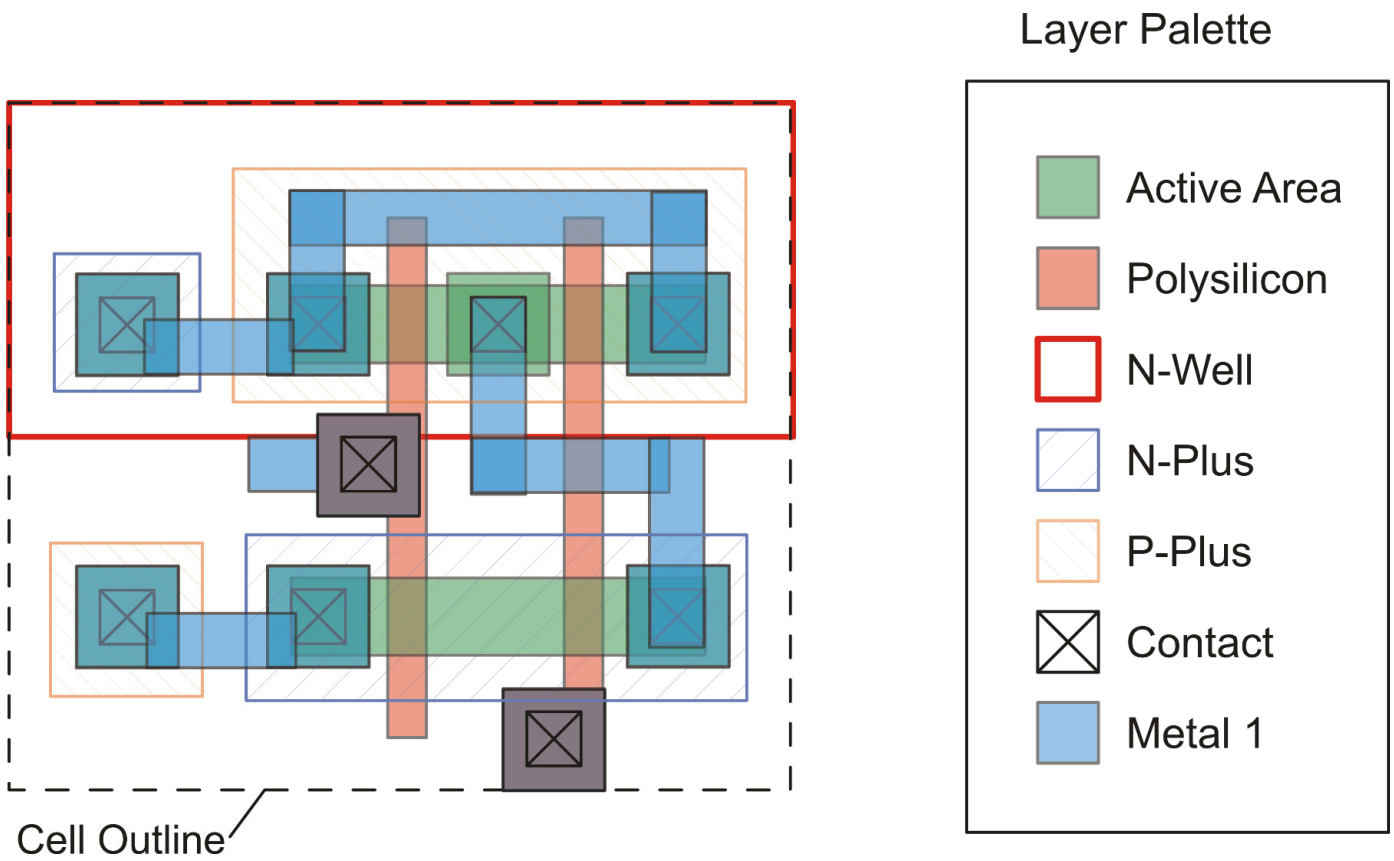
"Layout view"

• • •

?

