Universidad
Carlos III de Madrid

Circuit Simulation of digital circuits with VHDL

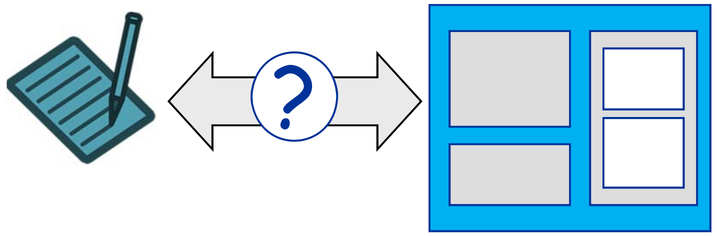
TEST BENCHES AND SIMULATION TOOLS

Outline

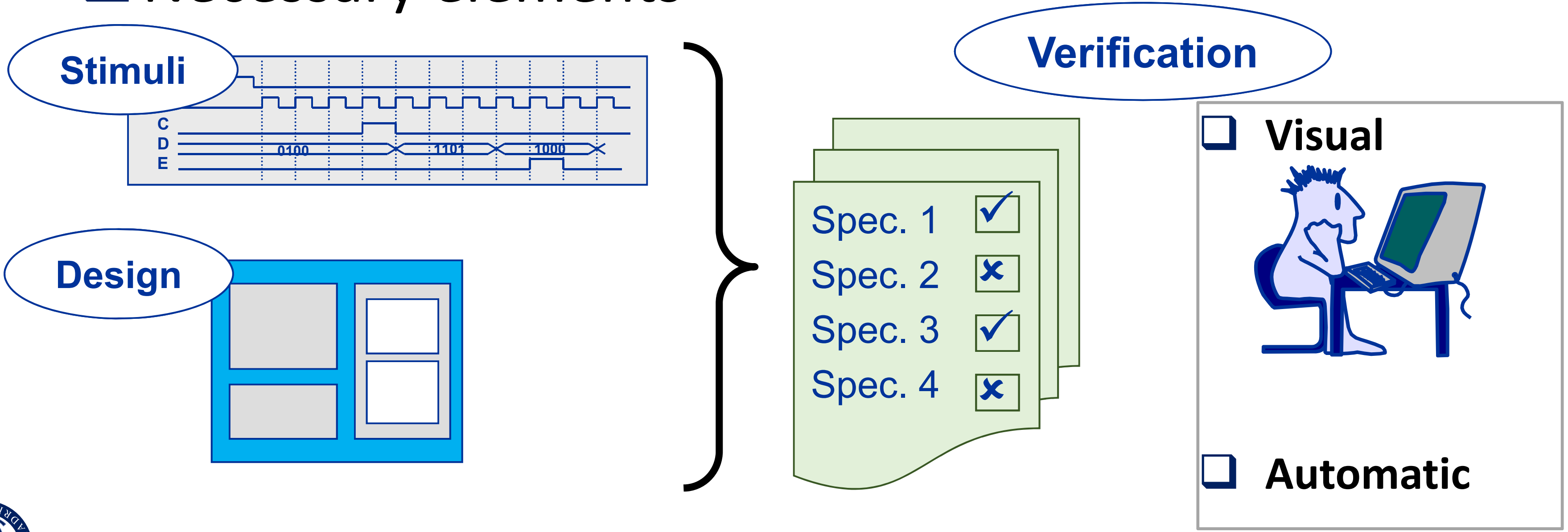
- Digital circuit simulation
- Simulating with VHDL
- Test bench design
  - Generating stimuli
  - Automatic checking
- Commercial tools: Modelsim

Digital circuit simulation

- Functional validation: Verify that the design behaves according to the specifications.

