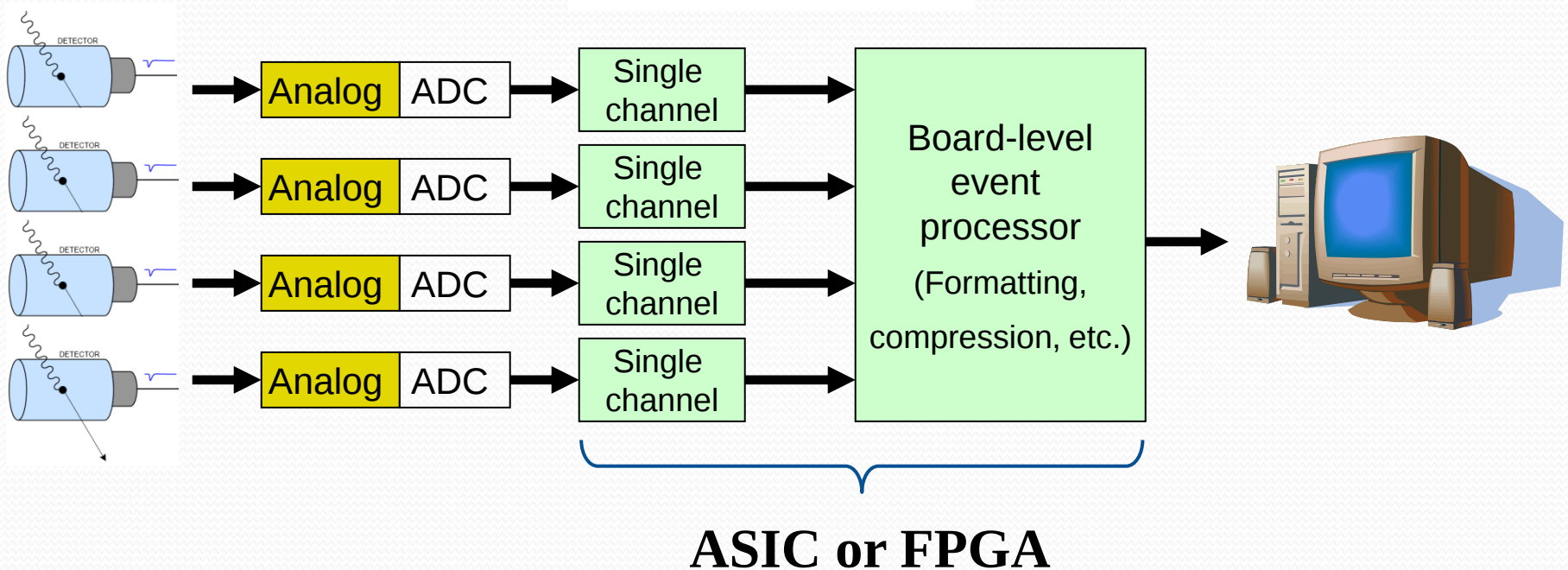
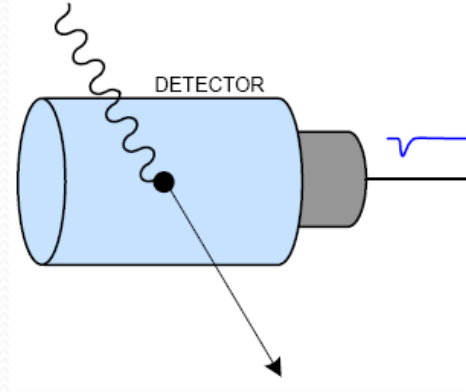