• • •

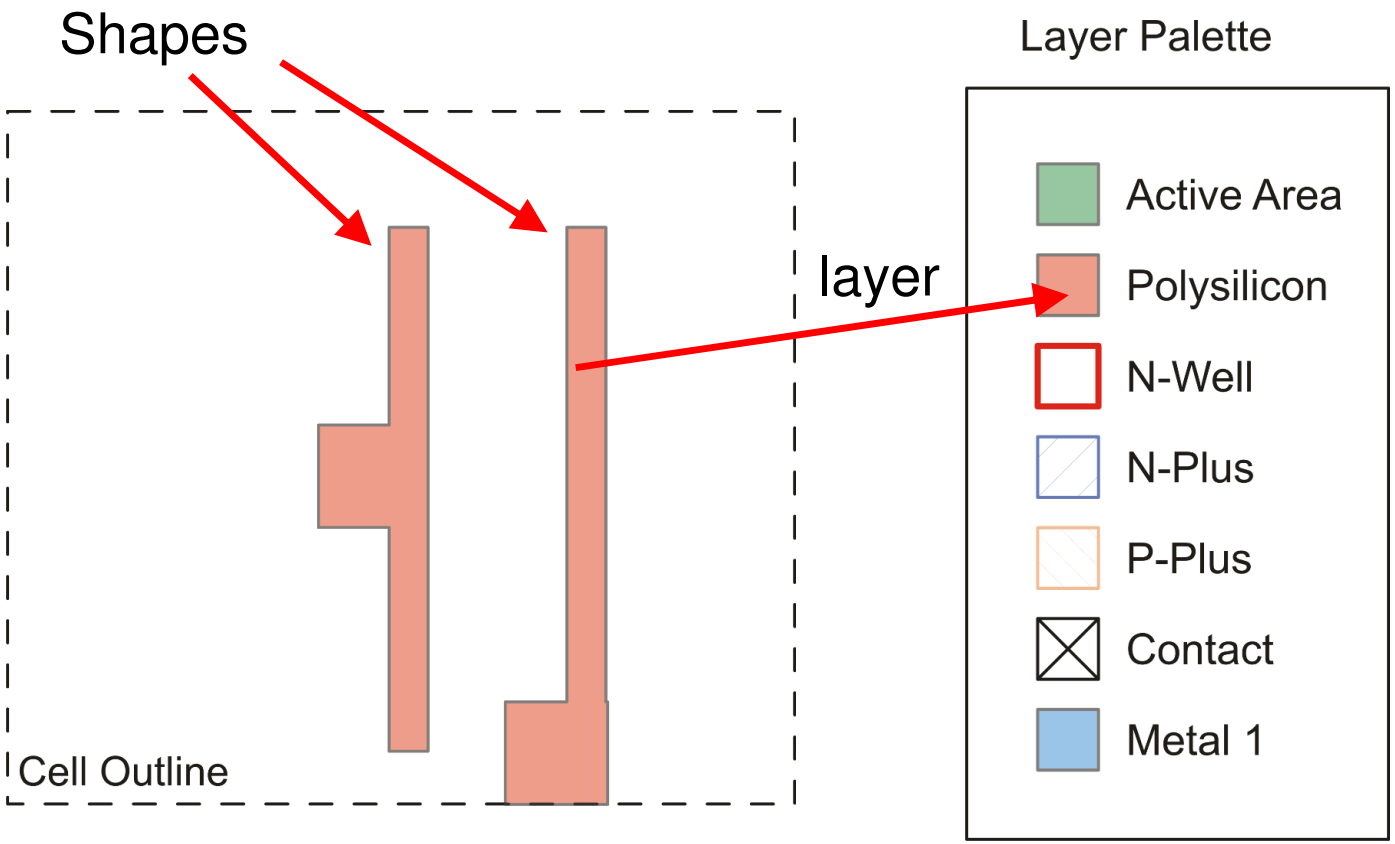
The layout view



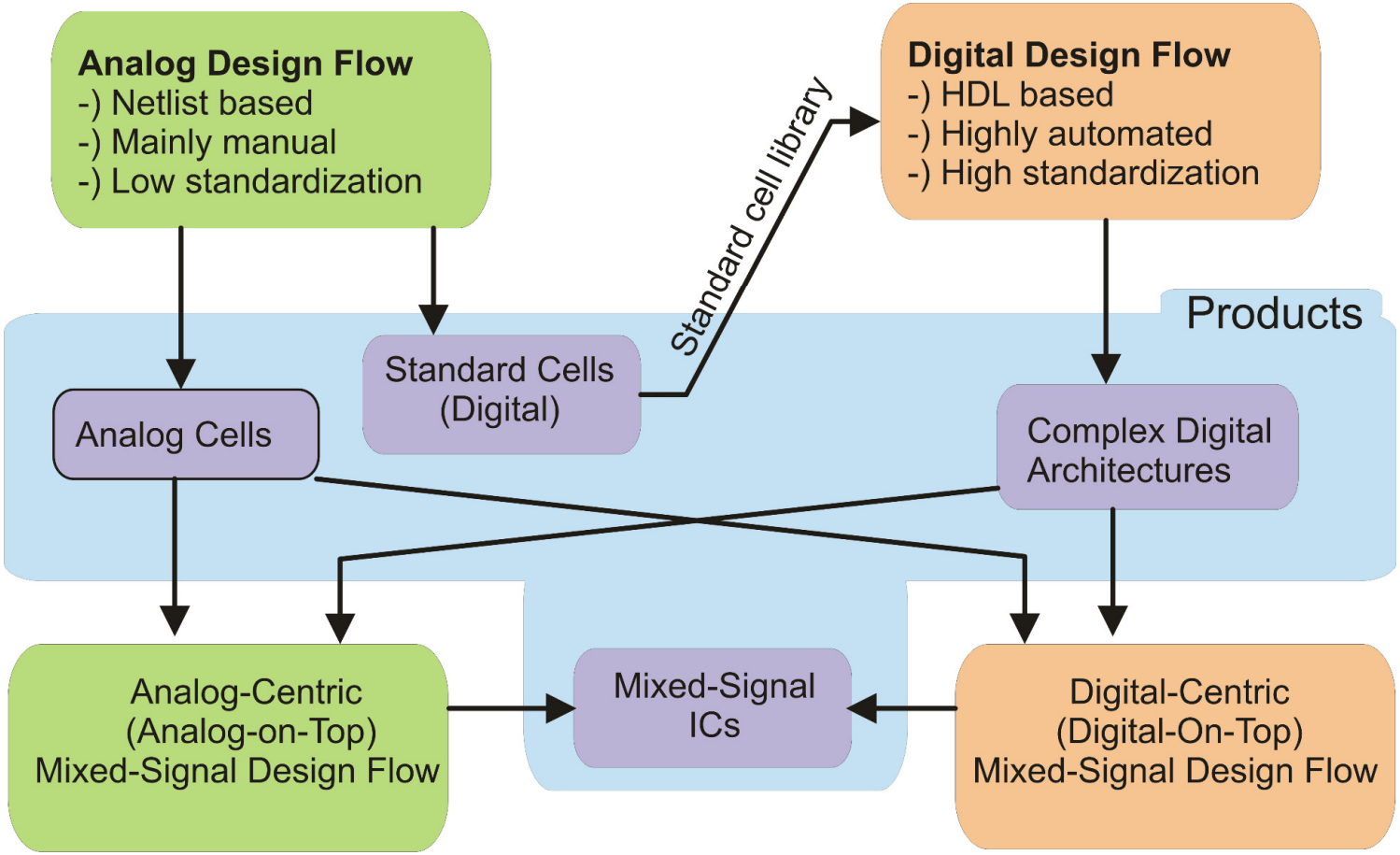
Shapes and layers

A **shape** selects a region of the cell area.

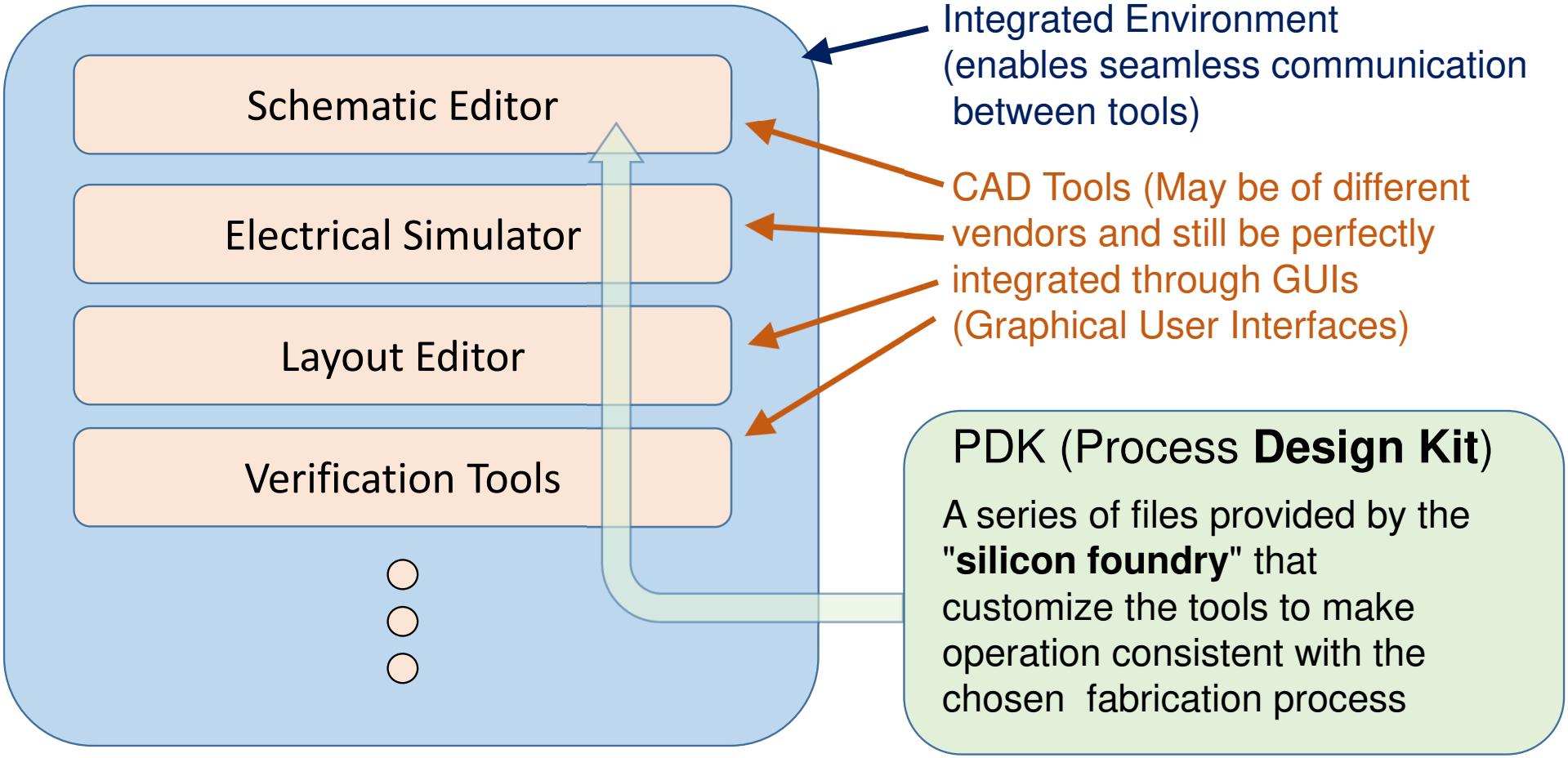
The **layer** associated to a shape tells specify what must be done in that region.