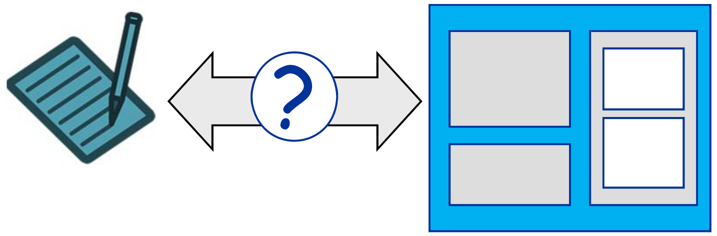


- Necessary elements

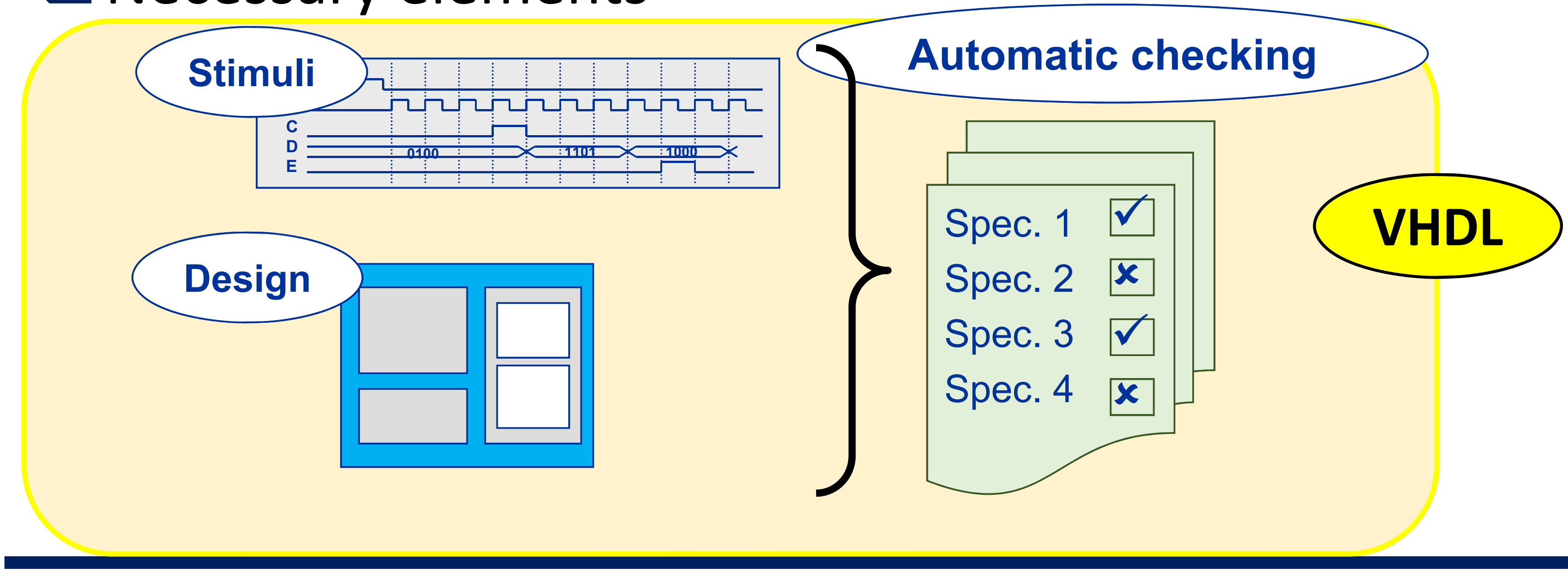


Digital circuit simulation

- Functional validation: Verify that the design behaves according to the specifications.