VHDL

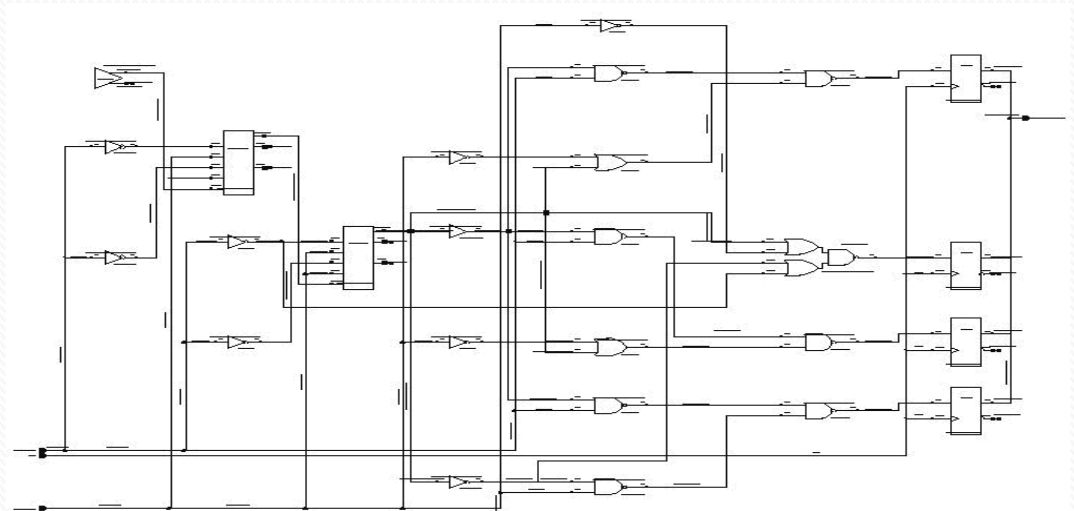
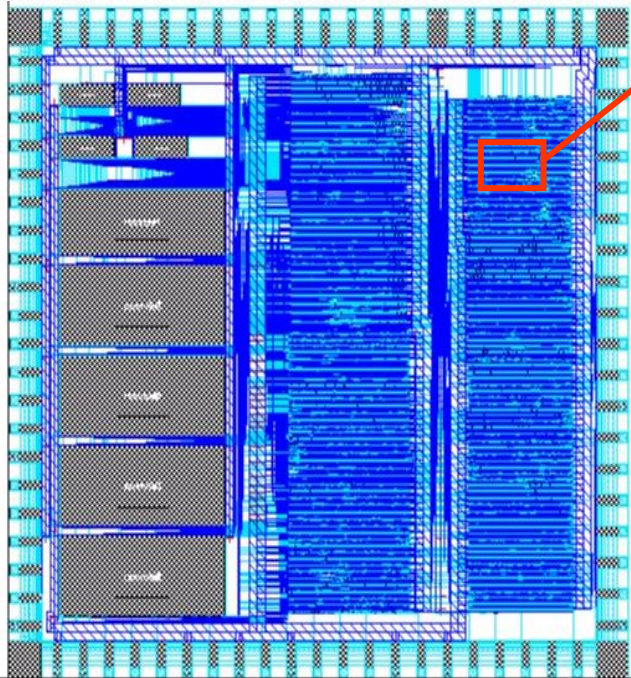
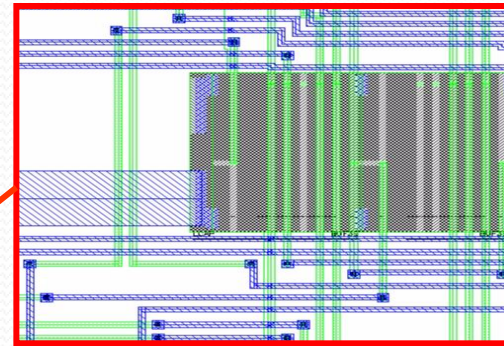
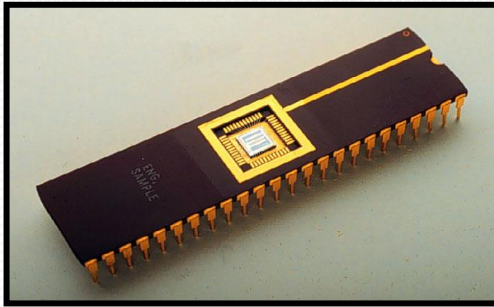
Main topics:

- An introduction to VHDL
- VHDL basics and syntax
- Advanced VHDL language structures
- Circuit examples: simulation and synthesis

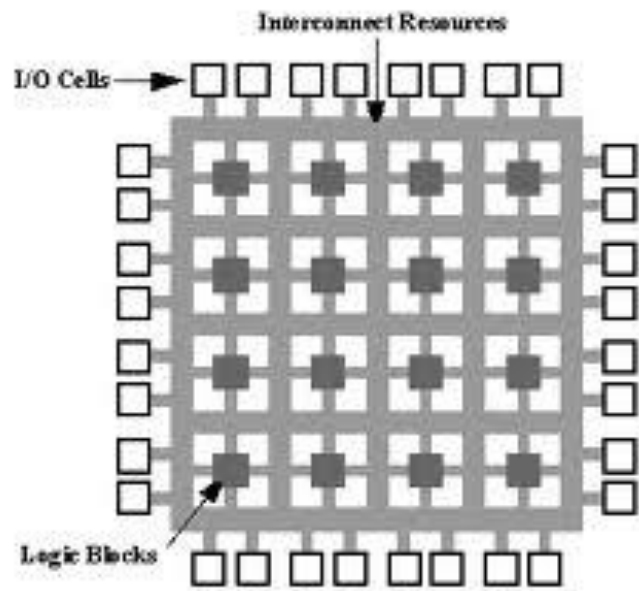
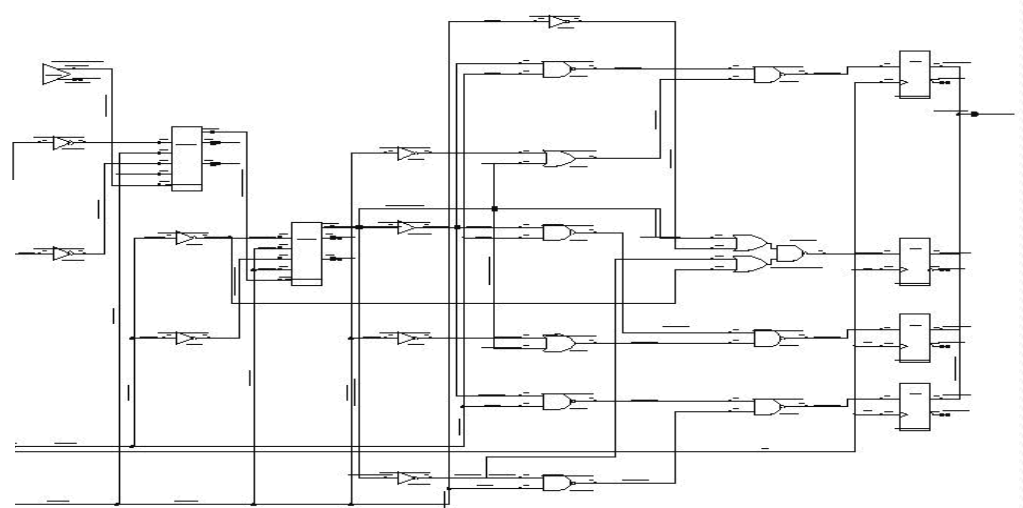
Detector readout chain



ASIC: Application Specific Integrated Circuit



FPGA: Field Programmable Gate Array



Schematic entry

Pros:

- intuitive for small projects
- many designers have been working in this way for years: why change ?

