Laboratorio di Progettazione Elettronica

Content:

VHDL Overview

Basic Concepts of digital design

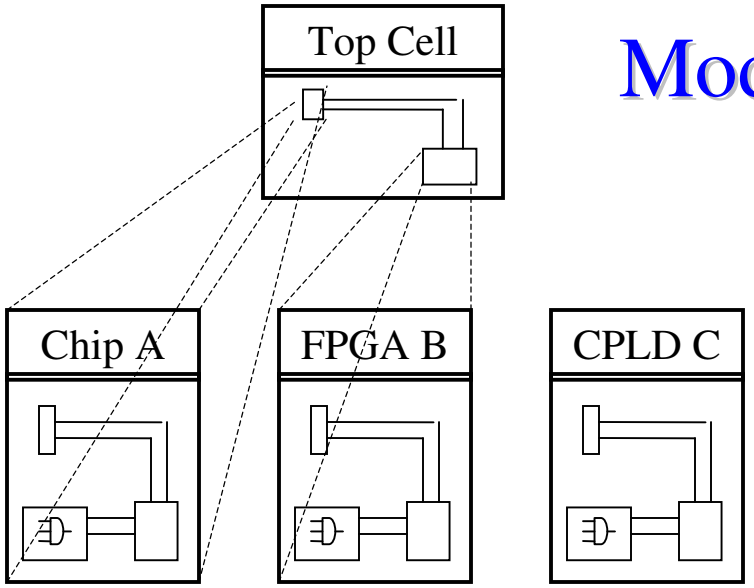
Methodologies: upon designer mixture of abstraction, modularity, hierarchy

Modularity: enables the designer to split big functional blocks and to write a model for each part

Hierarchy: lets the designer build a design out of sub-modules which may consist of several sub-modules, themselves. Each level of hierarchy may contain modules of different abstraction levels

Abstraction: hiding of too detailed information. It is necessary to differentiate between essential and non-essential information. Allows description of different parts of a system with different amount of detail. Modules which are needed only for the simulation do not have to be described as detailed as modules that might be synthesized. *Behavior, RTL, Gate, Layout* are the most common abstraction levels.

Modularity & Hierarchy



- Partitioning in several partial designs
- Restrict complexity
- Enable teamwork
- Study of alternative implementations

1) In this way, a complex system can be divided into more manageable subsystems
2) Different implementation alternatives can be examined for the (sub)modules
3) The existence of well defined subsystems allows several designers to ***work in parallel on the same project***

Why HDL ?

- Schematic entry with the current design complexities is not feasible (though you will find who still loves it)
- HDL allows to move the abstraction level up, in order to ease the design of complex designs.
- Software tools allow to easily move from one abstraction level to an other one.
- The HDL code can be easily simulated.
- Last (but not least): the HDL code is technology independent. The implications are …

VHDL: History

- early `70s: Initial discussions by american Department of Defense (Dod)
- late `70s: Definition of requirements
- mid `82s: Contract of development with IBM, Intermetrics and TI
- mid `84s: Version 7.2
- mid `86s: IEEE-Standard – First Standard
- 1987: DoD adopts the standard → IEEE.1076
- mid `88s: Increasing support by Computer Aided Engineer (CAE) manufacturers
- late `91s: Revision
- 1993: New standard – Official first update
- 1999: VHDL-AMS (Analog Mixed Signal) – Latest official upgrade
- The standard should be revised every 5 years

VHDL: Application Fields

Hardware design

- **ASIC** (Application Specific Integrated Circuit)
- **PASIC** (Programmable ASIC; only once)
- **FPGA** (Field Programmable Gate Array) – Xilinx, Altera
- **CPLD** (Complex Programmable Logic Device) – Xilinx, Altera
- *No DSP so far (Digital Signal Processor/Processing)*

Software design

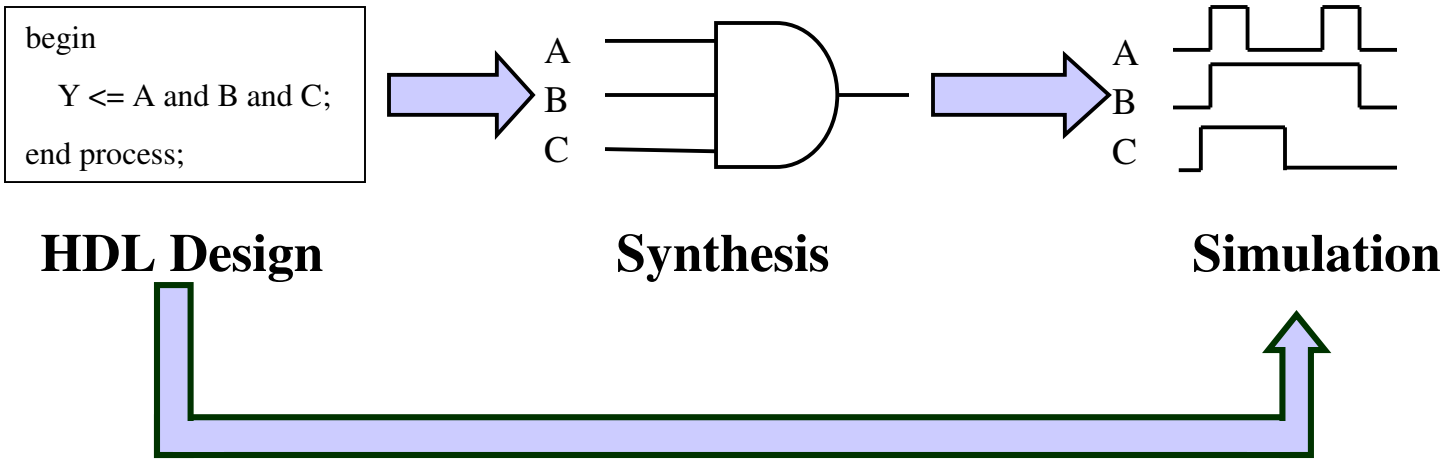
- VHDL - C interface (tool-specific)
- Main focus of research (hardware/software co-design)