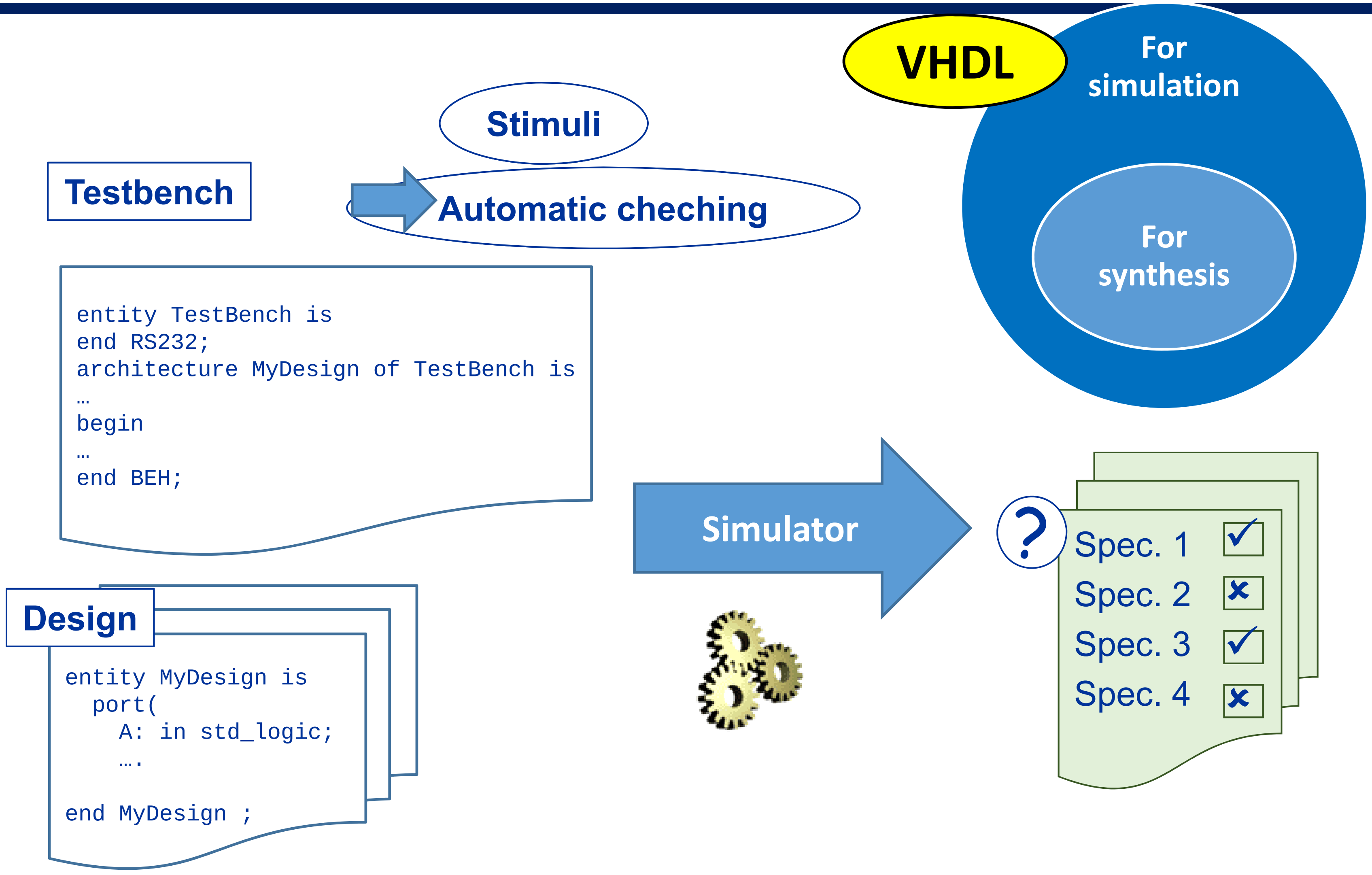
- Necessary elements



Digital circuit simulation: General issues

- Generate a set of inputs (stimuli) as much complete as possible
  - Checking all the specifications to meet by the circuit.
  - In case of FSMs (Finite State Machines), the circuit must pass through every possible state and cover any possible transition.
- Asynchronous initialization of the full system at the beginning of the circuit simulation
- Inputs should change at **inactive** clock edge.

Digital circuit simulation with VHDL



```
entity TestBench is
end RS232;
architecture MyDesign of TestBench is
…
begin
…
end BEH;
```

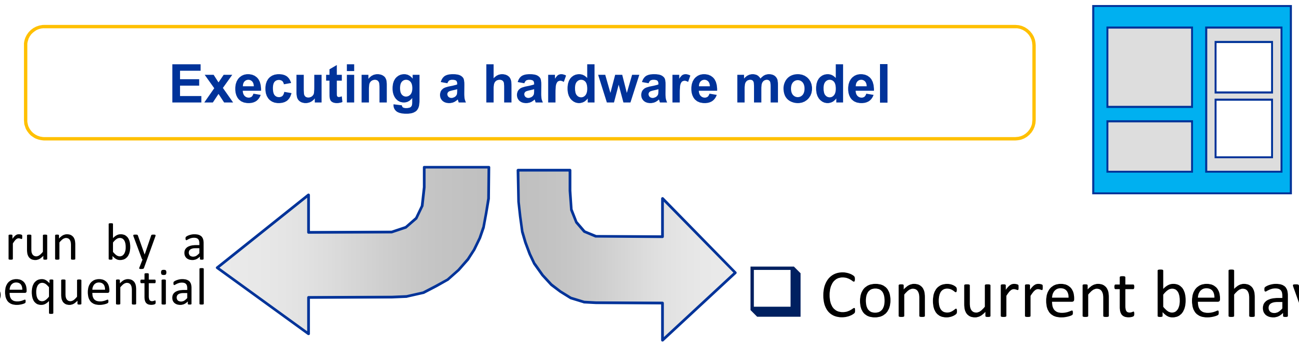
```
entity MyDesign is
  port(
    A: in std_logic;
    ….

end MyDesign ;
```

Digital circuit simulation with VHDL

- Advantages of using VHDL for simulation
  - VHDL was devised for simulating and specifying designs.
  - It supports automatic checking and sending information to the designer during the simulation (interactive checking)
  - It allows designers to model external interfaces at high level: modelling the external environment
  - Using files
  - Using delays and timings