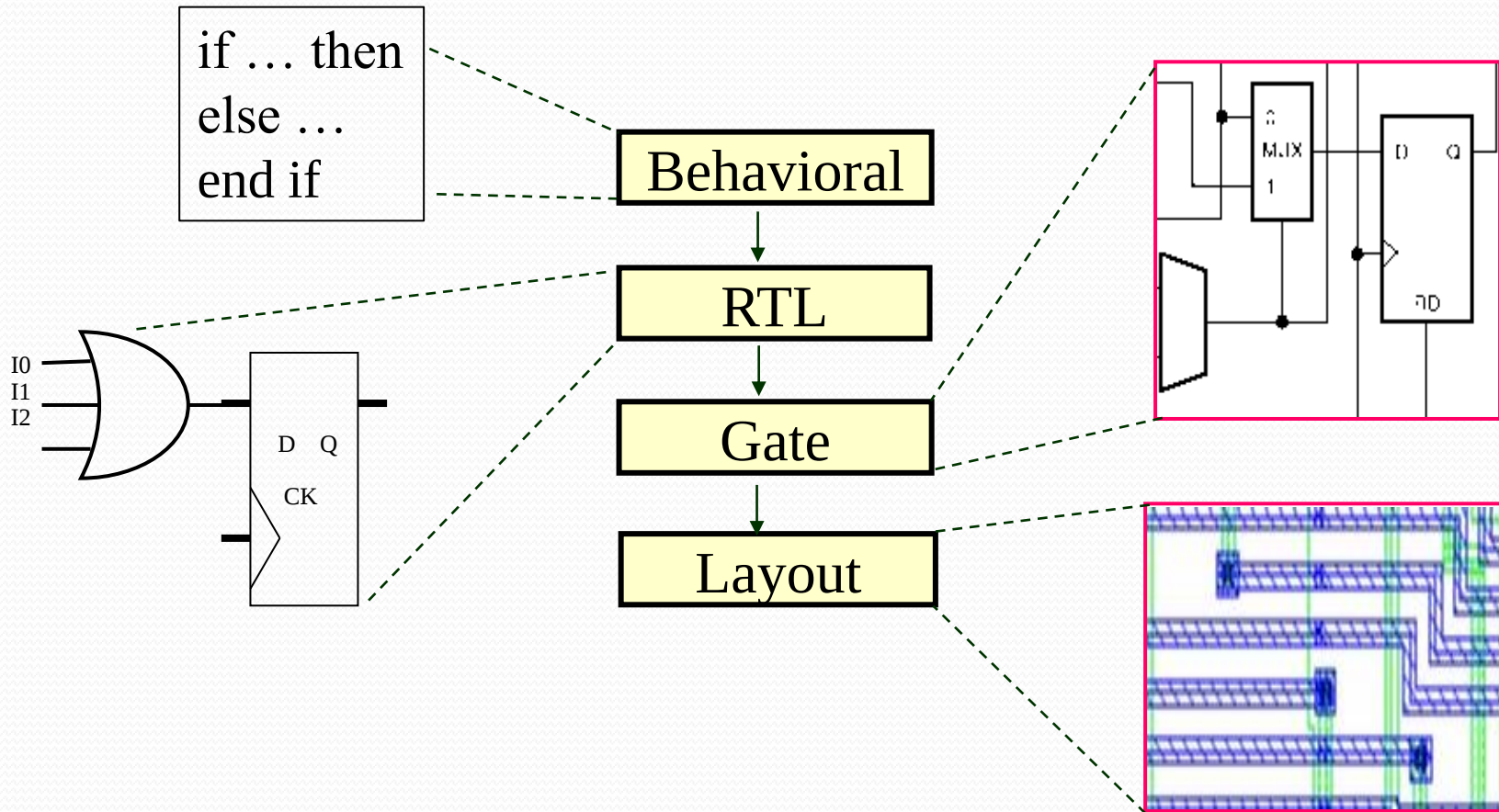
Cons:

- difficult to manage when complexity grows (million gates ASICs-FPGAs)
- difficult to port projects to different technologies
- difficult to maintain for a long lifetime
- difficult to port projects to different EDA tools

What is VHDL ?

- **VHDL** is an acronym of **VHSIC** (Very High Speed Integrated Circuits) **Hardware Description Language**
- **VHDL** : a formal language for specifying the behavior and structure of a digital circuit
- Allows to raise the entry abstraction layer to a level which is more user friendly than schematics

VHDL: Abstraction Levels



VHDL: Abstraction Levels

BEHAVIORAL: functional description of the model. Used at the very beginning stage of a design in order to be able to run a simulation as soon as possible. Also used to describe testbenches. **Such descriptions are usually simulatable, but not synthesizable.**

RTL: the description is divided into combinational logic and storage elements. The storage elements (flip-flops, latches) are controlled by a system clock. **The description is synthesizable.**

GATE: the design is represented as a netlist with gates (AND, OR, NOT, ...) and storage elements, all with cell delays. **The description has been synthesized.**

LAYOUT: the different cells of the target technology are placed on the chip and the connections are routed. After the layout has been verified, **the circuit is ready for the production process.**

VHDL: Behavioral Description

In a behavioral VHDL description, a Boolean function, for example, can be modeled as a simple equation (e.g. $i1 + i2 * i3$) plus a delay of N ns. The **worst case**, i.e. the longest delay to calculate a new output value, is assumed here. Functional behavior is modeled with the VHDL statement: **Process**

The key word “after”
has no meaning for
synthesis

....

process (A,B)

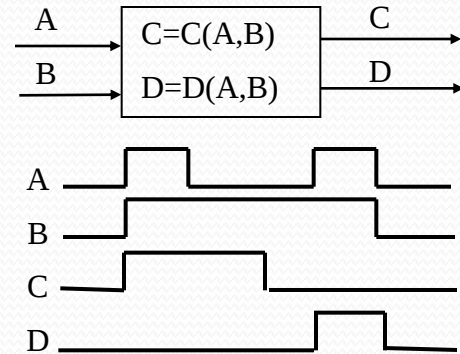
begin

$C \leq A * B$ after 50 ns;

$D \leq A * B * C$ after 100 ns;

end process;

....