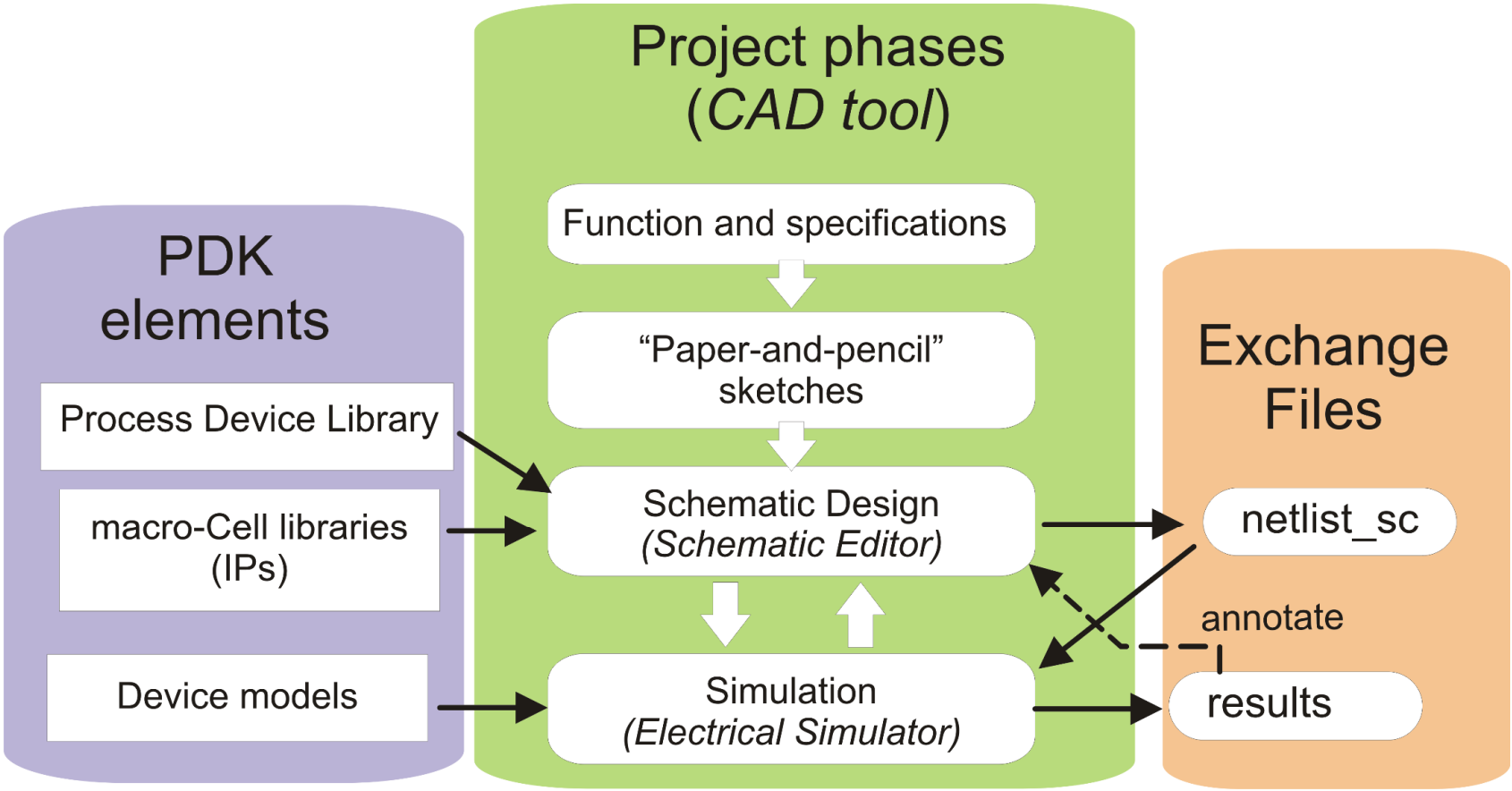
Design flows



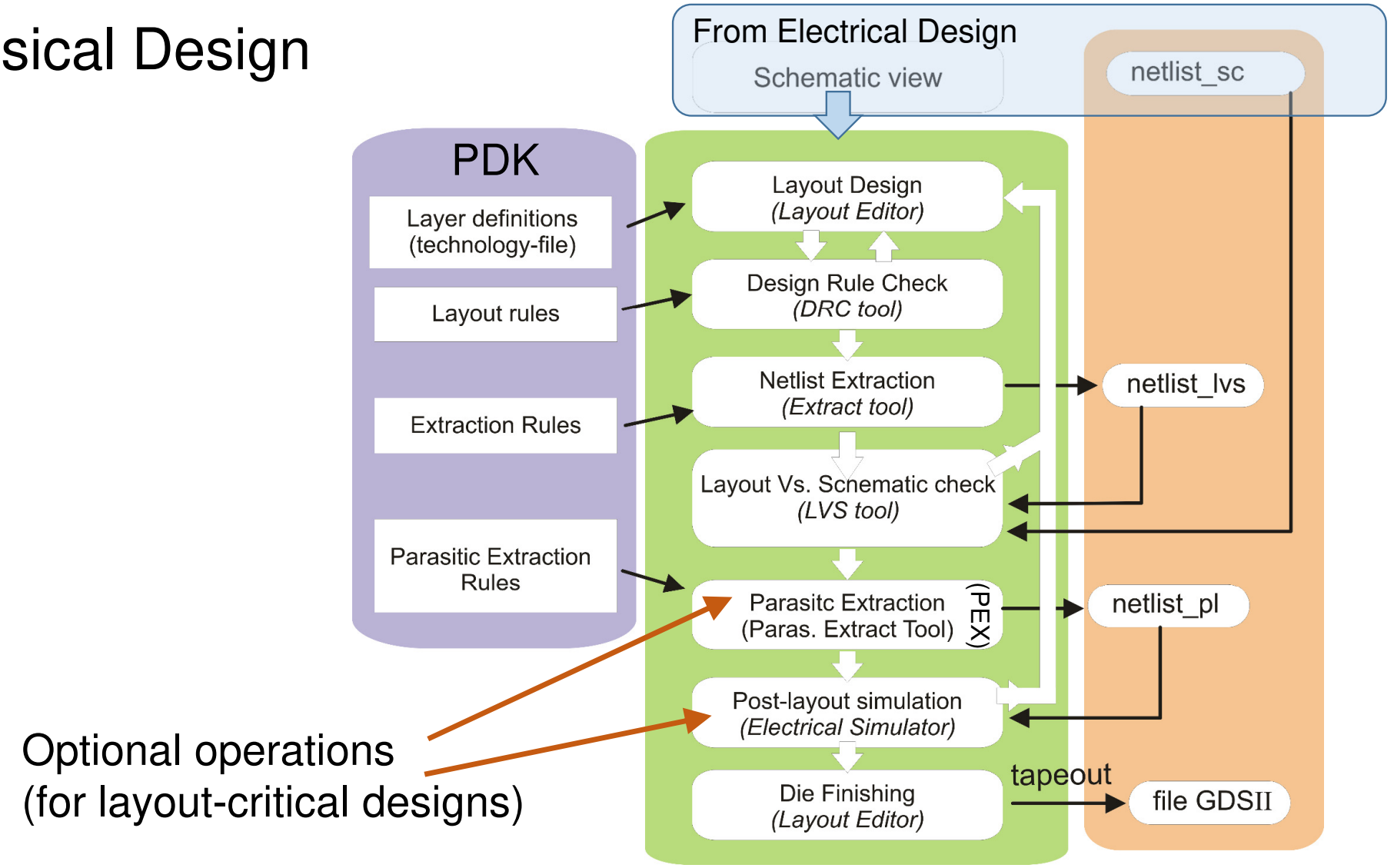
EDA (Electrical Design Automation) Environment