An ongoing field of research is the hardware/software co-design

VHDL: Basic Concepts

- **VHDL**:(Very high speed integrated circuit Hardware Description Language)
  modeling of digital systems - concurrent and sequential statements -
  machine readable specification
  man-and machine-readable documentation
- International Standards
  IEEE Std 1076-1987, IEEE Std 1076-1993,
  Analogue and Mixed-Signal extension: VHDL-AMS,IEEE Std 1076.1-1999
- Pure definition of language in the LRM (Language Reference Manual)
- VHDL is very suitable for the design phases from system level to gate level
- Last VHDL upgrade with Analogue and Mixed Signal language elements (VHDL-AMS). The digital VHDL have not been altered by the extension
- Only simulation is feasible for the analogue part because analogue synthesis is a very complex problem affected by many boundary conditions. The mixed signal simulation has to deal with the problem of synchronizing the digital and analogue simulators, which has not been solved adequately, yet (from the synthesis point of view). ☹
- Every transition from and to a different HDL abstraction level must be proven by a functional validation (simulation)
- Analogue and digital full-custom designs made by hand, particularly for layouts

HDLs: 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

VHDL: Abstraction Levels

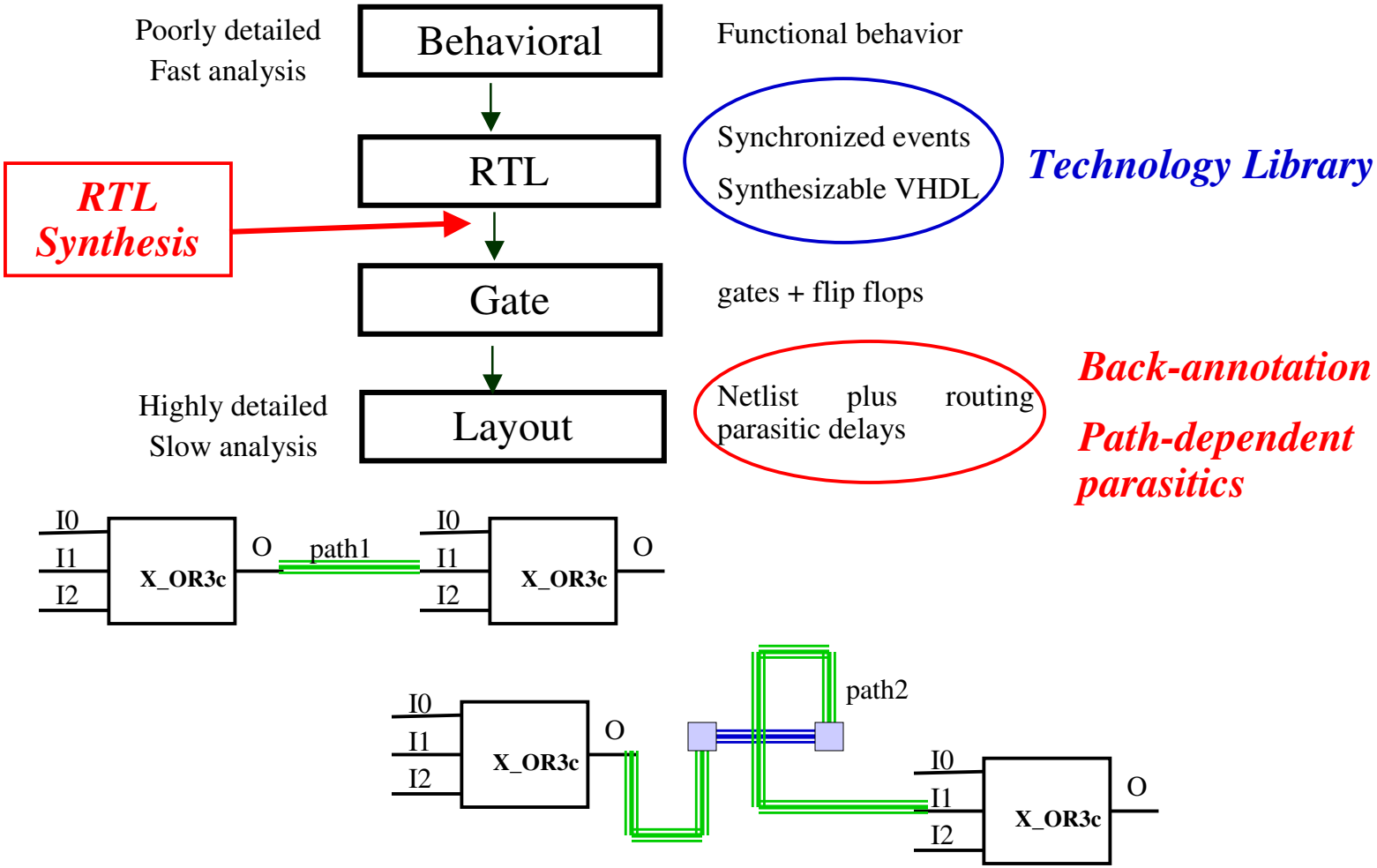
BEHAVIORAL: functional description of the model. Used at the very beginning stage of a design in order to be a able to run a simulation as soon as possible. Also used to describe testbenches. Such descriptions are usually simulatable, but not synthesizable.

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.