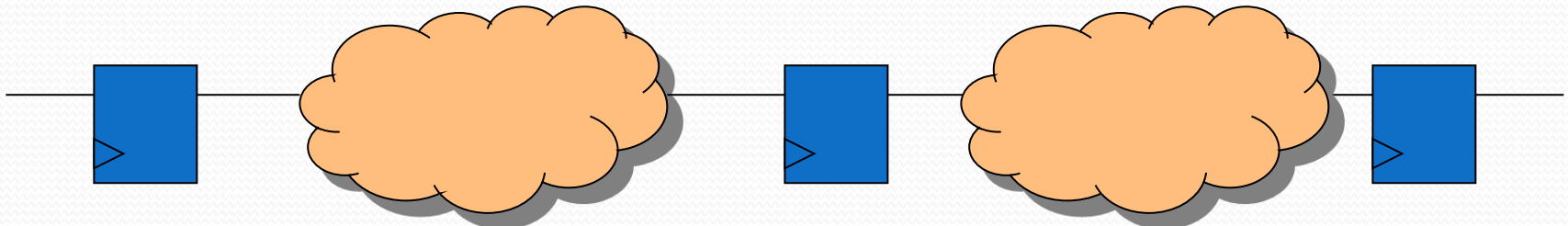
I/O No clock-delayed constraints

VHDL: RTL Description

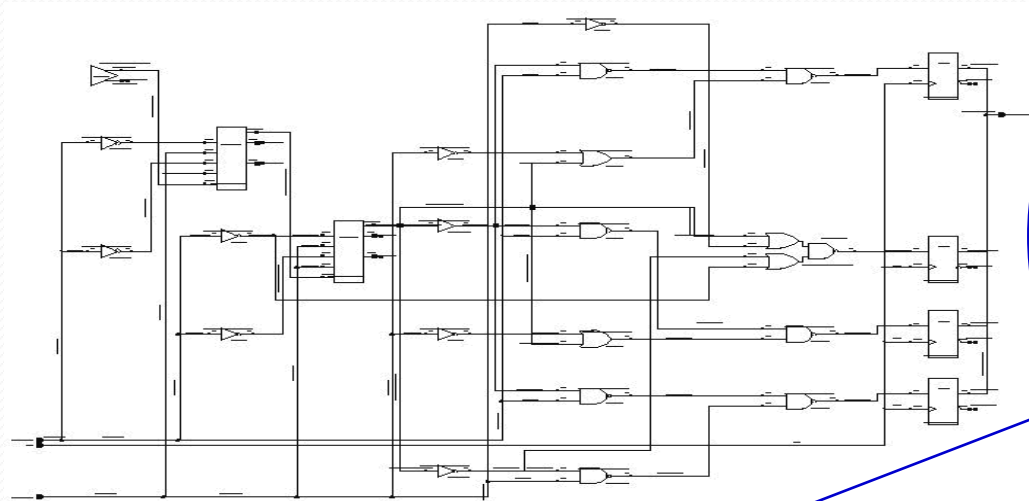
RTL level process descriptions:

Pure combinational: described with high level instructions, such as +, *, MUX ...

Synchronous: clocked described with Flip-Flops.



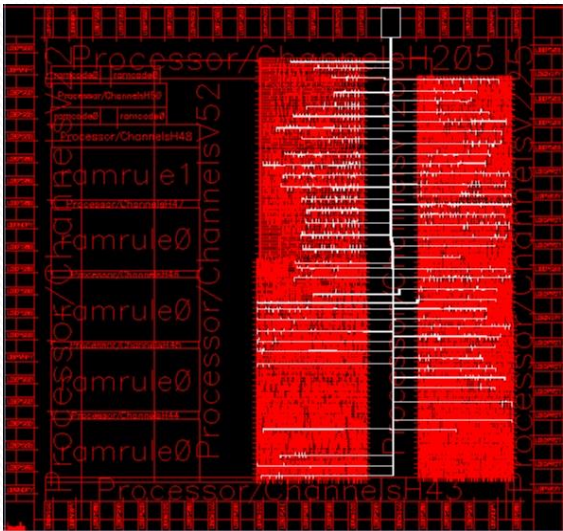
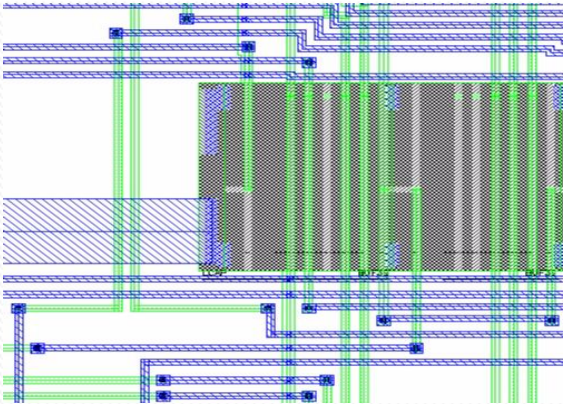
VHDL: Gate Level Description



```
.....  
U13 : H_OR3  
  port map (  
    I0 => net456,  
    I1 => net1801,  
    I2 => net1802,  
    O => net345 );  
.....
```