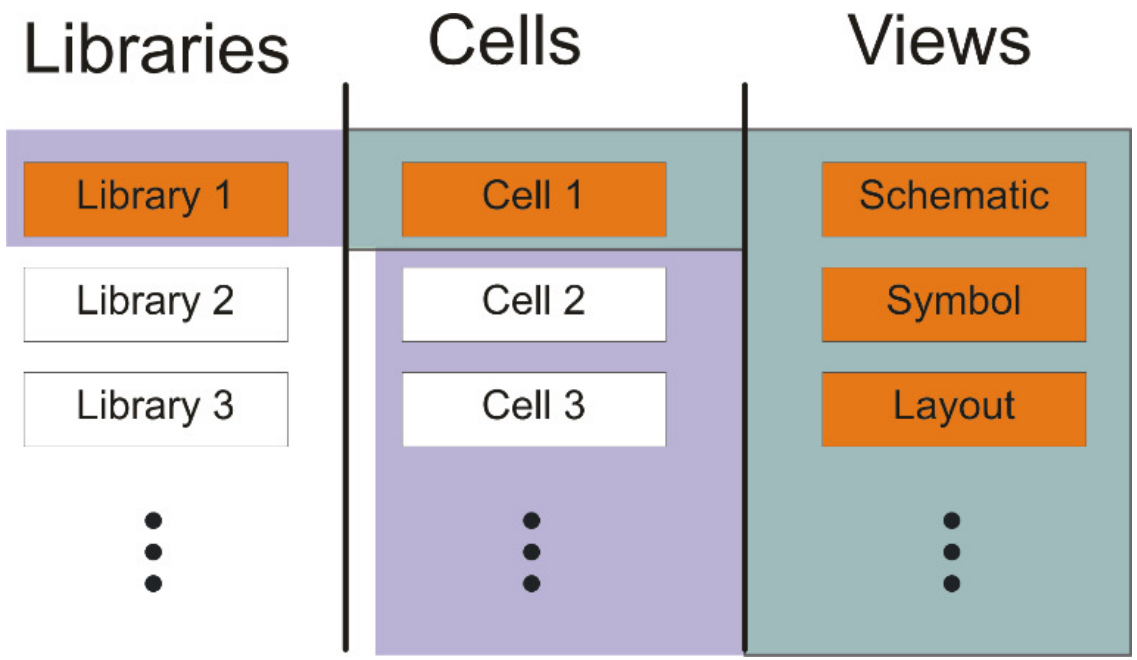
Analog Design Flow: Electrical Design



Physical Design



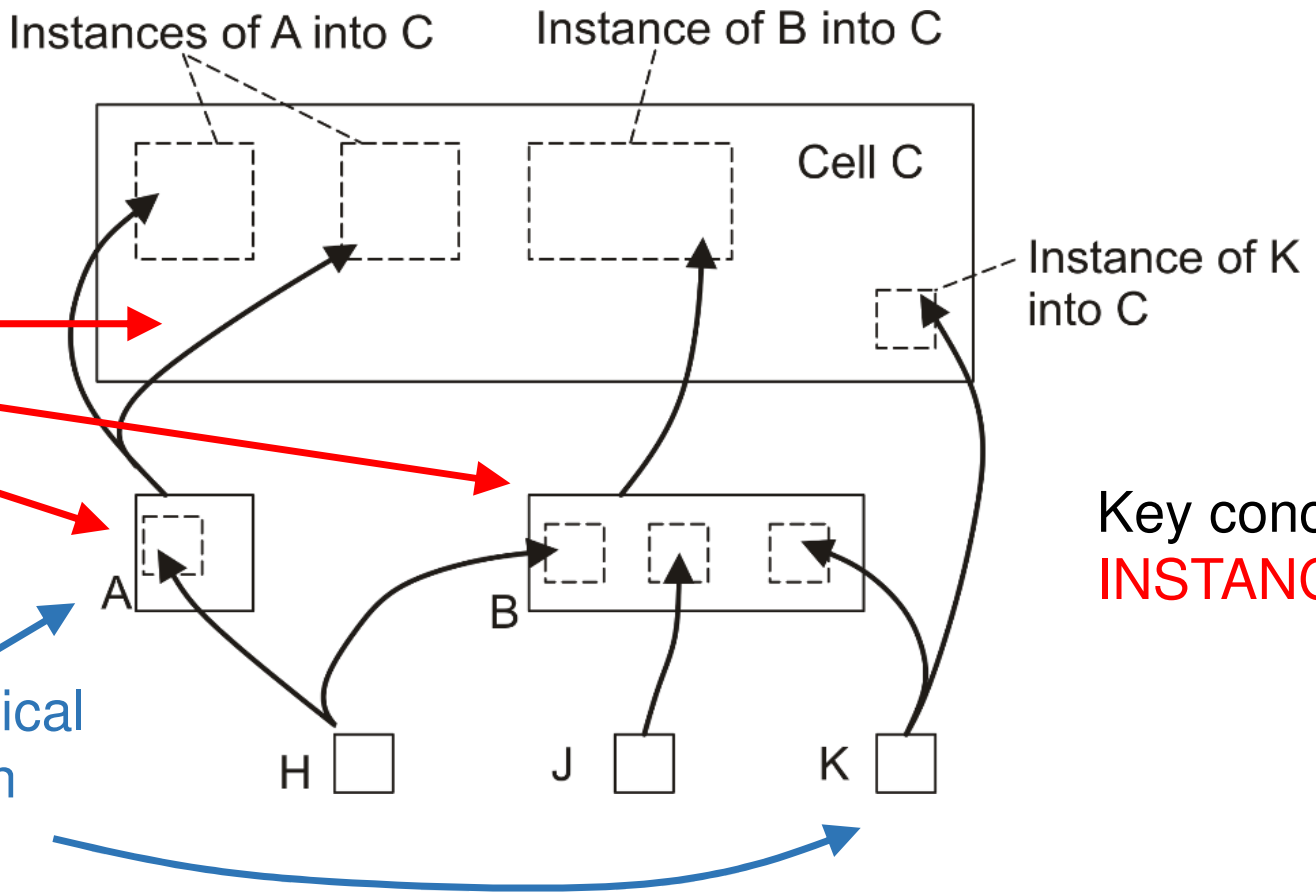
Structure of a project (= "design")



Libraries are containers for the cells that include additional information about the technology. Library can be provided with the PDK (devices and macrocells), with the CAD environment (ideal devices) or be created by the designer / design group.

Structure of a project: **Hierarchy**

Cell C is higher than A and B in the hierarchy, since A and B are instantiated into C

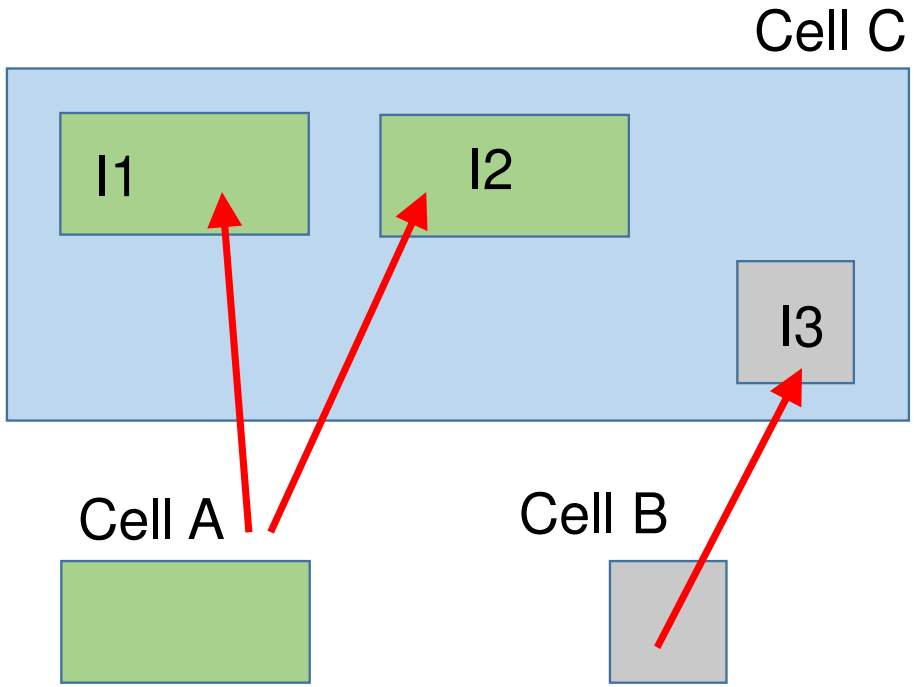


There is no hierarchical relationship between K and A

Key concept:
INSTANCE

Instances: properties

Instances of the same cell are **individual objects** just as two distinct devices in a printed circuit board



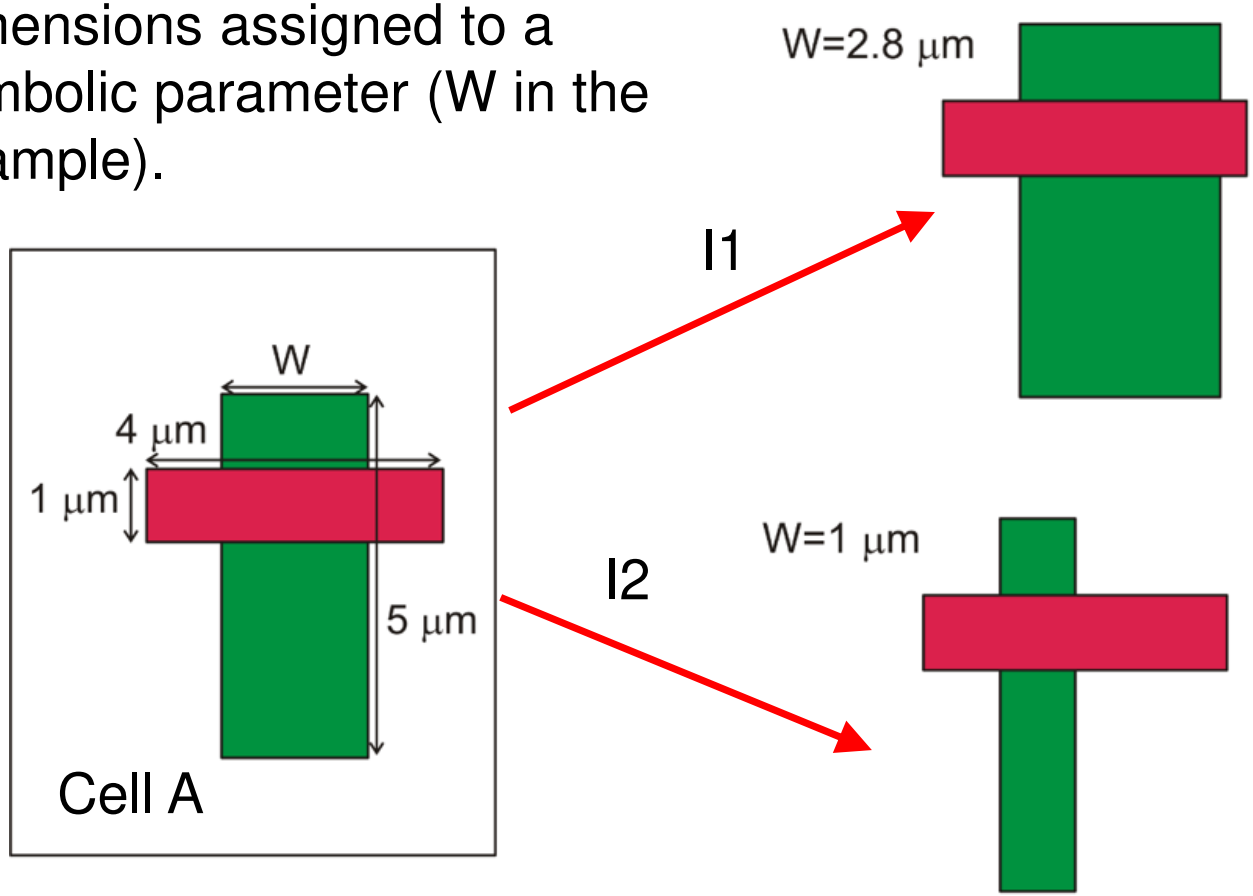
Library	Cell
my_design	C
.....	A
.....	B
.....	.....

In the "library browser" the hierarchy is not visible: all cells appear as they were all at the same level.