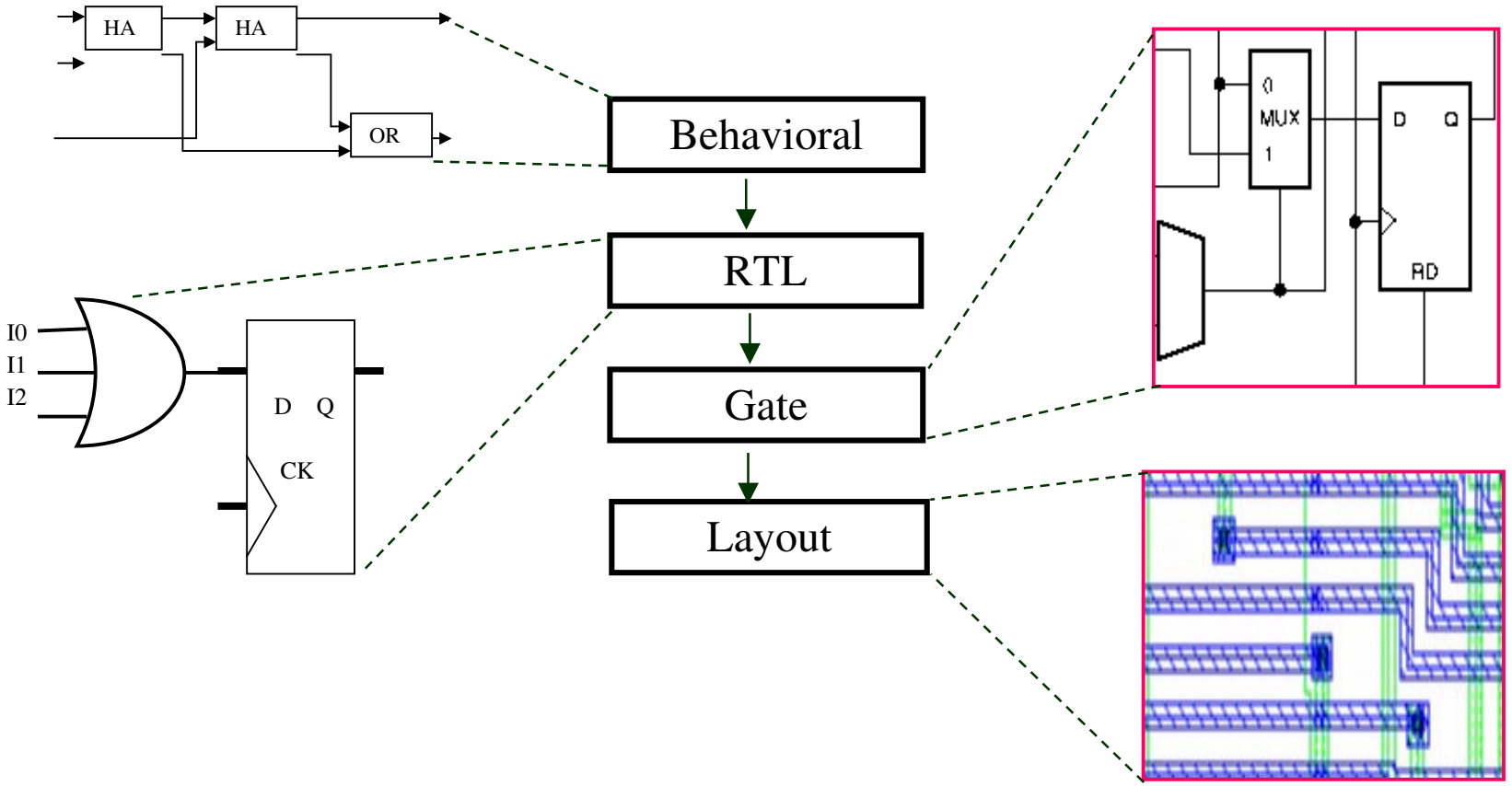
GATE: the design is represented as a netlist with gates (AND, OR, NOT, ...) and storage elements, all with cell delays. The description has been synthesized.

LAYOUT: the different cells of the target technology are placed on the chip and the connections are routed (back-annotated additional delays). After the layout has been verified, the circuit is ready for the production process.

VHDL: Abstraction Levels



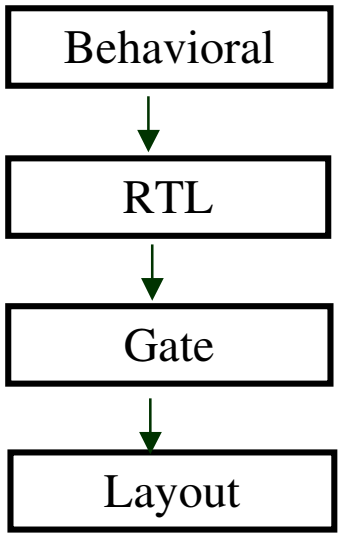
VHDL: Abstraction Levels



VHDL: Abstraction Levels

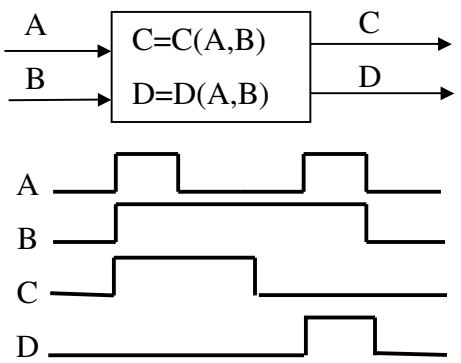
VHDL is suitable for describing the upper 3 abstraction levels. It is not suitable to design a layout.

The design entry in **BEHAVIORAL** and **RTL** level is usually done by means of text editors. *Behavioral synthesis is still a dream* of many researchers as only very simple behavior models are synthesizable. The last step from the **GATE** (**LOGIC**) level to the final **LAYOUT** has been widely *automated for digital standard cell designs* by means of an automated process named Place & Route.