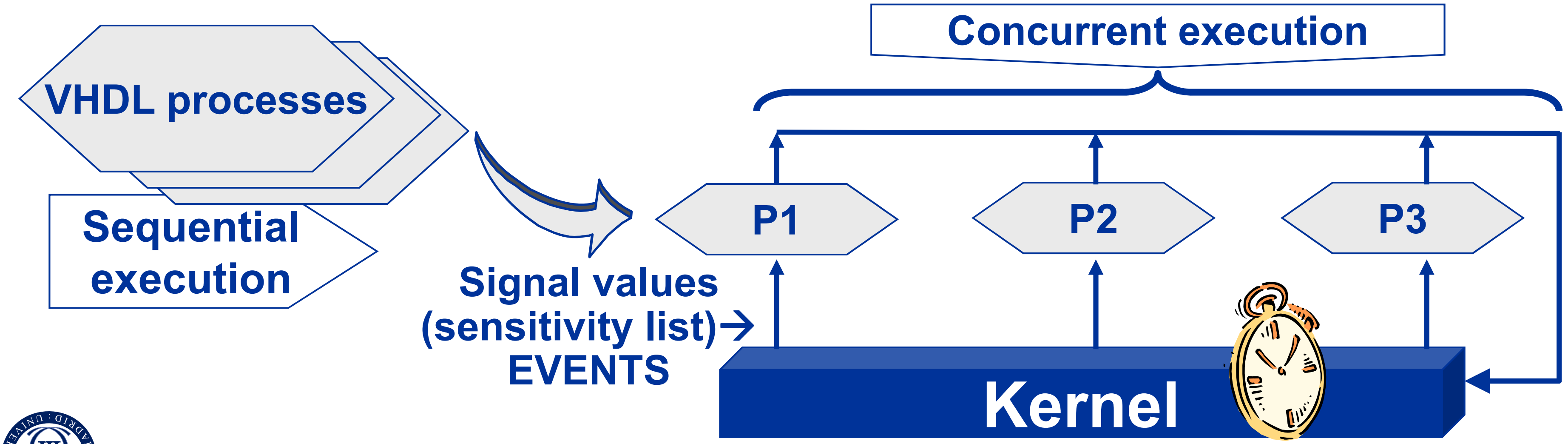
Digital circuit simulation with VHDL

- How does the digital simulator work (HDL)?: **Event simulation**



- Simulator tool is run by a microprocessor: Sequential execution of tasks

- Concurrent behavior

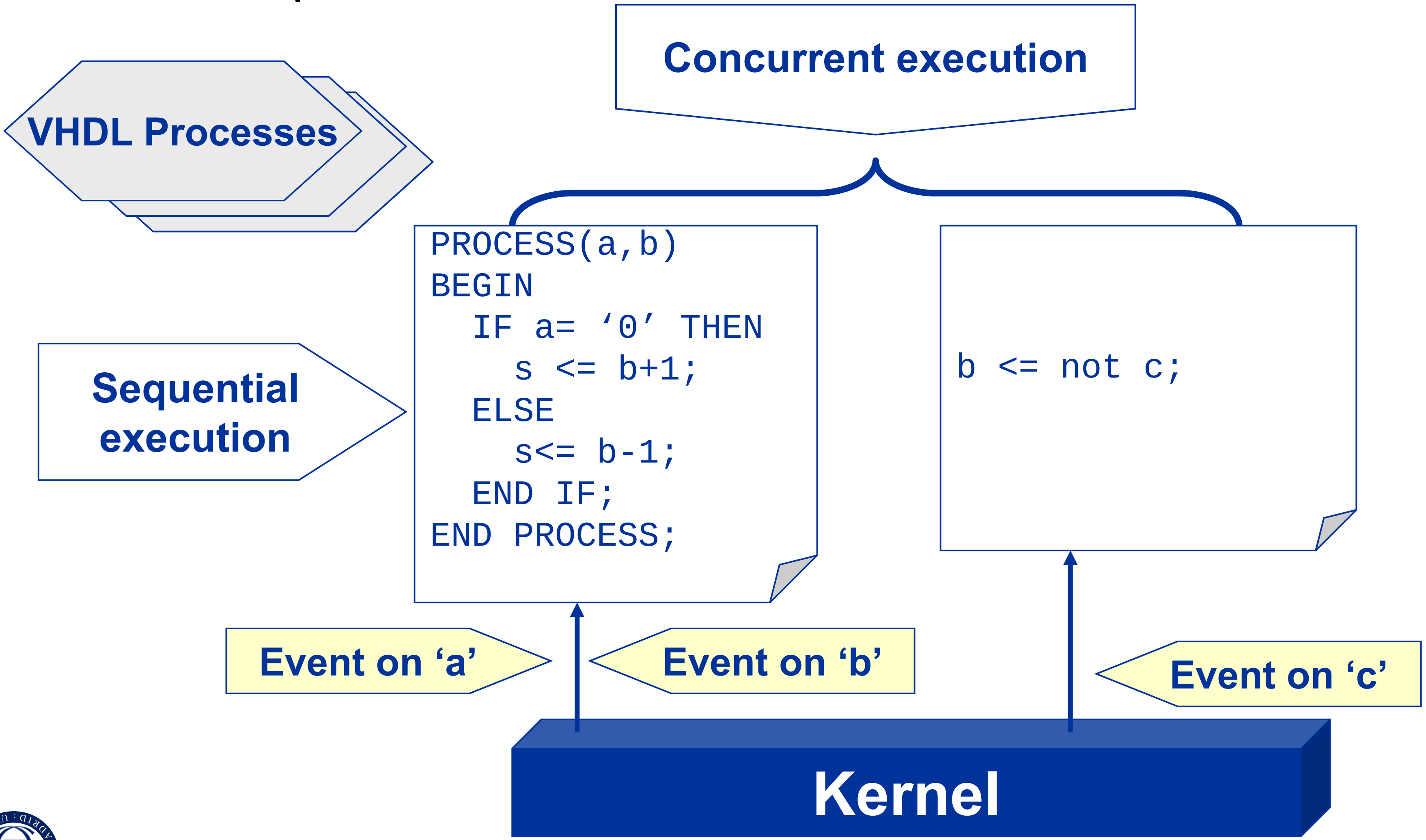


Digital circuit simulation with VHDL

- How does the digital simulator work (HDL)?: **Event-driven simulation**
  - When there is a change (event) in any of the signals included in the sensitivity list, the simulator kernel generates a list with pairs (event, instant).
  - For a given simulation instant, the kernel is in charge of:
    1. Accessing the event list and executing the *process* statements with some pending event for the current instant.
    2. Updating the event list with the new events generated during the process execution.
