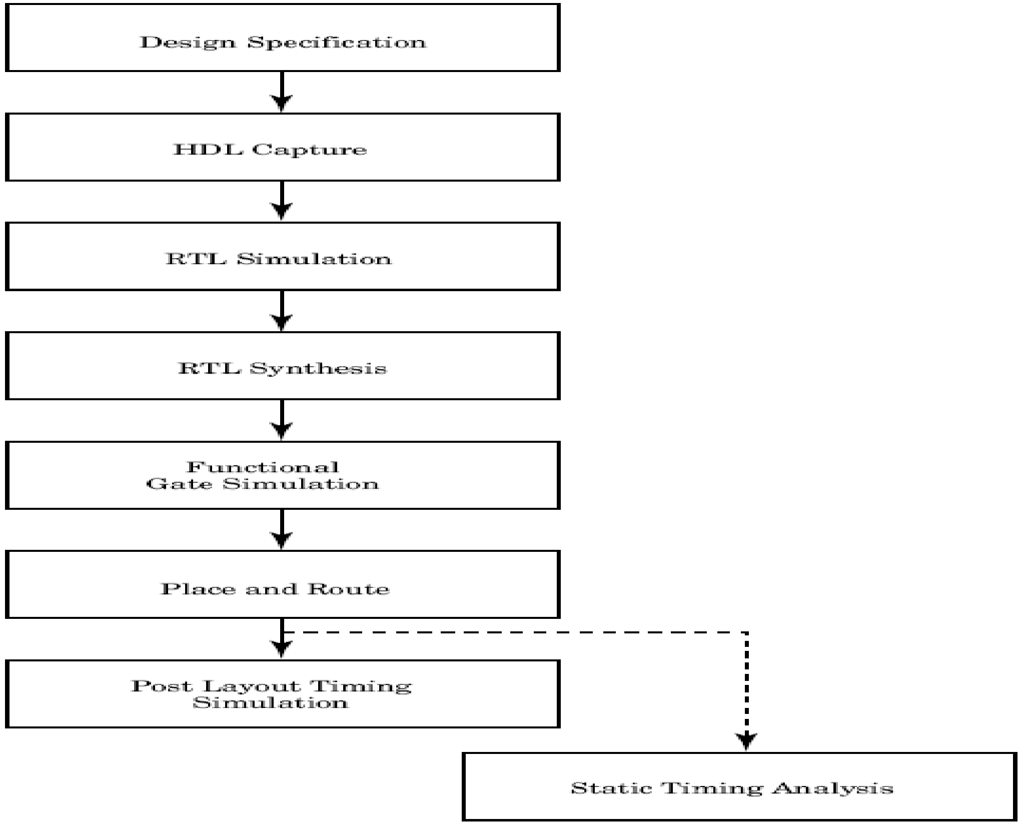
UNIVERSITÀ
DEGLI STUDI DI TRIESTE

dipartimento
di ingegneria
e architettura

15 - High level design flow

A.Carini – Progettazione di sistemi elettronici

High level design flow



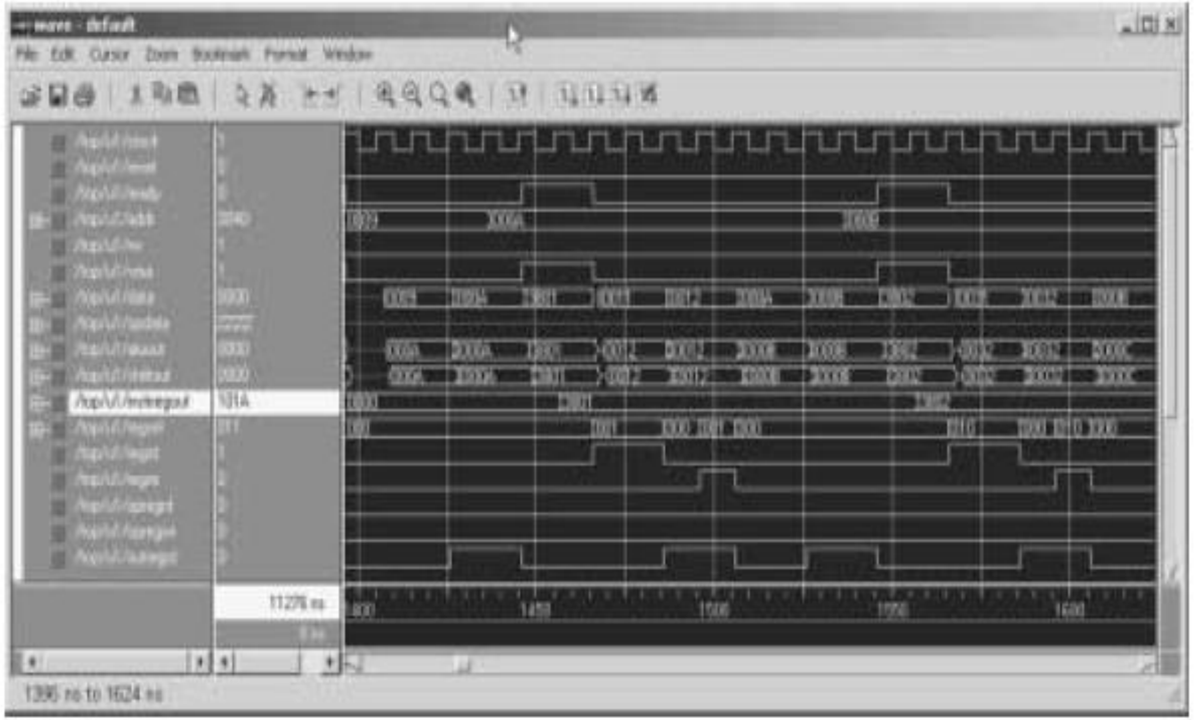
From design specifications to RTL model

- The specifications must specify what the system does and must give the information necessary for building the design.
- The specifications are typically written in the language of the designer.
- Sometimes the specifications are given in terms of a behavioral HDL model.
- More often, it is the designer himself that writes a behavioral model in order to verify that the specifications are clearly stated.