The only difference from real world is that modifying the original cell (e.g. Cell A) changes **all** the instances of A

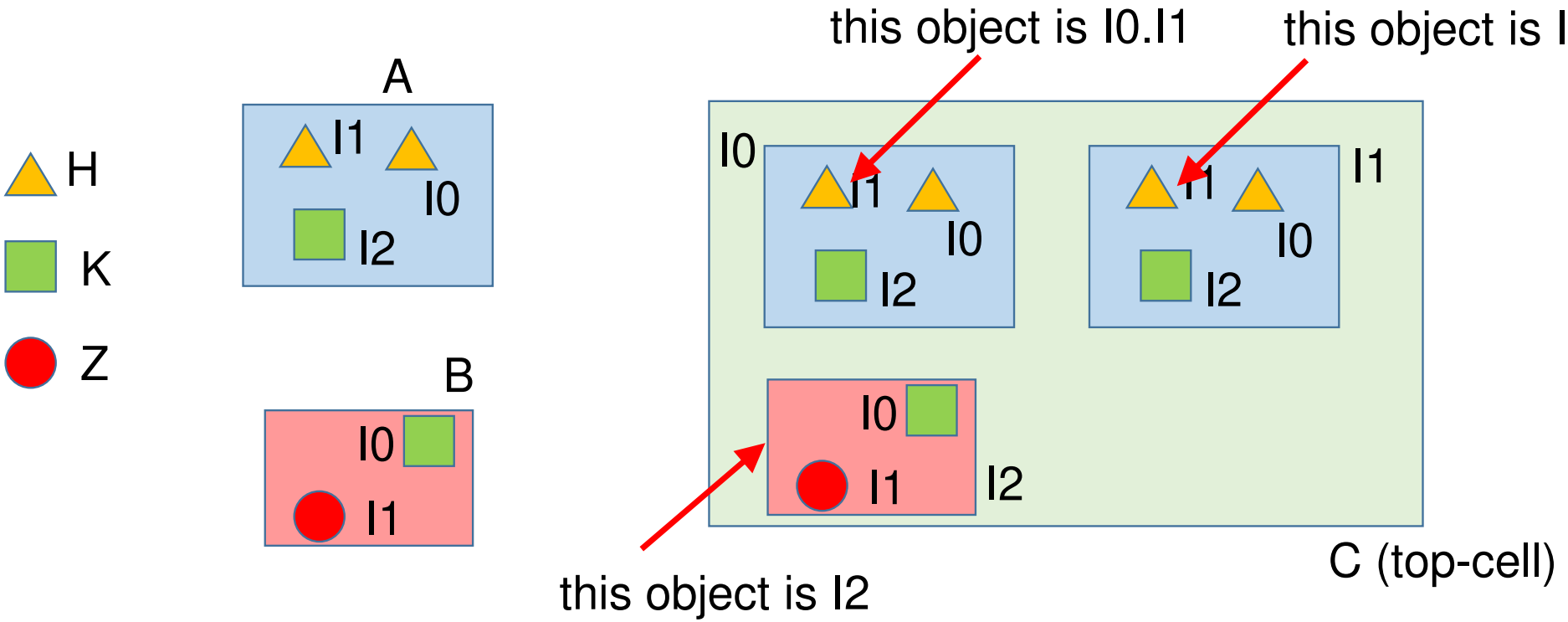
Parametric cells (P-cells)

P-cells have one or more dimensions assigned to a symbolic parameter (W in the example).



When a **P-cell** is instantiated, the value of all symbolic parameters should be chosen. In this way it is possible to create custom instances from a single P-cell.
P-cells are possible in both the schematic and layout view.

Nested instances: terminology (Cadence and Spice)



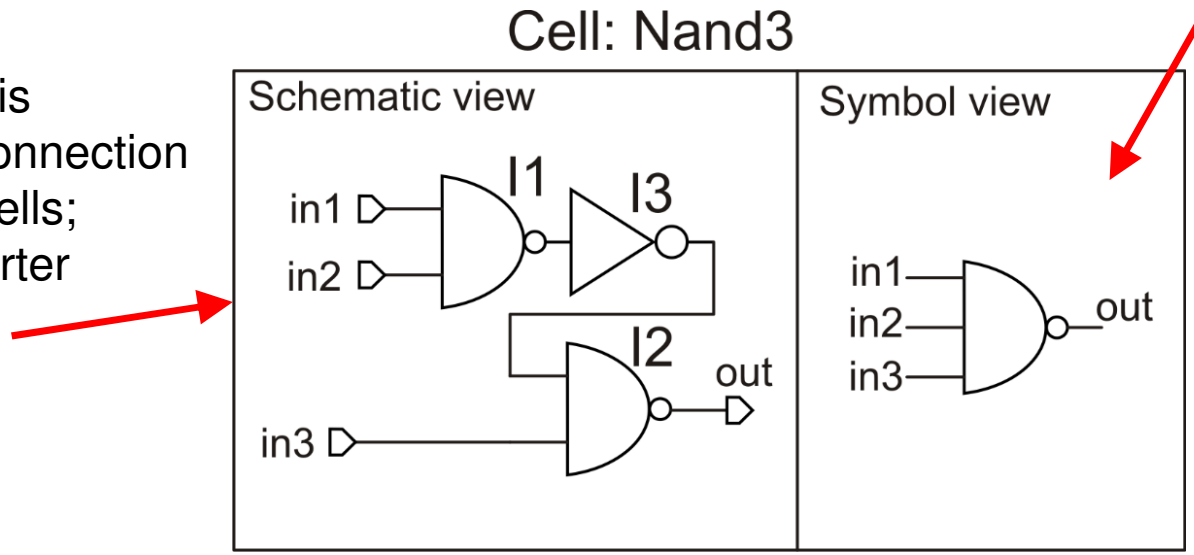
In some tools the "." is substituted by "/" , then I1.I0 becomes I1/I0

The schematic view

A schematic view is the representation of a cell as **connection of simpler cells**. Some cells do not have a schematic view because they cannot be resolved into simpler ones. They are the "atoms" of the design and are called "primitive cells" or "**primitives**".

In a schematic view, each cell that needs to be instantiated requires also a symbol view, where only the nodes used for the connections are displayed

The cell Nand3 is obtained as a connection of two simpler cells; Nand2 and Inverter



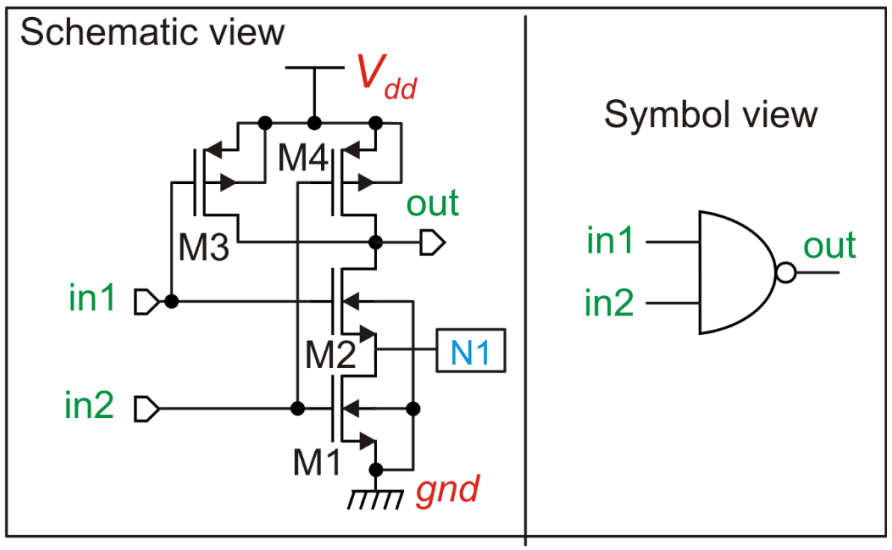
Elements of a schematic view

Nodes:

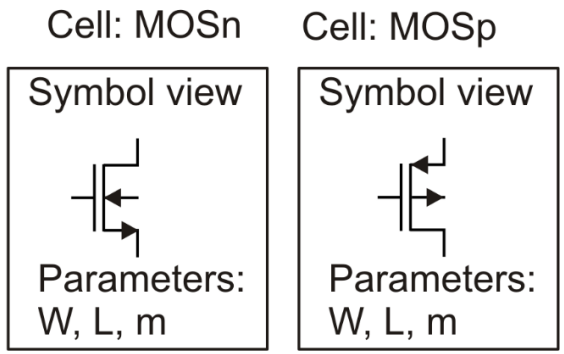
-) Terminals:
in1, in2, out

-) Global:
Vdd, gnd

-) Internal:
N1



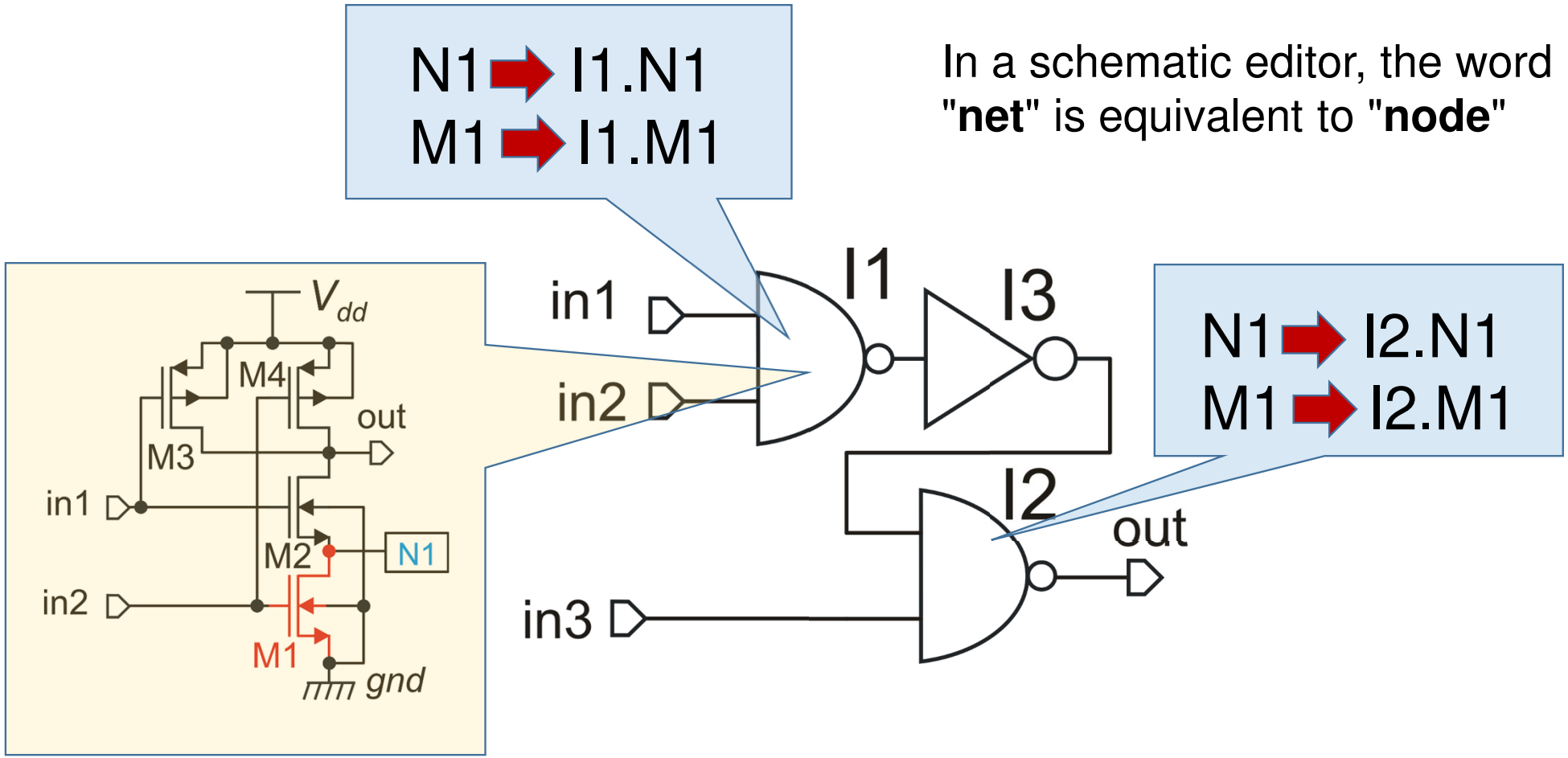
Primitive cells
(parametric)



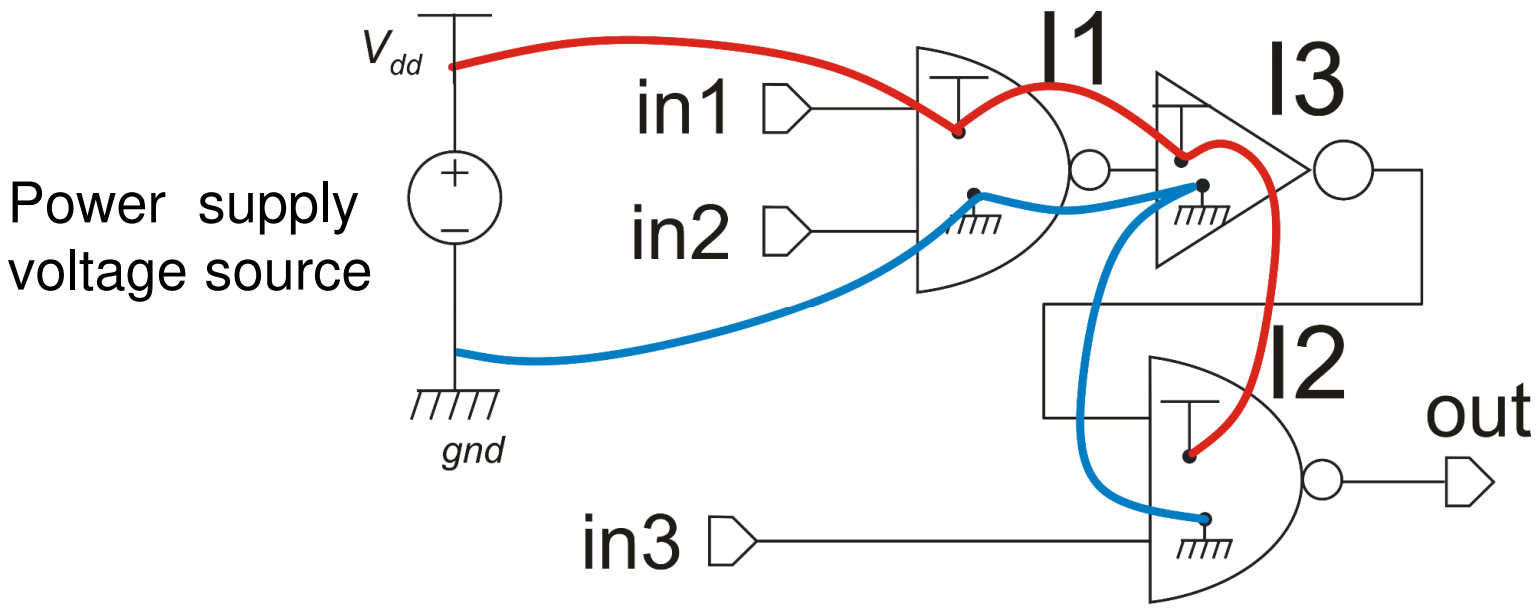
Note: internal nodes are automatically numbered and given a name such as: N1, N2 ....
It is possible to force an internal node to have an arbitrary name by assigning a "**label**" to it

Assigning the same name (label) to different nodes is equivalent to connect them to form a single node.

Access to internal nodes and devices: syntax

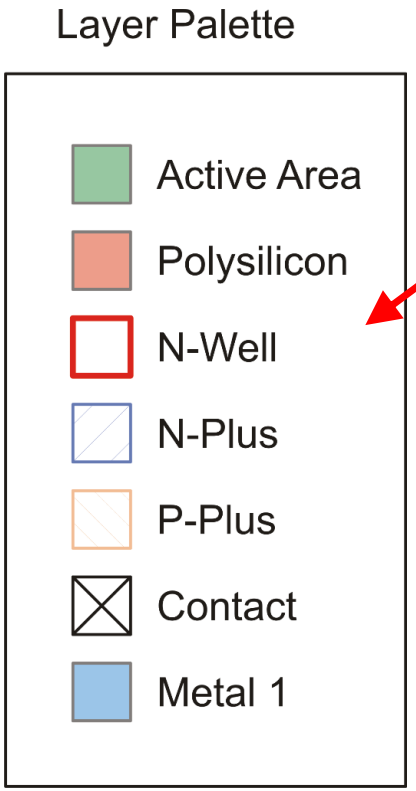


Global nodes



Global nodes of the same name (e.g. *gnd*) in different instances at whichever hierarchy level are all **connected together** with no need of specifying the connections (better readability of the circuit).