    3. If some of the new events have been generated for the current instant, go again to 1. Otherwise, time simulation is incremented until the next instant with active events.

Digital circuit simulation with VHDL

- Example

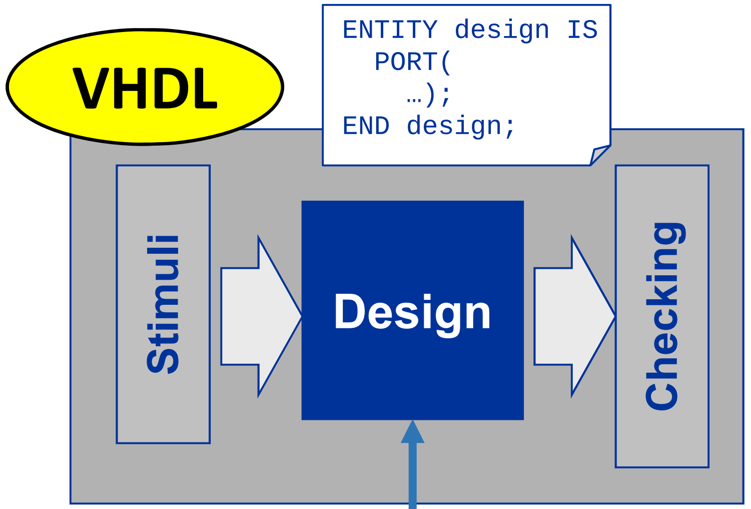


```
PROCESS(a,b)
BEGIN
  IF a= '0' THEN
    s <= b+1;
  ELSE
    s<= b-1;
  END IF;
END PROCESS;
```

```
b <= not c;
```

Test bench design

❑ Test bench: structure



```
ENTITY design IS
  PORT(
    …);
END design;
```

It is implemented as a component, as in hierarchical designs

```
ENTITY tb is
END tb;
ARCHITECTURE sim OF tb is
  COMPONENT design IS
    PORT(
      …);
  END COMPONENT;

SIGNAL s1 : std_logic;
  …
BEGIN

  D: design
  PORT MAP(
      …);
  …

  -- Stimuli generation!!!
  -- Checking!!!
END sim;
```

Empty entity

Component declaration

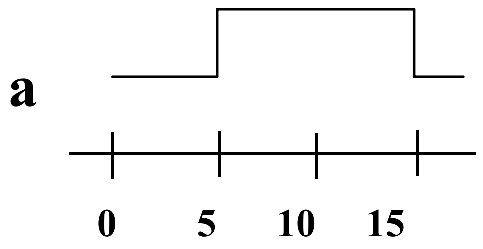
Declaration of signals and constants:
- I/O connections
- …