- Then, the designer writes a synthesizable RLT description of the system.

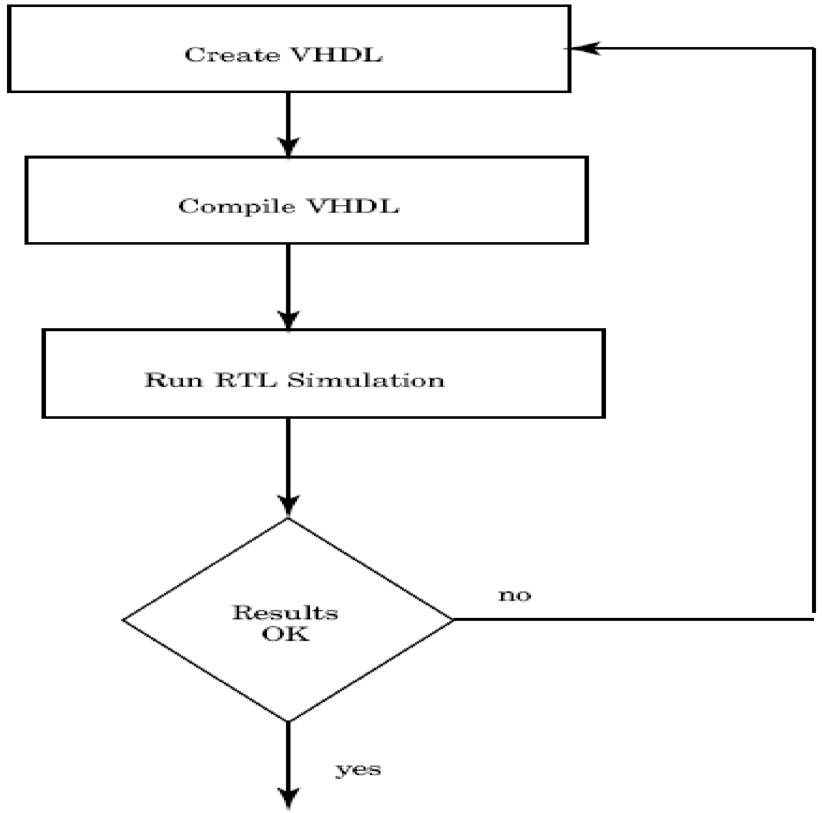
RTL simulation

- After removing all syntax and semantic errors, the designer verifies the code by means of an RTL simulation.
- The designer uses appropriate test benches for driving the VHDL model and for checking that the results are correct.
- The VHDL description is analyzed, elaborated and executed by a VHDL simulator.
- The simulator output data, provided in a textual form or a waveform form, are used for verifying the design correctness and for finding possible errors.
- In case of errors, the designer modifies the VHDL code, repeats the analysis and elaboration, and the RTL simulation for checking the system.