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***



I/O No clock-delayed constraints

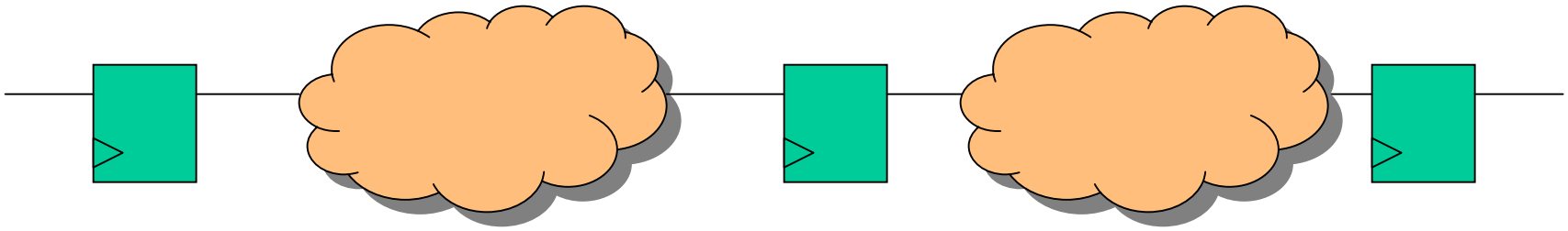
The key word "after" has no meaning for synthesis

```
….
process (A,B)  - - sensitivity list
begin
        C <= C(A,B) after 50 ns;
        D <= D(A,B) after 100 ns;
end process;
…..
```

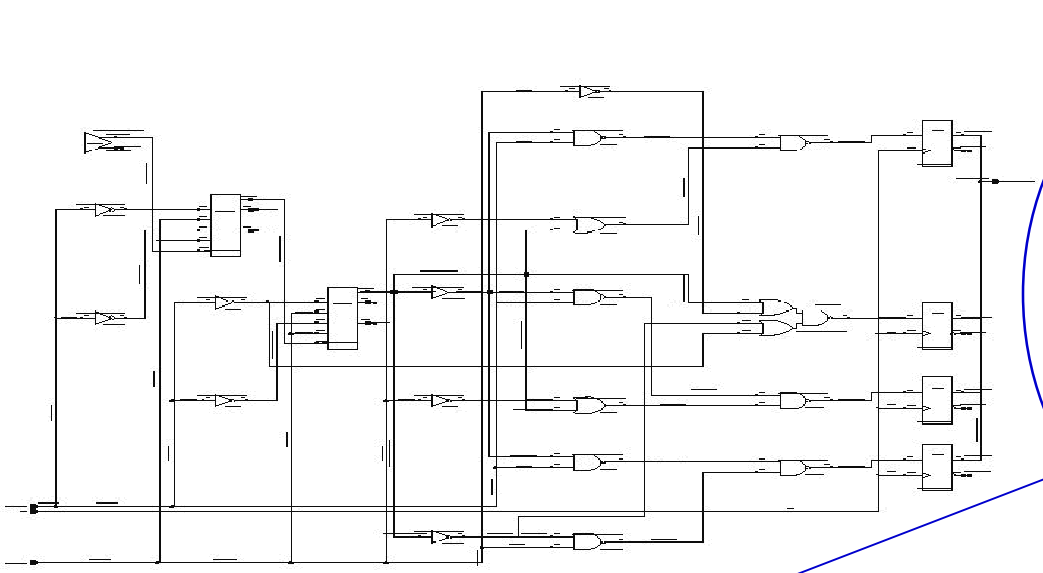
VHDL: RTL Description

RTL level process descriptions:

- ***Pure combinational:*** described with high level instructions, such as +, *, MUX …
- ***Synchronous:*** clocked described with Flip-Flops.



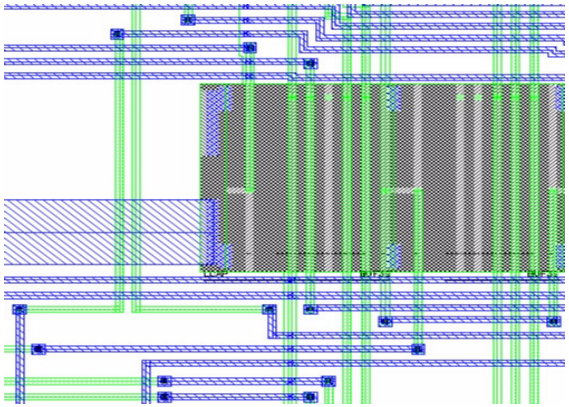
VHDL: Gate Level Description

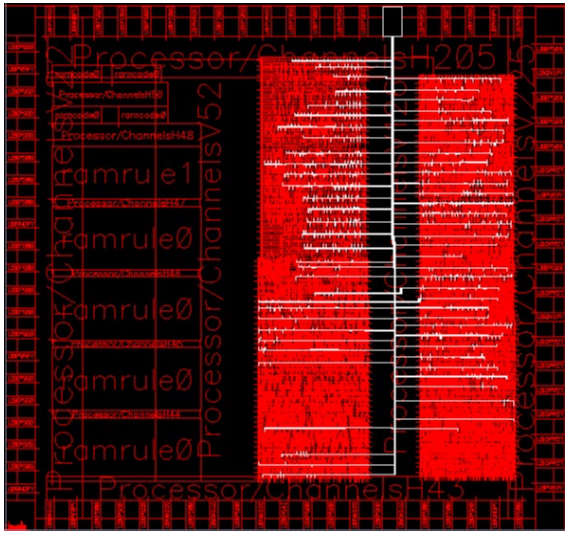


```
.......
U13 : H_OR3
   port map (
     I0 => net456,
     I1 => net1801,
     I2 => net1802,
     O => net345 );
.......
```