Component (instance and usage)

Stimuli

Automatic checking

Waveform generation using concurrent statements



```
a <= ‘0’, ‘1’ AFTER 5 NS, ‘0’ AFTER 15 NS;
```

- Concurrent assignment of a waveform
  - Sequence of clauses *<value> AFTER <delay>*
  - Delays are absolute (counted from the execution of the statement, typically at the beginning of the simulation)

WAIT statement

- Used to generate waveforms in sequential style (within a process)
- A WAIT statement suspends the execution of a process until a condition is met

signals	WAIT ON a;	Wait until an event happens on a
time	WAIT FOR 10 ns;	Wait for 10 ns
conditions	WAIT UNTIL c1;	Wait until the condition is true
	WAIT;	Never resumes

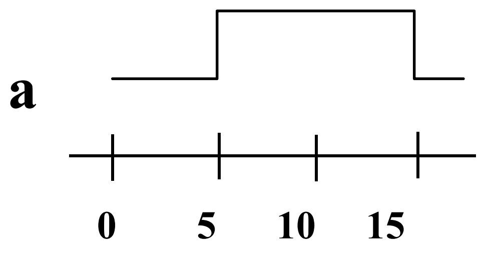
Two types of processes

PROCESSES WITHOUT SENSITIVITY LIST	PROCESSES WITH WAIT STATEMENTS
❖ Execution is suspended at the end of the process ❖ Execution resumes when an event happens in the sensitivity list ❖ WAIT statements are not allowed ❖ They can be synthesized	❖ Execution is cyclic and is only suspended when a WAIT statement is reached (there must be at least one WAIT statement) ❖ Sensitivity list is not allowed ❖ They cannot be synthesized

```
PROCESS (a, b)
BEGIN
        s <= a AND b;
END PROCESS;
```

```
PROCESS
BEGIN
        s <= a AND b;
        WAIT ON a, b;
END PROCESS;
```

Waveform generation using sequential statements



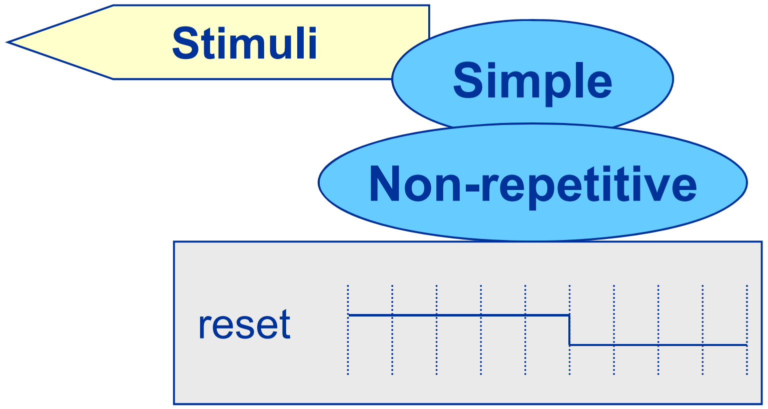
```
PROCESS
BEGIN
        a <= ‘0’;
        WAIT FOR 5 NS;
        a <= ‘1;
        WAIT FOR 10 NS;
        a <= ‘0’;
        WAIT;
END PROCESS;
```

- Sequential assignment of a waveform
  - Assignment followed by WAIT statement
  - Wait times are always relative to the execution time!
  - If a WAIT statement is not included at the end, the waveform is periodic

Stimuli generation: Reset

- Sequential style

```
-- Stimuli generation
PROCESS
BEGIN
      reset <= ‘1’;
      WAIT FOR 50 NS;
      reset <= ‘0’;
      WAIT;
END PROCESS;
```