RTL simulation



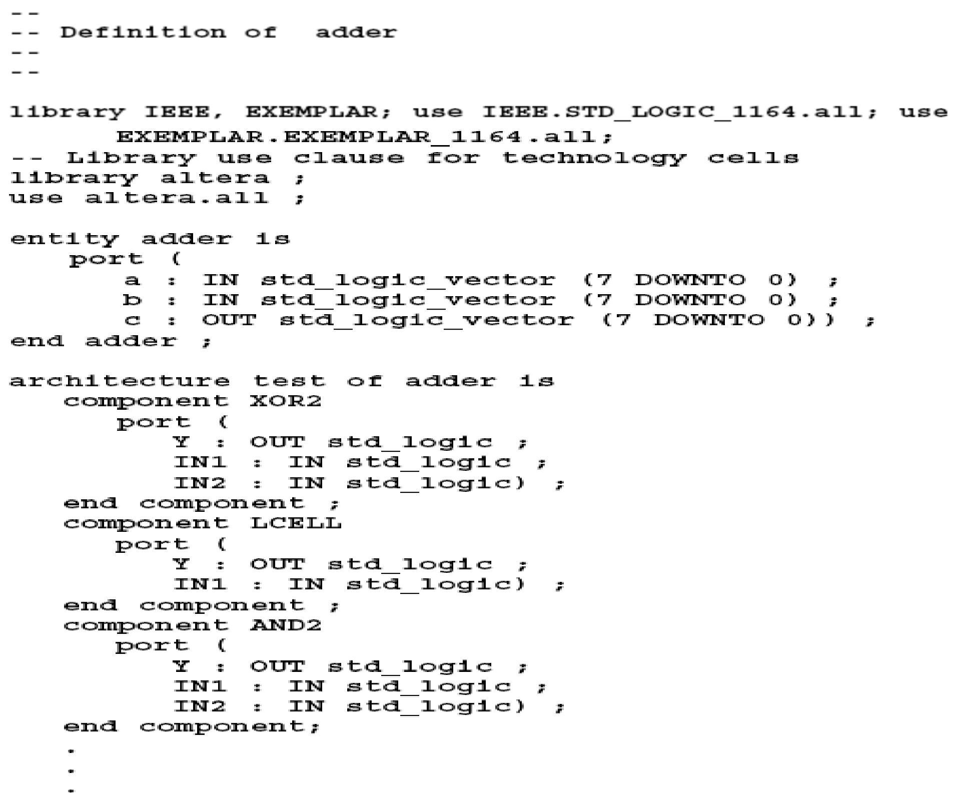
RTL simulation



VHDL netlist

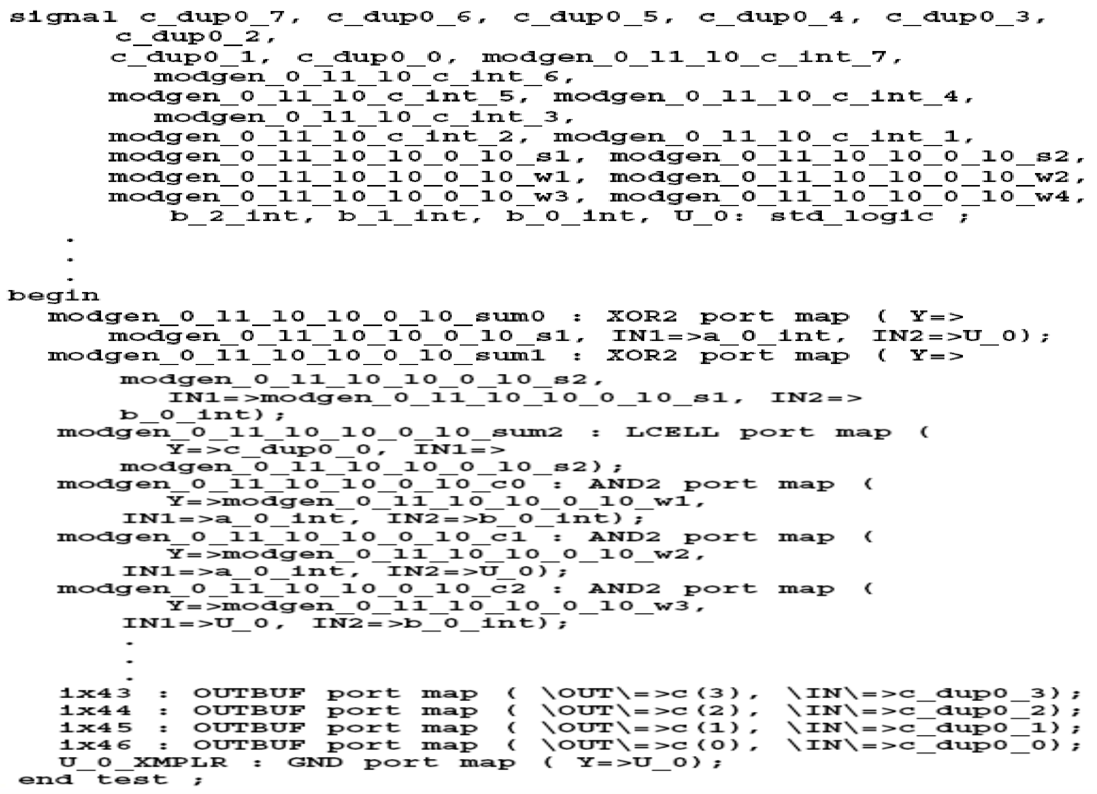
- The objective of the VHDL synthesis is that of creating a design that implements the requested functionalities and that meets the designer’s constraints in terms of area, speed, etc.
- The VHDL synthesis tools translates an RTL VHDL description into a netlist composed by cells of the target technology.
- Possible errors in this phase are usually due to the use of non-synthesizable constructs.
- At the end of the synthesis process, the synthesizer produces a netlist and a certain number of report files.
- The designer can inspect the netlist, but principally will use the report files for determining the quality of the output synthesis.
- The principal information that we find in these report files is the timing information and area occupation information.
- The area occupation information is used from the designer for determining how many resources are utilized by the design.
- The timing reports provides information about the critical paths.
- The designer can also inspect the netlist, that sometimes may assume the form of a VHDL code.