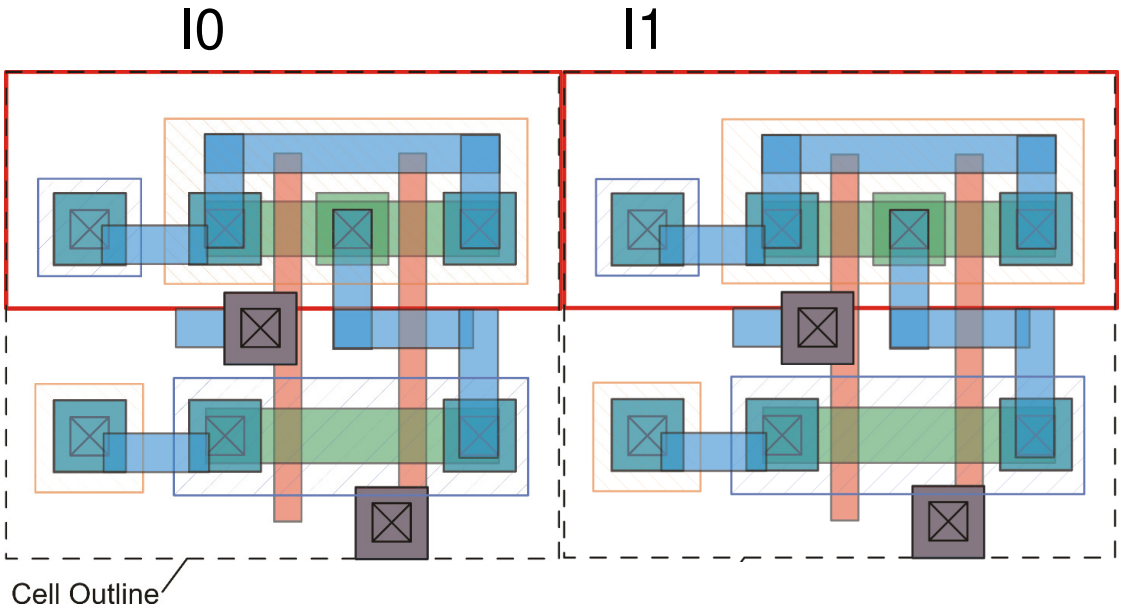
Elements of a Layout Editor



Type of layers

- **Technology or "Tooling" Layers.**
  (Used to define the result of elementary process steps) i
- **Derived Layers**
  (obtained by logical operations of other layers)
- **Service Layers**
  (used to various purposes, e.g. mark shapes as pins or define the boundary of a cell)

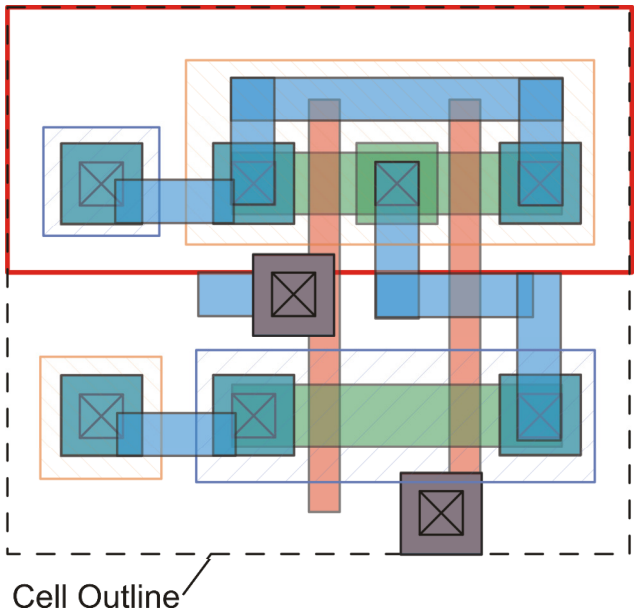
Instances in the Layout Editor



There is not a perfect equivalent of the "symbol" view in the layout. Therefore, the instantiated cells appear with all the complexity of the internal layout. Instances can be connected, moved, rotated etc, but not modified.

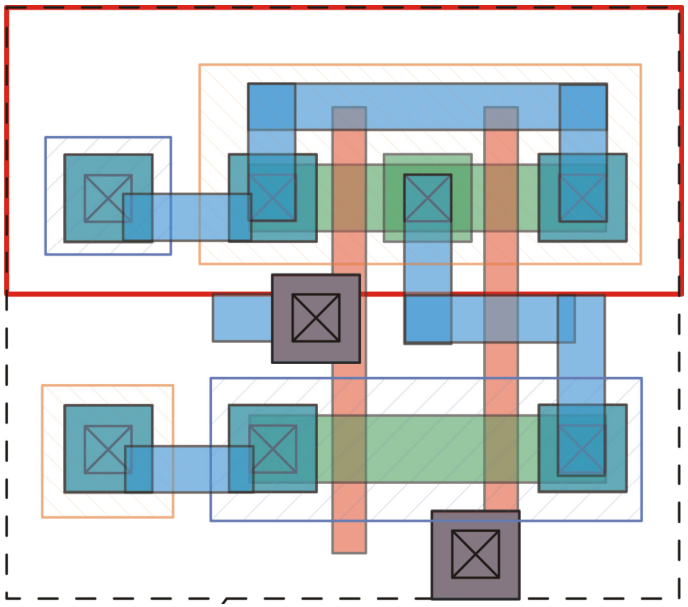
The "abstract" view

The abstract view is a simplified layout where only the layers used for interconnections are shown. The abstract view is used to hide the full content of the cell (to avoid cloning) or to simplify automatic place & rout operation.



While the symbol view is strictly required in a schematic design, the abstract view can be avoided all the cell that we have to instantiate own a full layout view

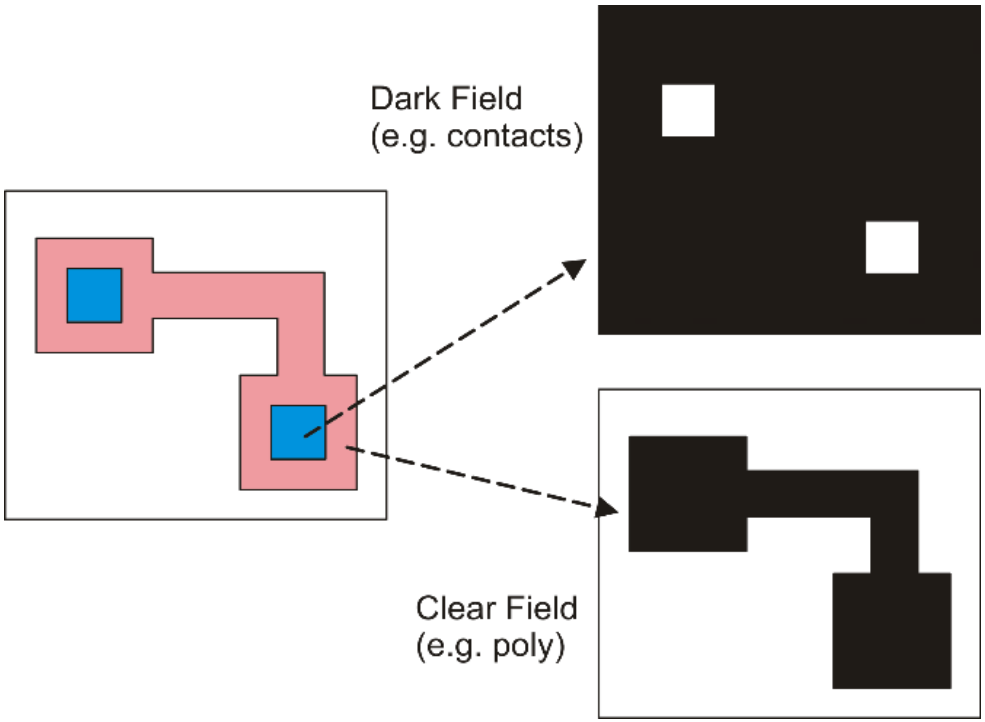
Layout elements: layers, shapes and masks



Tooling Layers directly affect the mask production but there is not an exact correspondence between the shapes of a layer and a single mask.

- Shapes correspond to the final result on silicon. For example, if I draw a 1 µm wide polysilicon line, the corresponding mask can be wider or narrower to compensate for proximity effects or under-etch. The final result will be as close as possible to 1 mm.
- A shape may indicate an object that requires more than one photolithographic step and then more than a single mask.
- Shapes in more than 1 layer can be merged to form a single mask, or to produce different masks through logic combinations (AND, OR …). These operation are indicated as "mask merging rules".