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

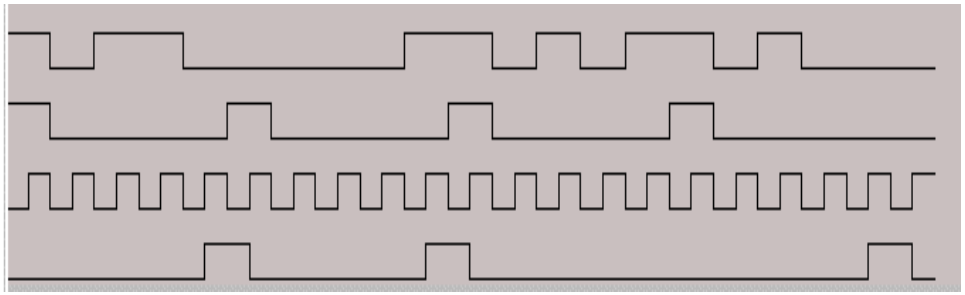




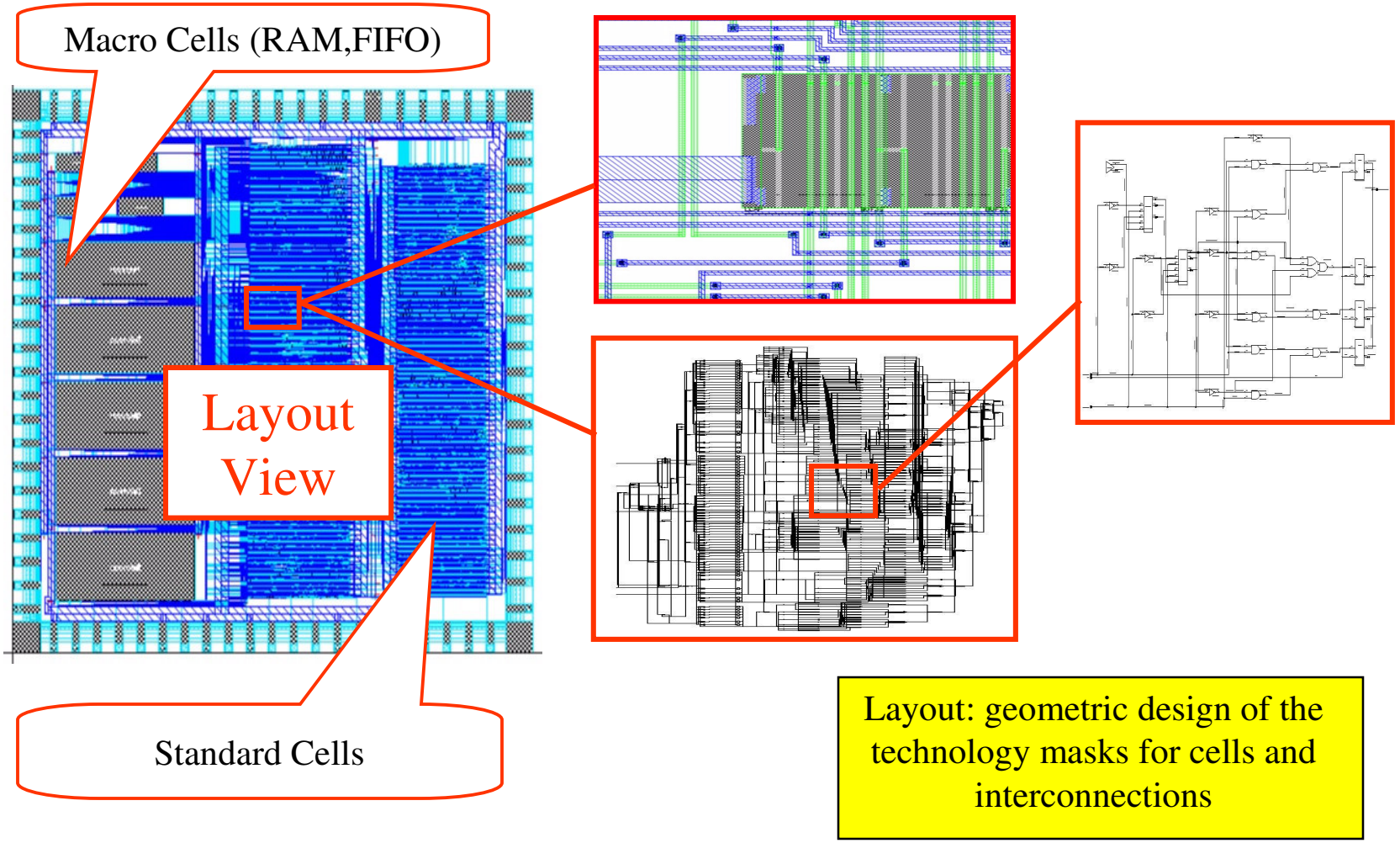
If the layout is completed, the wire lengths and thus the propagation delays due to parasitics will be known. The design can be simulated on gate level netlist added with propagation delays, after ***back-annotation***, and, consequently, ***the timing behavior of the entire circuit can be validated***.

The back-annotated delays may make up the main part of the entire delay in larger designs, ***especially for very deep sub-micron technologies*** (< 0.35µm).