A gate level description contains a **list of the gates** of the design. It holds the actual instantiation of the components and lists their interconnection. An equivalent schematic of the gate structure is shown. Each element of the circuit (e.g. U13) is instantiated as a component (e.g. H_OR3) and connected to the corresponding signals (net456, net1801, net1802, net345). All used gates are part of the technology library where additional information like area, propagation delay, capacity, etc. is stored. Here delays can be applied to the used gates for simulation and timing information is part of the synthesis library. ***This enables a rough validation of the timing behavior.*** 😊

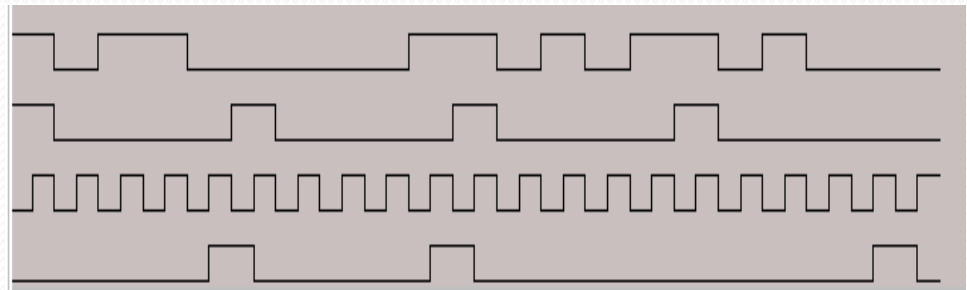
VHDL: Layout Level Description



If the layout is completed, the wire lengths and thus the propagation delays due to parasitics will be known. The design can be simulated on gate level netlist added with propagation delays, after **back-annotation**, and, consequently, **the timing behavior of the entire circuit can be validated**.

The back-annotated delays may make up the main part of the entire delay in larger designs, **especially for very deep sub-micron technologies** ($< 0.35\mu\text{m}$).

Nevertheless the **simulation is fully digital**.



Origin of VHDL

- VHDL originated in the early **1980s**
 - The American Department of Defense initiated the development of VHDL
 - because the US military needed a standardized method of describing electronic systems
- VHDL was standardized in **1987** by the IEEE (Institute of Electrical and Electronics Engineers)
- ANSI Standard in **1988**
- Revised version in 1993
 - IEEE Std-1076-1993
- It is now accepted as one of the most important standard languages for:
 - specifying
 - verifying
 - designing of electronics

VHDL vs. schematic entry

- easier to develop code for who's already used to programming in C or similar languages
- shorter development times for electronic design
- simpler maintenance
- easier to port to other technologies

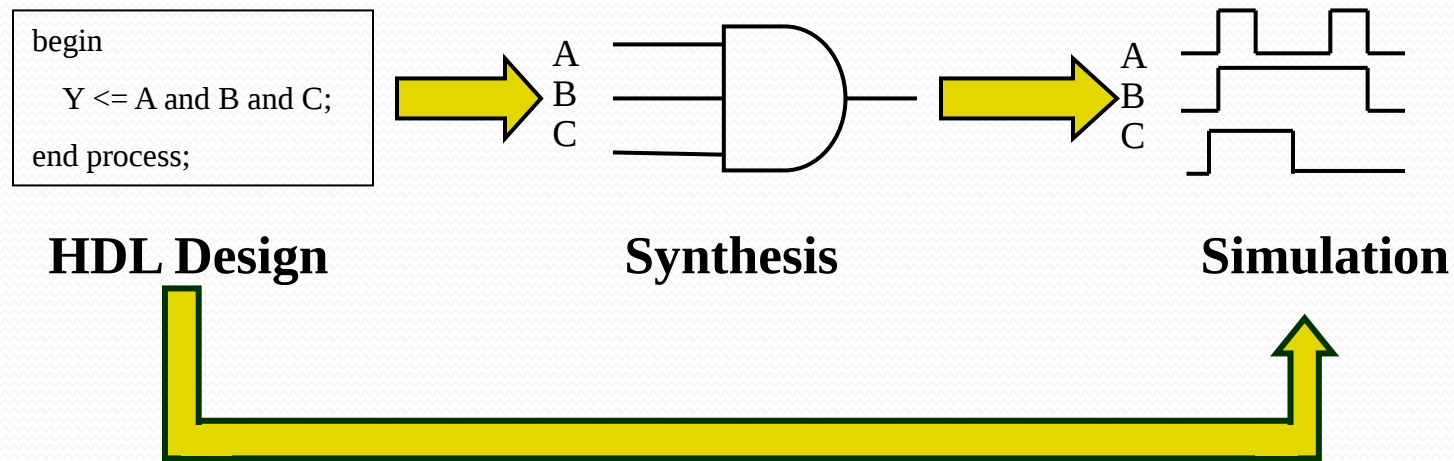
EDA tools

- Good VHDL tools and VHDL simulators have also been developed for PCs
- Prices have fallen dramatically, enabling smaller companies to use VHDL, too
- There are also PC synthesis tools, primarily for FPGAs and EPLDs
- Nowadays
 - to design an ASIC you need a good Linux-based PC;
 - to design a FPGA you need an average Linux/Windows PC.
- Main EDA vendors:
 - Cadence: <http://www.cadence.com>
 - Synopsys: <http://www.synopsys.com>
 - Mentor Graphics: <http://www.mentor.com>