VHDL netlist



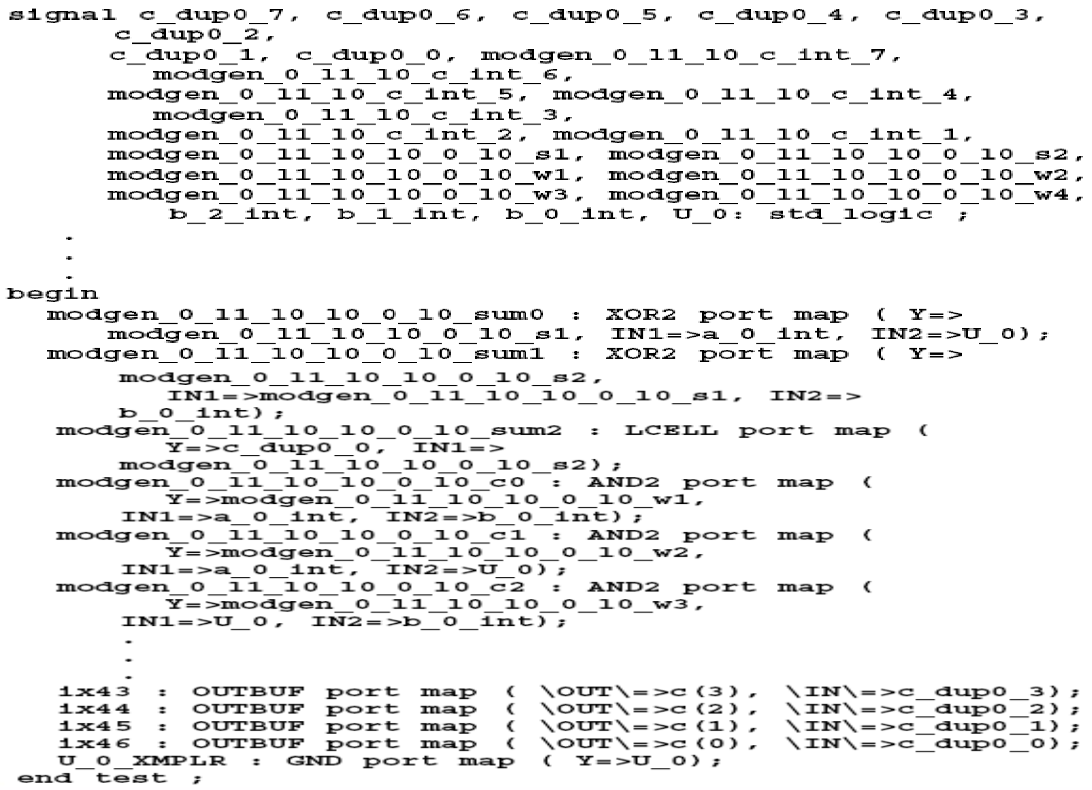
```
--
-- Definition of  adder
--
--

library IEEE, EXEMPLAR; use IEEE.STD_LOGIC_1164.all; use
        EXEMPLAR.EXEMPLAR_1164.all;
-- Library use clause for technology cells
library altera ;
use altera.all ;

entity adder is
    port (
        a : IN std_logic_vector (7 DOWNTO 0)  ;
        b : IN std_logic_vector (7 DOWNTO 0)  ;
        c : OUT std_logic_vector (7 DOWNTO 0))  ;
end adder ;

architecture test of adder is
    component XOR2
        port (
            Y : OUT std_logic ;
            IN1 : IN std_logic ;
            IN2 : IN std_logic)  ;
    end component ;
    component LCELL
        port (
            Y : OUT std_logic ;
            IN1 : IN std_logic)  ;
    end component ;
    component AND2
        port (
            Y : OUT std_logic ;
            IN1 : IN std_logic ;
            IN2 : IN std_logic)  ;
    end component;
    .
    .
    .
```

VHDL netlist



```
signal c_dup0_7, c_dup0_6, c_dup0_5, c_dup0_4, c_dup0_3,
      c_dup0_2,
      c_dup0_1, c_dup0_0, modgen_0_l1_l0_c_int_7,
         modgen_0_l1_l0_c_int_6,
      modgen_0_l1_l0_c_int_5, modgen_0_l1_l0_c_int_4,
         modgen_0_l1_l0_c_int_3,
      modgen_0_l1_l0_c_int_2, modgen_0_l1_l0_c_int_1,