Nevertheless the ***simulation is fully digital.***

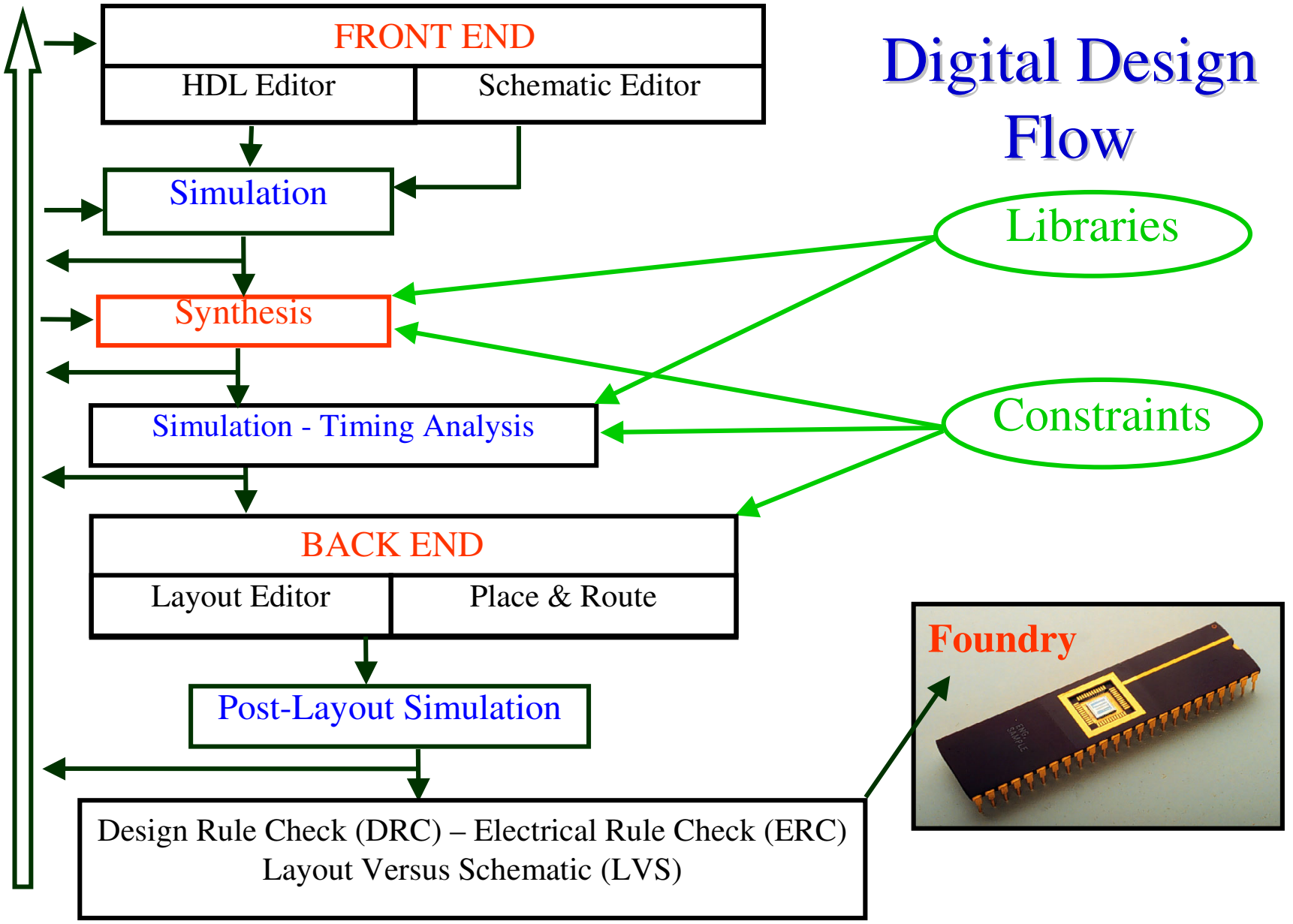


Digital ASIC Development with VHDL



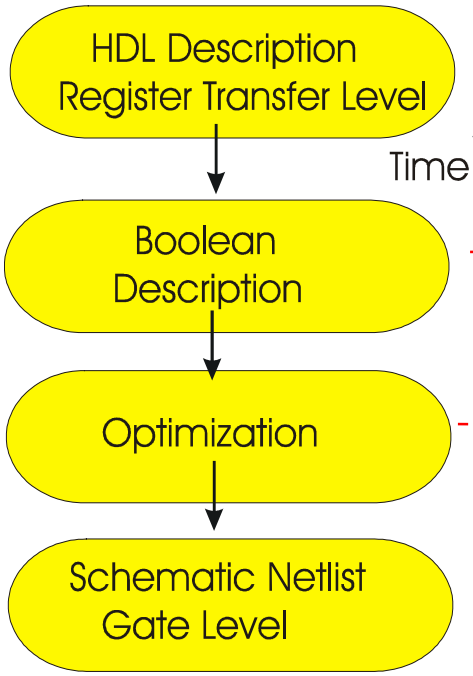
Layout: geometric design of the technology masks for cells and interconnections

Digital Design Flow

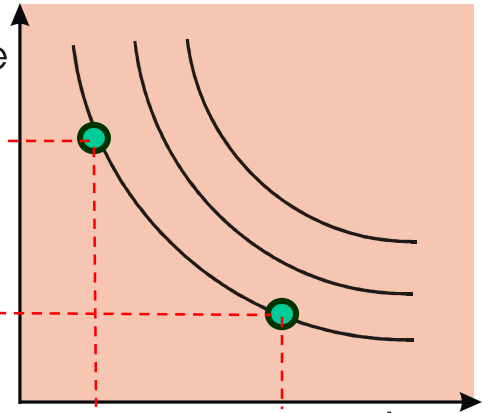


HDLs: Synthesis

- **Synthesis: it is a translation of a VHDL abstraction level to a lower, more detailed, one**
- ☹ Only a subset of HDL statements are synthesizable
- Synthesis programs extract the (non-generally unique) Boolean functions from the HDL model, then map it onto the elements of an ASIC gate library or a programmable device such as an FPGA.