- Concurrent style

```
-- Stimuli generation
reset <= ‘1’, ‘0’ after 50 ns;
```

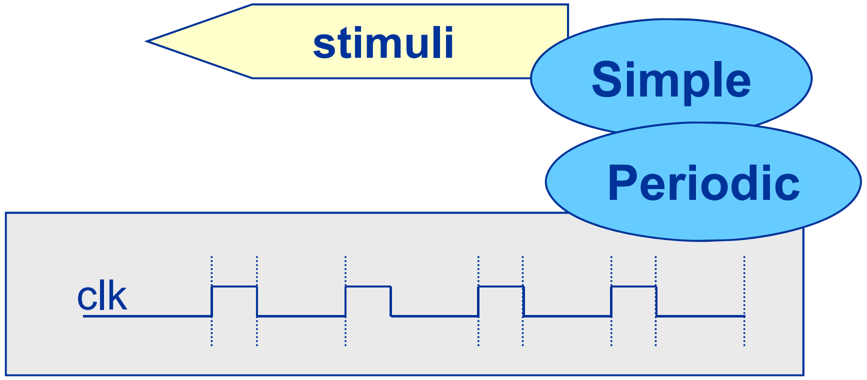
Stimuli generation: clock

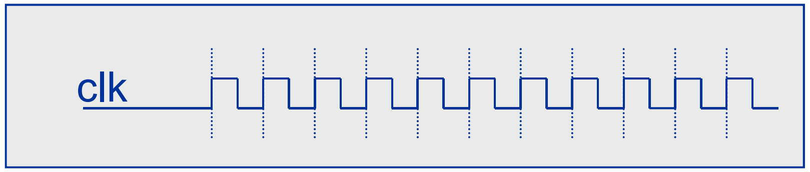
- Asymmetric

```
-- Stimuli generation
PROCESS
BEGIN
        clk <= '1';
        WAIT FOR 10 NS;
        clk <= '0';
        WAIT FOR 40 NS;
END PROCESS;
```

- Symmetric

```
-- Stimuli generation
PROCESS
BEGIN
        clk <= NOT clk;
        WAIT FOR 10 NS;
END PROCESS;
```



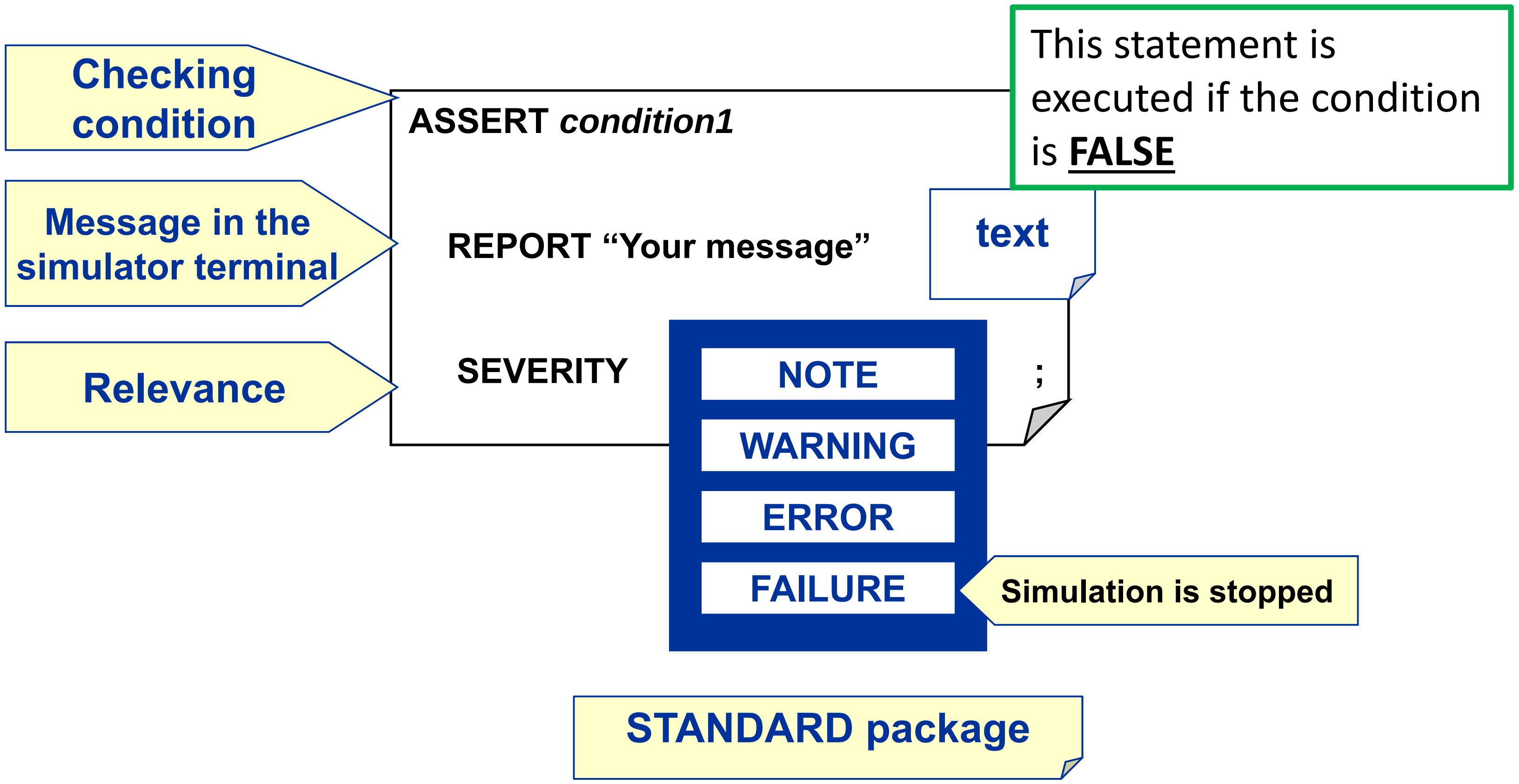


```
-- Initializing the signal is required!
signal clk: std_logic := '0';
```

```
-- Concurrent
clk <= NOT clk AFTER 10 NS;
```