      modgen_0_l1_l0_l0_0_l0_s1, modgen_0_l1_l0_l0_0_l0_s2,
      modgen_0_l1_l0_l0_0_l0_w1, modgen_0_l1_l0_l0_0_l0_w2,
      modgen_0_l1_l0_l0_0_l0_w3, modgen_0_l1_l0_l0_0_l0_w4,
          b_2_int, b_1_int, b_0_int, U_0: std_logic ;
    .
    .
    .
begin
  modgen_0_l1_l0_l0_0_l0_sum0 : XOR2 port map ( Y=>
      modgen_0_l1_l0_l0_0_l0_s1, IN1=>a_0_int, IN2=>U_0);
  modgen_0_l1_l0_l0_0_l0_sum1 : XOR2 port map ( Y=>
       modgen_0_l1_l0_l0_0_l0_s2,
          IN1=>modgen_0_l1_l0_l0_0_l0_s1, IN2=>
       b_0_int);
   modgen_0_l1_l0_l0_0_l0_sum2 : LCELL port map (
          Y=>c_dup0_0, IN1=>
       modgen_0_l1_l0_l0_0_l0_s2);
   modgen_0_l1_l0_l0_0_l0_c0 : AND2 port map (
          Y=>modgen_0_l1_l0_l0_0_l0_w1,
       IN1=>a_0_int, IN2=>b_0_int);
   modgen_0_l1_l0_l0_0_l0_c1 : AND2 port map (
          Y=>modgen_0_l1_l0_l0_0_l0_w2,
       IN1=>a_0_int, IN2=>U_0);
   modgen_0_l1_l0_l0_0_l0_c2 : AND2 port map (
          Y=>modgen_0_l1_l0_l0_0_l0_w3,
       IN1=>U_0, IN2=>b_0_int);
       .
       .
       .
   ix43 : OUTBUF port map ( \OUT\=>c(3), \IN\=>c_dup0_3);
   ix44 : OUTBUF port map ( \OUT\=>c(2), \IN\=>c_dup0_2);
   ix45 : OUTBUF port map ( \OUT\=>c(1), \IN\=>c_dup0_1);
   ix46 : OUTBUF port map ( \OUT\=>c(0), \IN\=>c_dup0_0);
   U_0_XMPLR : GND port map ( Y=>U_0);
 end test ;
```