Two different gate-level solution depending on constraints for synthesis

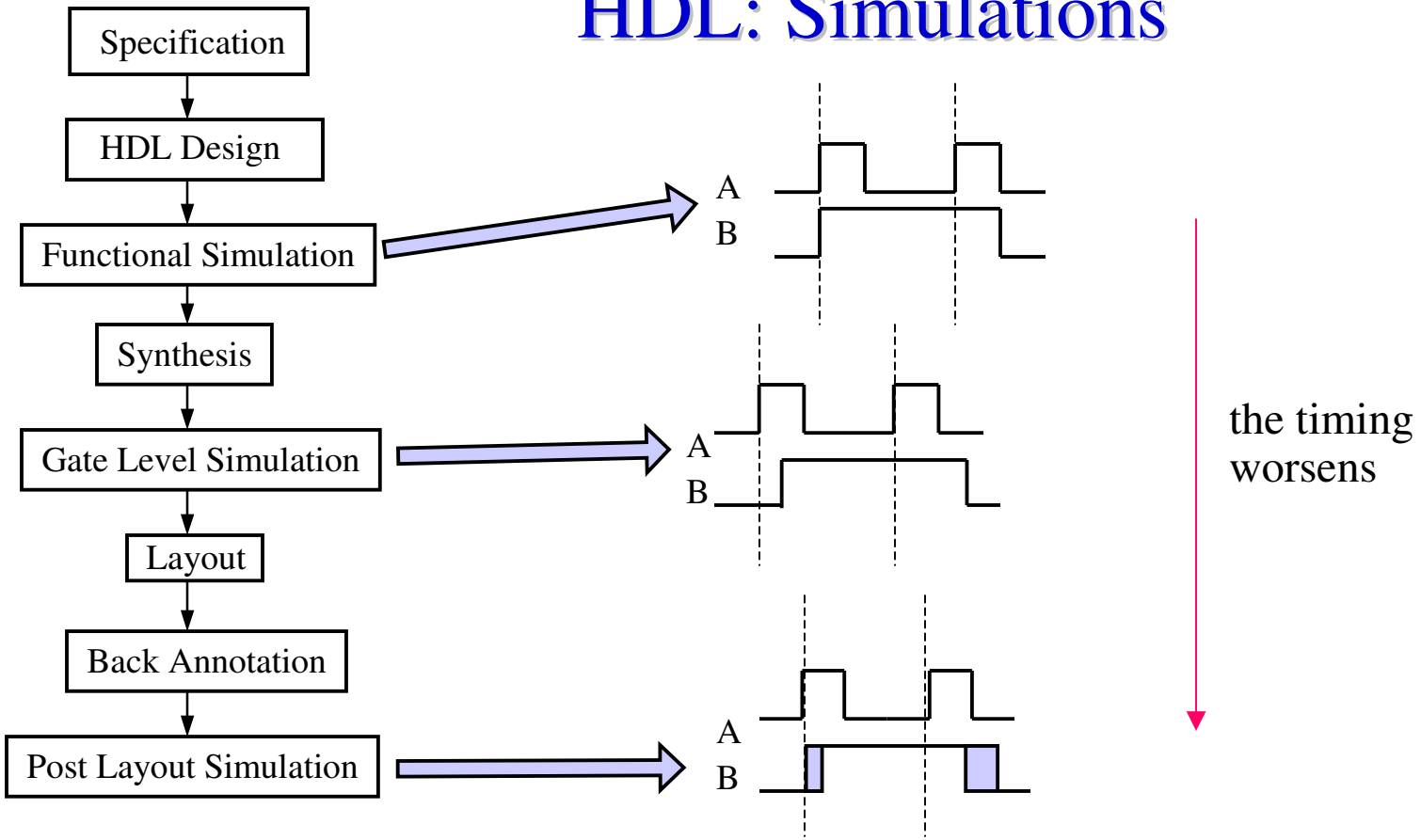


Cost Function minimization

$\text{Time}^{\alpha} * \text{Area}^{\beta}$

- The SYNTHESIS result is a NETLIST of the module on the gate level
- Then the circuit LAYOUT for an ASIC technology can be created by means of other tools from the NETLIST description.

HDL: Simulations



- *Specification covers functional aspects and timing behavior*
- *If the model shows the desired behavior the VHDL code may be synthesized; but there is no guarantee the synthesis succeeds* ☹

Digital ASIC Development with VHDL

- Synthesis selects the appropriate standard cells (Boolean gates, FF, etc.) from the ASIC library in order to reproduce the functional description/behavior. The library holds the information about all available gates and their parameters such as fan-in, fan-out, delay, etc.
- The whole gate delays along the longest paths, from the output to the input of every Flip Flop, must be less than the defined clock period
- As soon as a model built (by the foundry) of ASIC library elements is available, a simulation on gate level can be performed (pre-layout). *Here the designer has the first idea about maximum clock frequency and critical paths.* ☺
- Then the LAYOUT design may be started.
- The propagation delay along the signal wires have to be estimated first because the real values are available after the LAYOUT is finished. The process of feeding these values back into the VHDL model is called *back annotation. Once again the circuit must be checked, whether it fulfills the specified timing constraints.*

Vantaggi della progettazione con il linguaggio VHDL

- linguaggio di programmazione con istruzioni concorrenti,
- trasportabilità del progetto verso:
  - librerie tecnologiche diverse,
  - moduli all'interno di altri progetti,
- verifica della correttezza funzionale e circuitale,
- minor tempo di progettazione.