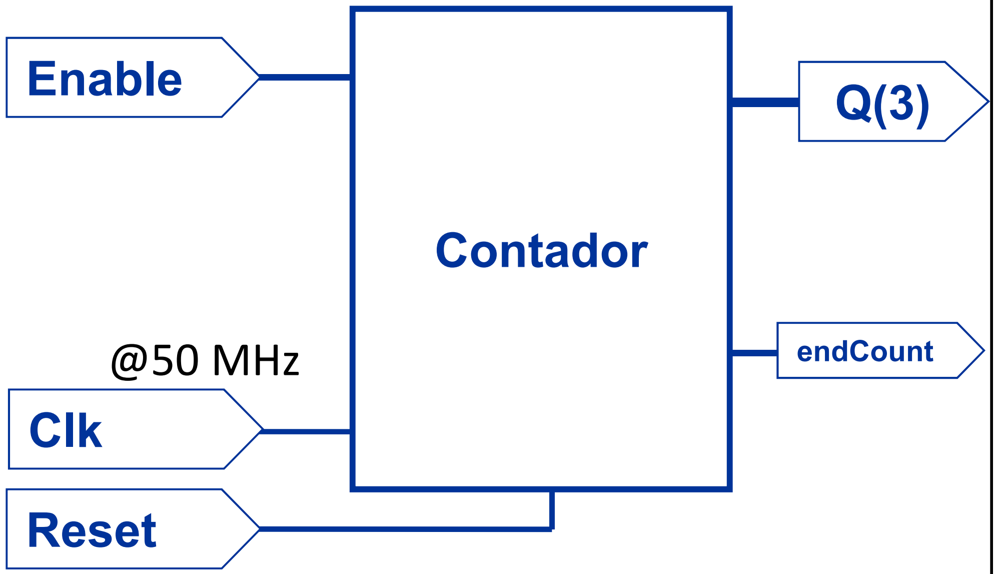
Automatic checking

❑ Testbench: Automatic checking



Digital circuit simulation with VHDL

- Testbench: Example counter from 0 to 5



```
entity counter is
 port(
  clk        : in std_logic;
  reset      : in std_logic;
  enable     : in std_logic;
  endCount : out std_logic;
  Q           : out unsigned(2 downto 0)
 );
end contador ;
```

Note: This example shows a simple testbench, where all the specifications are not checked.

Digital circuit simulation with VHDL

- Testbench: Example counter from 0 to 5

```
ENTITY tb is

END tb;

ARCHITECTURE sim OF tb is
--Component declaration
  component counter
  port(
    clk         : in std_logic;
    reset       : in std_logic;
    enable      : in std_logic;
    endCount : out std_logic;
    Q           : out unsigned(2 downto 0)
  );
end component;
--Signal and constant declarations
signal clk       : std_logic:= ‘0’;
signal reset    : std_logic;
signal enable: std_logic;
signal endCount : std_logic;
signal Q         : unsigned(2 downto 0);
constant T      : time := 20 ns;
```

```
begin
--Component instantation
  MAP_CUT: counter
  port map(
    clk          => clk,
    reset        => reset,
    enable      => enable,
    endCount => endCount ,
    Q              => Q
  );
--Stimuli generation and automatic checkings
clk <= not clk after T/2;

process
begin
  reset    <= ‘1’; --At the begining
  enable <= ‘0’;
  wait for T;
  reset    <= ‘0’; --Disable reset
  wait for T; --Count has not started yet
  enable <= ‘1’;
  wait until endCount = ‘1’; --count has finished
  assert Q = to_unsigned(5,3)
    report “Error: Last value is not 5”
    severity error;
  assert false
     report “End of simulation”
     severity failure;
end process;
```

Stop simulation

Commercial tools

- ❑ Modelsim (Mentor Graphics®) → www.mentor.com/
- ❑ VCS (Synopsys®) → www.synopsys.com
- ❑ Incisive Enterprise Simulator (Cadence®) → www.cadence.com
- ❑ ISIM (Xilinx™) → www.xilinx.com