VHDL netlist



```
signal c_dup0_7, c_dup0_6, c_dup0_5, c_dup0_4, c_dup0_3,
      c_dup0_2,
      c_dup0_1, c_dup0_0, modgen_0_11_10_c_int_7,
         modgen_0_11_10_c_int_6,
      modgen_0_11_10_c_int_5, modgen_0_11_10_c_int_4,
         modgen_0_11_10_c_int_3,
      modgen_0_11_10_c_int_2, modgen_0_11_10_c_int_1,
      modgen_0_11_10_10_0_10_s1, modgen_0_11_10_10_0_10_s2,
      modgen_0_11_10_10_0_10_w1, modgen_0_11_10_10_0_10_w2,
      modgen_0_11_10_10_0_10_w3, modgen_0_11_10_10_0_10_w4,
          b_2_int, b_1_int, b_0_int, U_0: std_logic ;
    .
    .
    .
begin
   modgen_0_11_10_10_0_10_sum0 : XOR2 port map ( Y=>
       modgen_0_11_10_10_0_10_s1, IN1=>a_0_int, IN2=>U_0);
   modgen_0_11_10_10_0_10_sum1 : XOR2 port map ( Y=>
        modgen_0_11_10_10_0_10_s2,
           IN1=>modgen_0_11_10_10_0_10_s1, IN2=>
        b_0_int);
    modgen_0_11_10_10_0_10_sum2 : LCELL port map (
           Y=>c_dup0_0, IN1=>
        modgen_0_11_10_10_0_10_s2);
    modgen_0_11_10_10_0_10_c0 : AND2 port map (
           Y=>modgen_0_11_10_10_0_10_w1,
        IN1=>a_0_int, IN2=>b_0_int);
    modgen_0_11_10_10_0_10_c1 : AND2 port map (
           Y=>modgen_0_11_10_10_0_10_w2,
        IN1=>a_0_int, IN2=>U_0);
    modgen_0_11_10_10_0_10_c2 : AND2 port map (
           Y=>modgen_0_11_10_10_0_10_w3,
        IN1=>U_0, IN2=>b_0_int);
        .
        .
        .
    ix43 : OUTBUF port map ( \OUT\=>c(3), \IN\=>c_dup0_3);
    ix44 : OUTBUF port map ( \OUT\=>c(2), \IN\=>c_dup0_2);
    ix45 : OUTBUF port map ( \OUT\=>c(1), \IN\=>c_dup0_1);
    ix46 : OUTBUF port map ( \OUT\=>c(0), \IN\=>c_dup0_0);
    U_0_XMPLR : GND port map ( Y=>U_0);
 end test ;
```