Summary

1) There is a clear distinction between a ***pure behavioral*** model and ***RTL level*** modeling for synthesis
2) VHDL permits a structural (***modular***) and ***hierarchical description*** of a digital system
3) It is possible to describe ***timing behavior*** in VHDL
4) A simulatable behavioral design can be non synthesizable
5) ***Back-annotation*** adds more delays due to interconnections

Questions

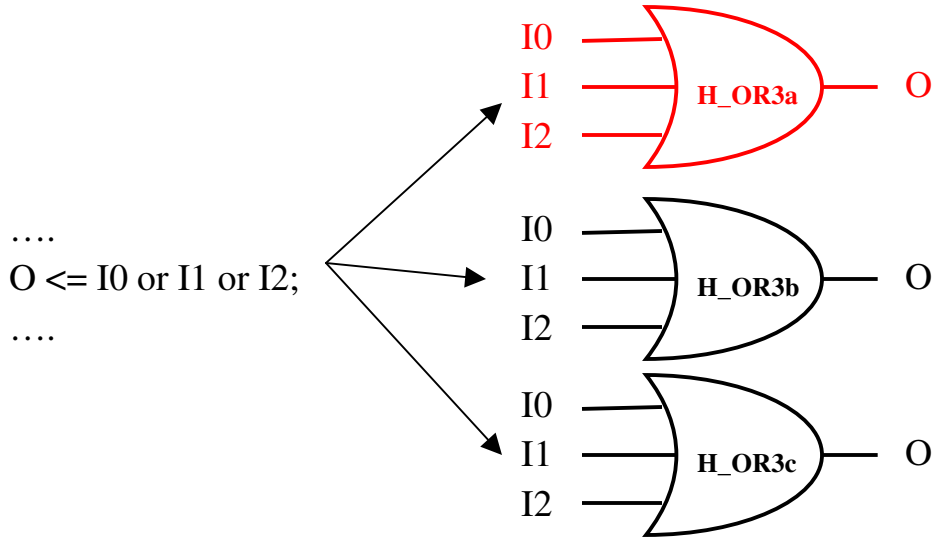
1) For what VHDL is mainly used ?
2) Main differences among other programming languages?
3) What is an FPGA?
4) What is an ASIC?
5) What is the difference between modularity and hierarchy?
6) What is the abstraction level?
7) What is the synthesis process?
8) What is a netlist?
9) When a VHDL code may be not synthesizable?
10) What is a layout view?
11) What is the back-annotation?

HDLs: Instantiation

After a VHDL model has been synthesized a specific netlist of instantiated components is created. This way not only a logic gate is implemented for a given task but also a given physical gate with specific features in terms of delays, fan-out, etc. A specific technology library may have several different gates with the same logic functionality but different physical condition.

In the example the instantiated component is not just a 3-inputs OR but it is the H_OR3a.

```
…….
U13 : H_OR3a
   port map (
     I0 => net456,
     I1 => net1801,
     I2 => net1802,
     O => net345);
…….
```



VHDL: Sequential & Concurrent Statements

Execution of assignments:

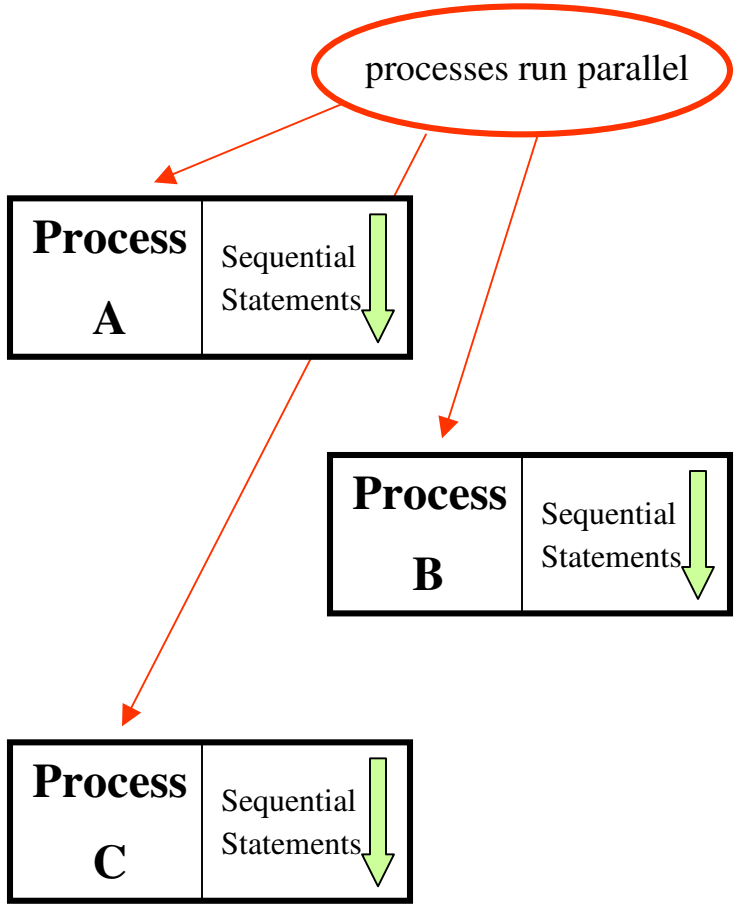
- **Sequential**
- **Concurrent**

Sequential Statements

- Executed sequentially
- Successive statements can override the effects of previous ones
- Significant order of the assignment

Concurrent Statements

- Active continuously
- No matter of the statement order
- Good to model parallel circuits



VHDL: Sequential & Concurrent Statements

```
architecture rtl of shift is
signal y0,y1,y2: std_logic_vector(1 downto 0);
begin

   c <= a + b;                    – Concurrent Statement
   d <= a & b(3 downto 0);        – Concurrent Statement

   run0:process(ck,a,y0)          – Concurrent Structure
   begin
        if(ck'event and ck='1')then -- Sequential statement
                  y0 <= a;          -- Sequential statement
                  y1 <= y0;         -- Sequential statement
         end if;
   end process;
   run1:process(y1,y2)            – Concurrent Structure
   begin
                  y2 <= y1;         -- Sequential statement
                  y   <= y2;        -- Sequential statement
    end process;
end rtl;
```