New possibilities

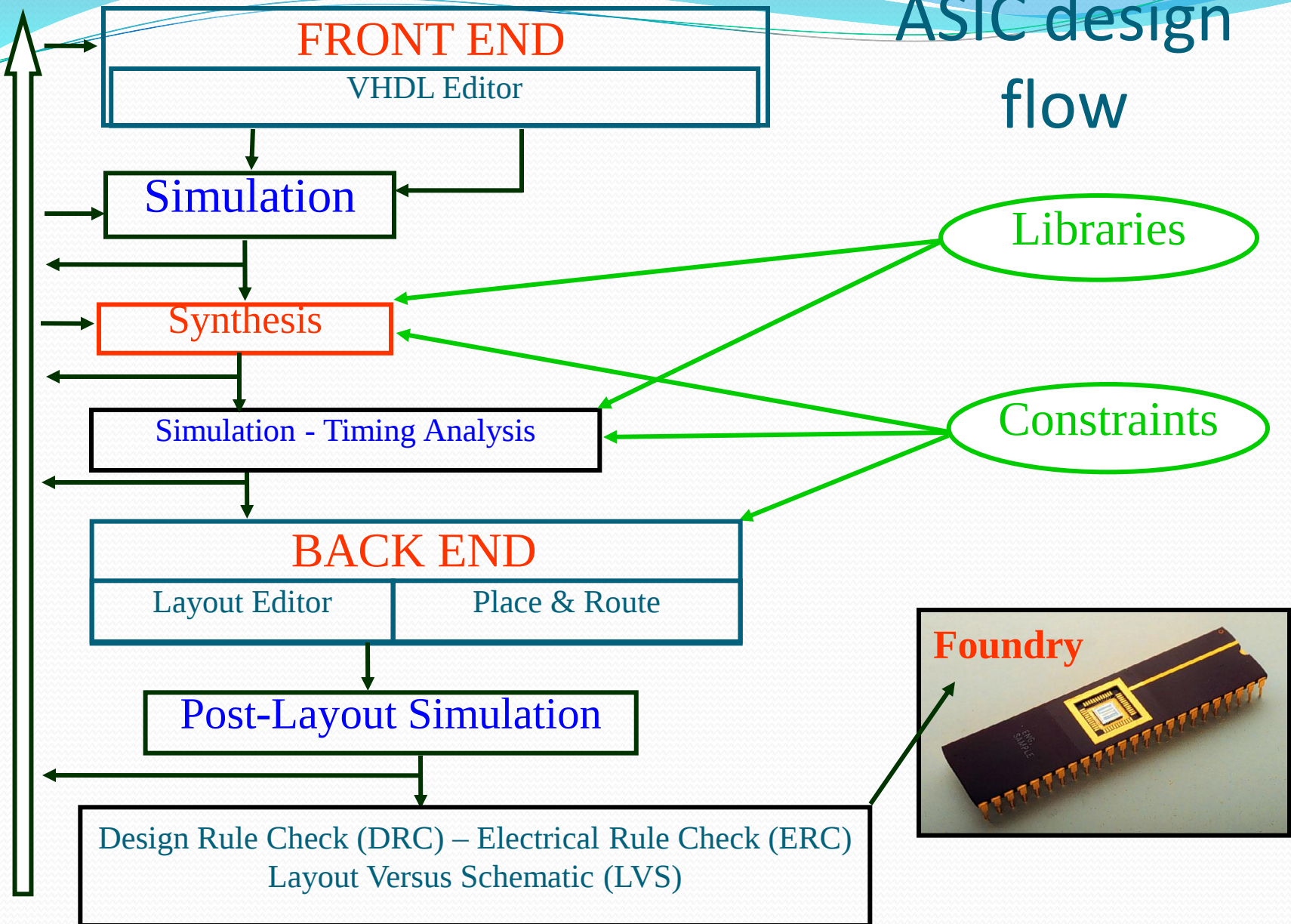
- VHDL frees the designer from having to use von Neumann structures
- It allows him to work with real **concurrency** instead of **sequential** machines
- This opens up completely new possibilities for the designer