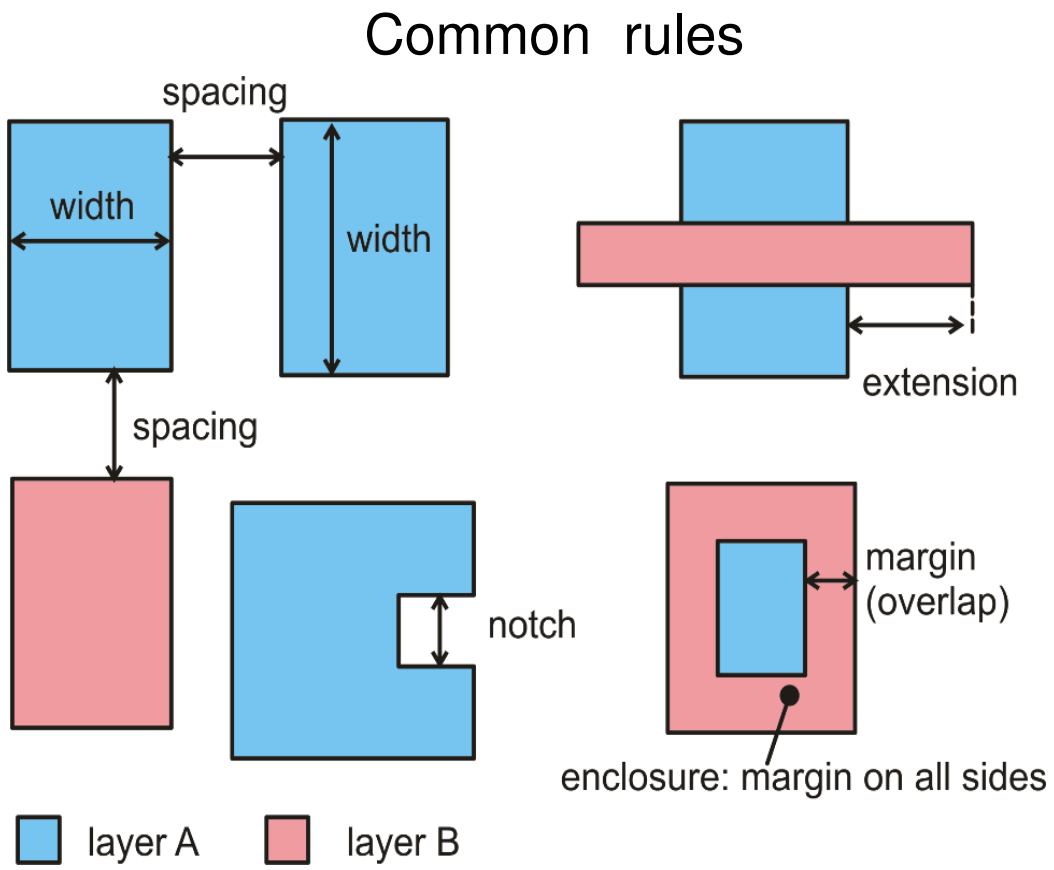
Dark Field, Clear Field



The shapes drawn with a layer can be converted into the dark pattern of the mask on a clear background (=Clear Field)
or clear (transparent) patterns on a opaque background (=Dark Field)

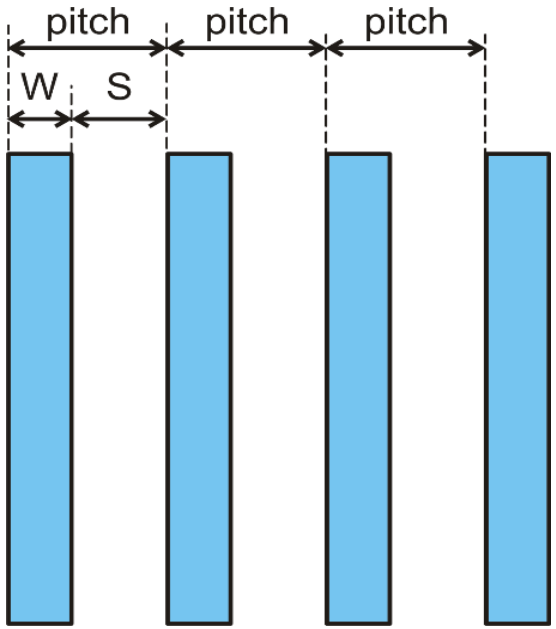
Topological Layout Rules (TLR)

Topological Layout Rules indicate how to draw shapes that can be actually fabricated maintaining the function that the designer intend to obtain.
For example, conducting lines should be wide enough to guarantee that they are not interrupted in same points.



The pitch: combination of TLRs

Considering a particular layout object, that can be a single conducting line or even a complex memory cell, the pitch is the minimum step by which we can repeat that object without violating layout rules.
The picture shows the case of conducting lines, that can be stepped to form a data BUS.

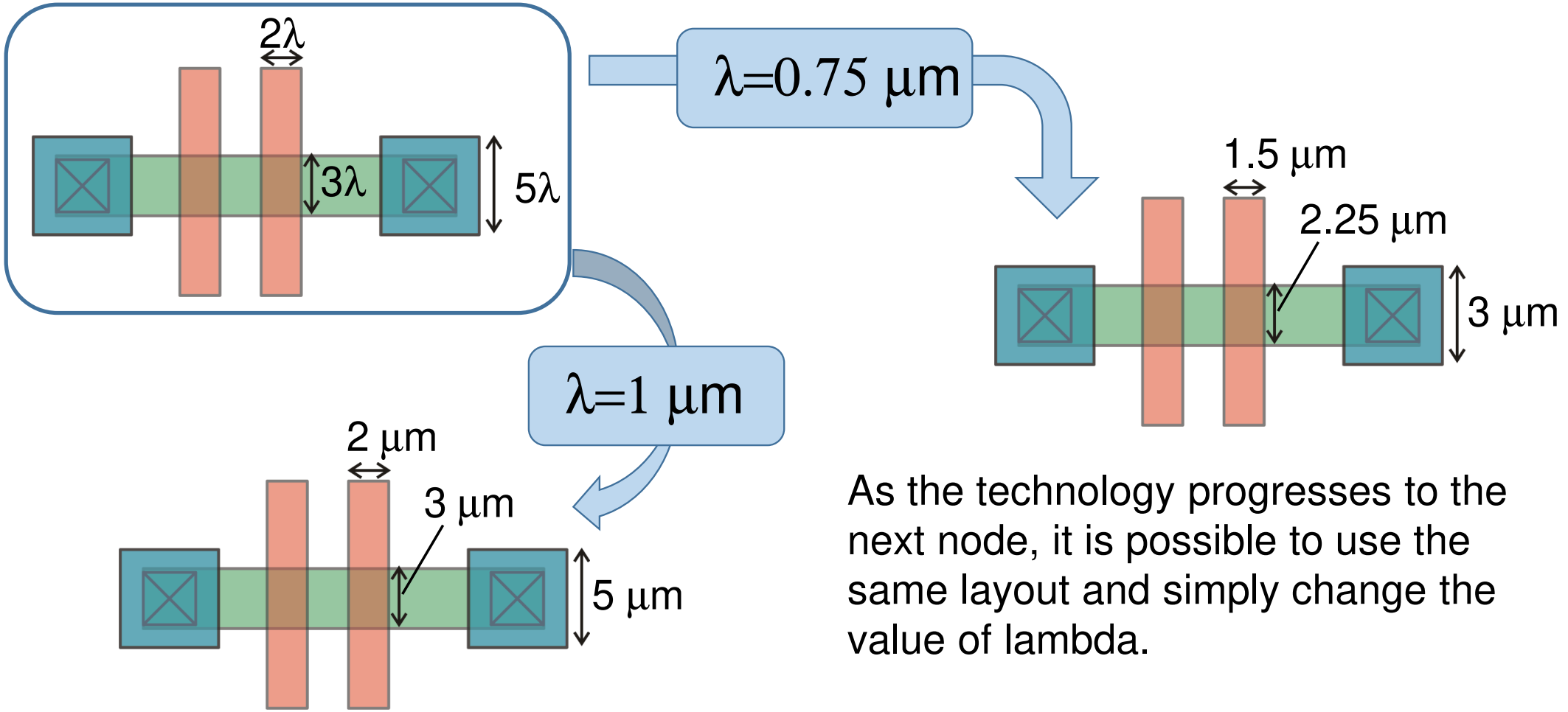


$$pitch = W + S$$

Micron and Lambda (scalable) Rules

- **Micron rules** are today the industrial standard. In Micron Rules, all Topological Layout Rules are expressly indicated in length units, typically microns (μm) or nanometers (nm).
- **Lambda rules**, proposed by Mead and Conway, were popular in the 80s. In lambda rules, Topological Layout Rules are expressed as multiples of an elementary length called lambda (λ).

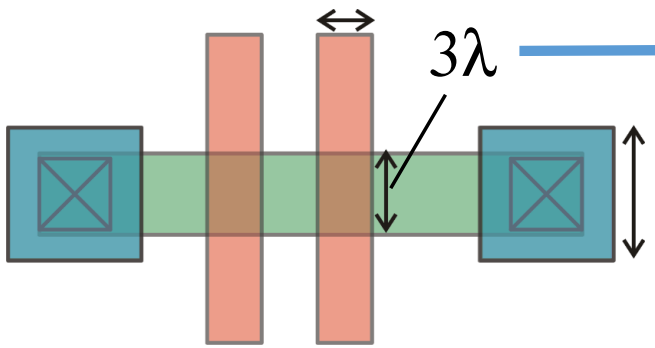
Lambda rules and uniform TLR scaling



As the technology progresses to the next node, it is possible to use the same layout and simply change the value of lambda.

Limits of Lambda Rules

CMOS Process
0.18 µm



λ=0.09 µm

Technology upgrade (one "node")

2λ	→	0.18 µm	× 0.72	→ 0.13 µm
3λ	→	0.27 µm	× 0.81	→ 0.22 µm
5λ	→	0.45 µm	× 0.89	→ 0.40 µm

CMOS Process
0.13 µm

λ=??

Lambda Rules today

- Starting from roughly the 1 μm technology node, TLR stopped to improve by a uniform factor. In order to save the scalability of the original layout designed with lambda rules, it would be necessary to scale all rules using the scaling factor of the TLR that have improved less (factor 0.89 in the previous example). This would turn into a layout of non-minimum area and part of the advantages of the technology progress would be lost.
- For this reason, scalable rules (i.e., lambda rules) are used today only for educational purposes (e.g., in the American MOSIS program:).
- It is common practice of silicon foundry to update mature processes by applying a uniform shrink to all rules. This shrink is generally modest (e.g. new rules are multiplied by 0.9) and arrives when the technology has already advanced to a new node.