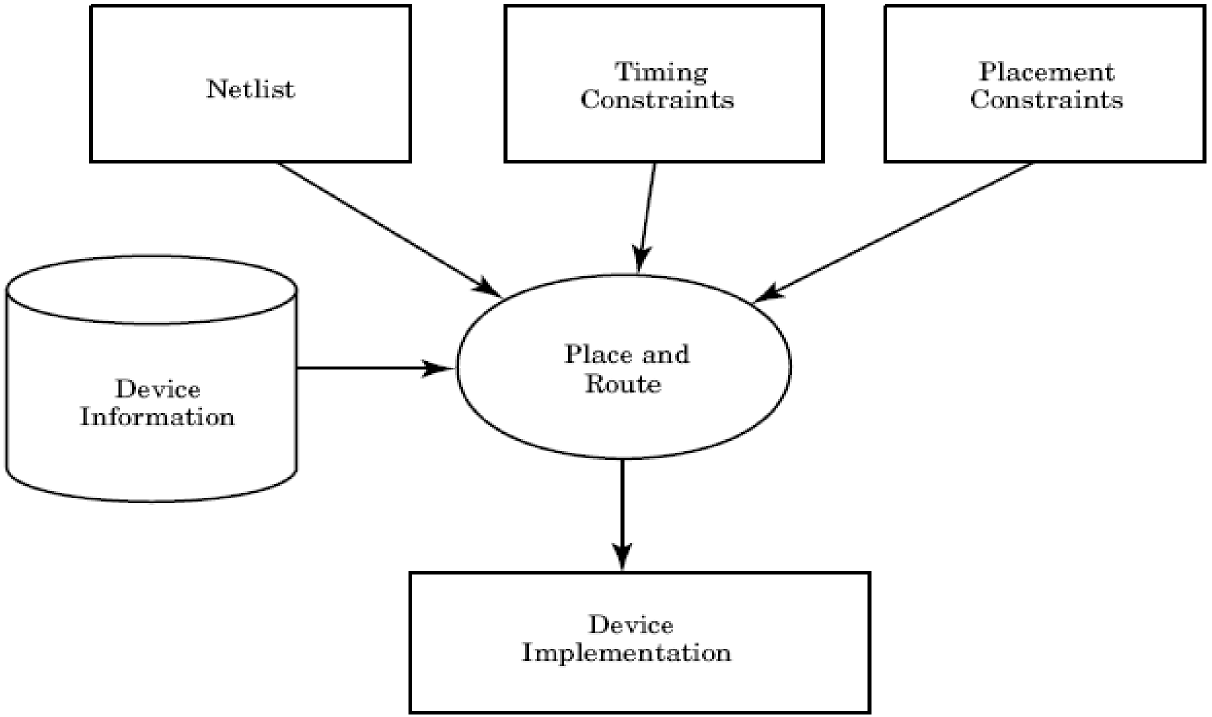
Gate level functional verification

- Some designers may want to verify rapidly the synthesis output, to be sure that the synthesized system is correct.
- A part from the case of tool bugs, the synthesized system should provide the same functional behavior of the RTL model.
- For verifying the code, the designer executes the *gate level functional verification*, i.e., he simulates the netlist together with a library of synthesis primitives, using the same test benches of the RTL verification.
- If the simulation provides the same results as before, the synthesis has hopefully not introduced any error; if the results are different, the designer will have to find the reason of the incongruence.

Place and routing

- The place and route tools are used for mapping the synthesized netlist on the target device.
- They place each netlist primitive in an appropriate position on the target device and they route the signals between the primitives for connecting the devices according to the netlist.
- The place and route tools are typically strongly dependent on the device architecture. These tools are developed for taking advantage of every possible architectural or routing benefit allowed by the device.
- The inputs of the place and route operation are the netlist and the timing constraints.

Place and routing



Post layout timing simulation

- After the place and route process is complete, the designer verifies the result obtained.
- The most common verification method is the gate level post layout timing simulation.
- This simulation combines the netlist used for place and routing, with the timing file obtained from the place and route tool.
- The simulation checks both the functionality and timing performance and verify that the specifications are met.
- Typically, the same test bench of the RTL simulation is used also for the post layout timing simulation.

Static timing analysis

- For small projects, the simulation is a good method for verifying the functionality and the timing performance of the output system.
- For larger projects, the designer can use the *static timing analysis* for verifying that the timing requirements are met.
- A static timing analyzer traces each path in the design and computes the corresponding propagation delay.
- It outputs a report with the propagation delay of every path or of the most critical paths.

See:

- Douglas L. Perry, «VHDL programming by example» McGraw Hill,
  - Chapter 11