VHDL: basic concepts

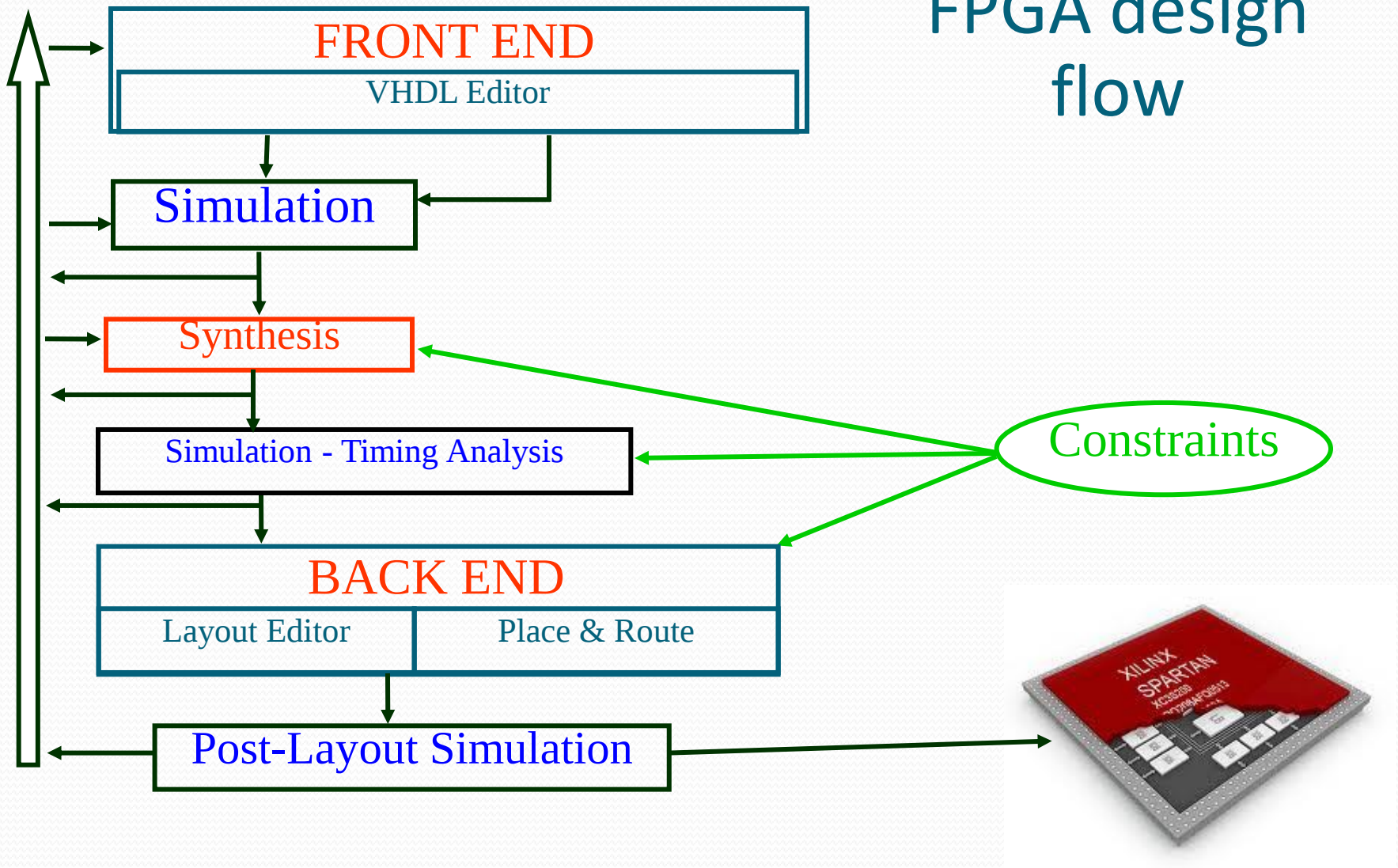


- Possibility to 'execute' the code
- During the development phase the HDL description has to become more and more precise until it is really possible to build the chip
- The (automatic) translation of a less detailed HDL description into a more elaborated one is called SYNTHESIS. By default Synthesis means RTL Synthesis i.e. a translation from RTL to Gate abstraction level

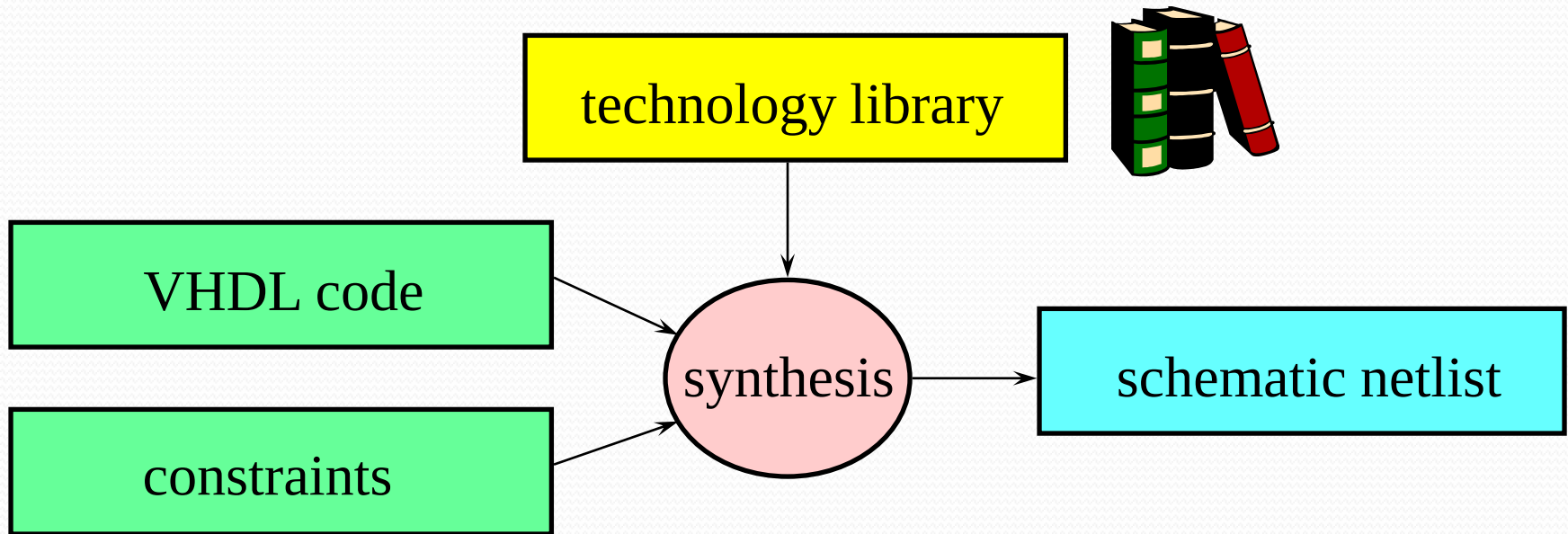
ASIC design flow



FPGA design flow



VHDL: Synthesis

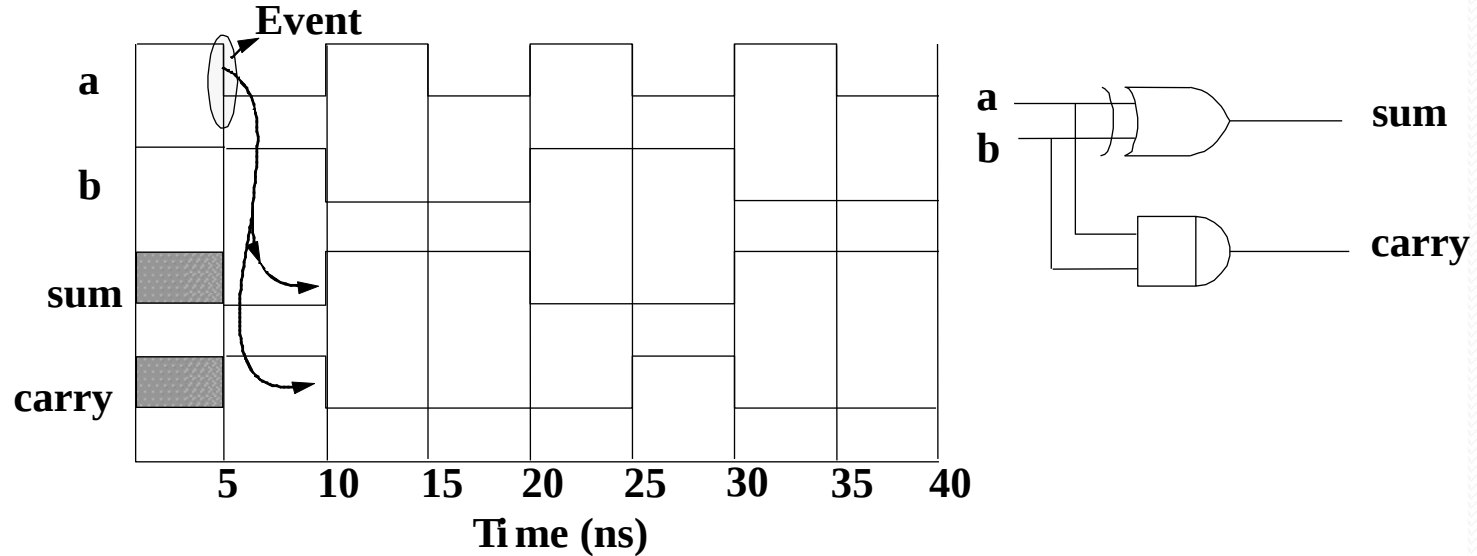


Different schematic can be obtained from the same VHDL code depending on the constraints being applied:

High speed required → large area circuit

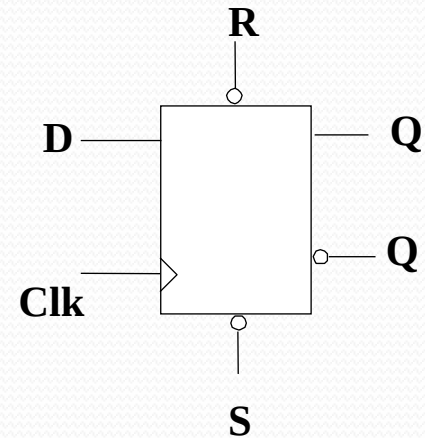
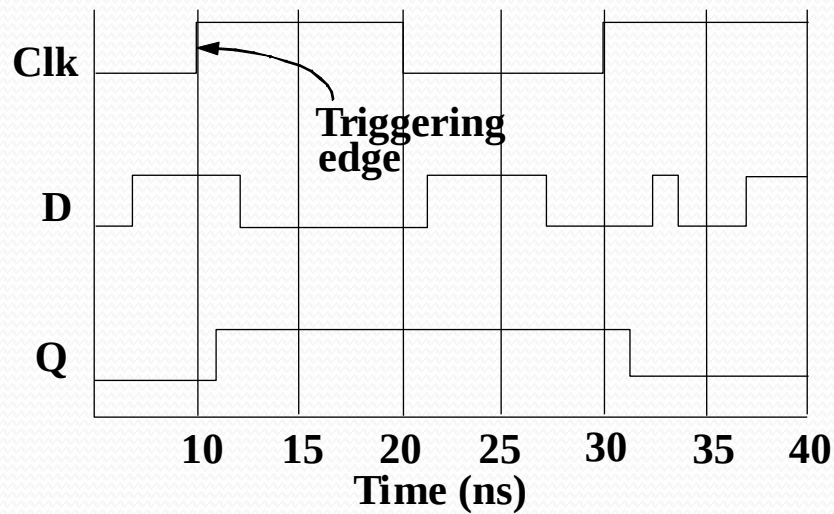
Small area circuit required → slow circuit

VHDL: Simulation



- Signal value changes at specific points in time
- Time ordered sequence of events produces a **waveform**
- Simulation tools allow to simulate both the VHDL code and the schematics.

VHDL: Simulation



- Timing: computation of events takes place at specific points in time
- Need to “wait for” an event: in this case the clock

References

- VHDL Language Reference Manual (anywhere on the Web)
- VHDL Second Edition, Douglas L. Perry (available in lab)
- Slides available at:
<http://www.bo.infn.it/~falchier/teaching.html>