Simulation with VHDL

Index

- Simulation types in ISE simulator
- How to create a simulation *test-bench*
- *Test-bench* of a combinational entity
- *Test-bench* of a sequential entity
- Basics of the ISE simulator

Simulation types in ISE simulator

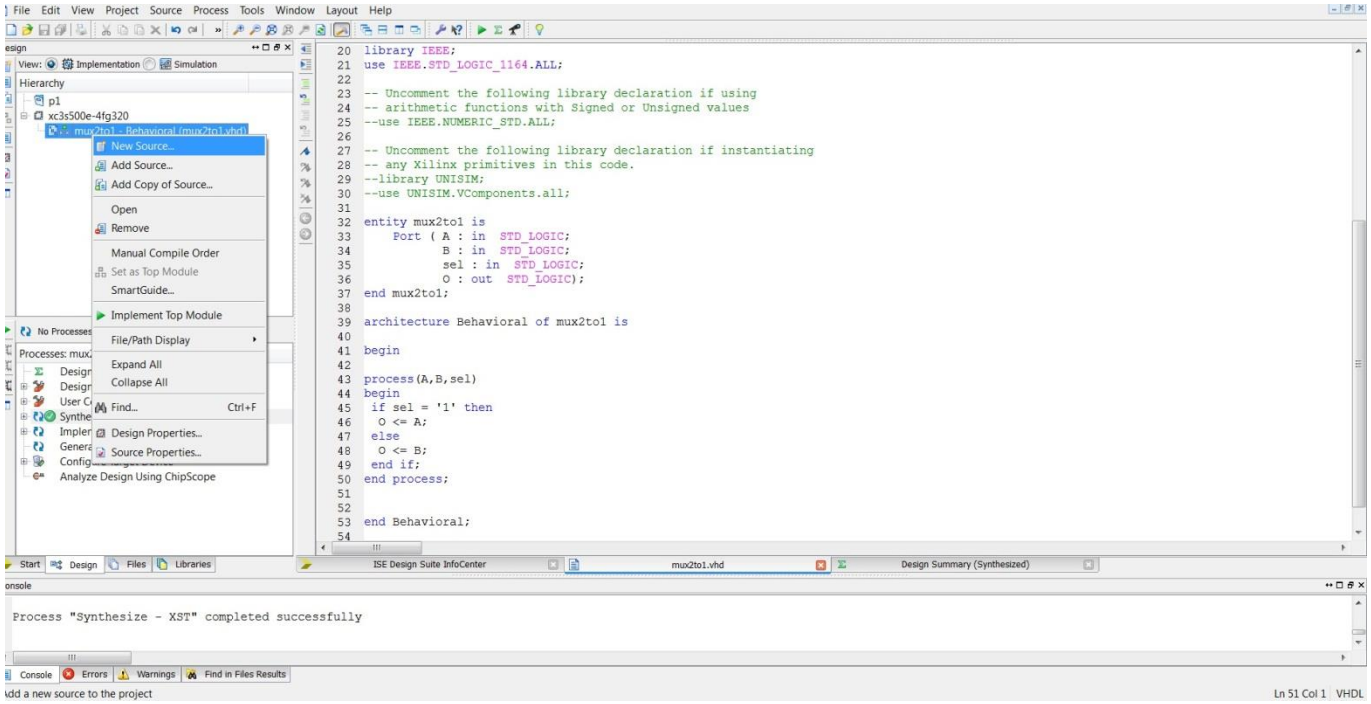
- Xilinx ISE Simulator features four different simulation types
  - *Behavioral* simulation — No delays are considered
  - *Post-Translate* simulation — At each simulation more and more details about delays in logic components and routes are added
  - *Post-Map* simulation
  - *Post-Route* simulation — Representative delay values of the implemention

- We will focus on the *behavioral* simulation that will verify the functional algorithm of the VHDL code

Simulation with VHDL

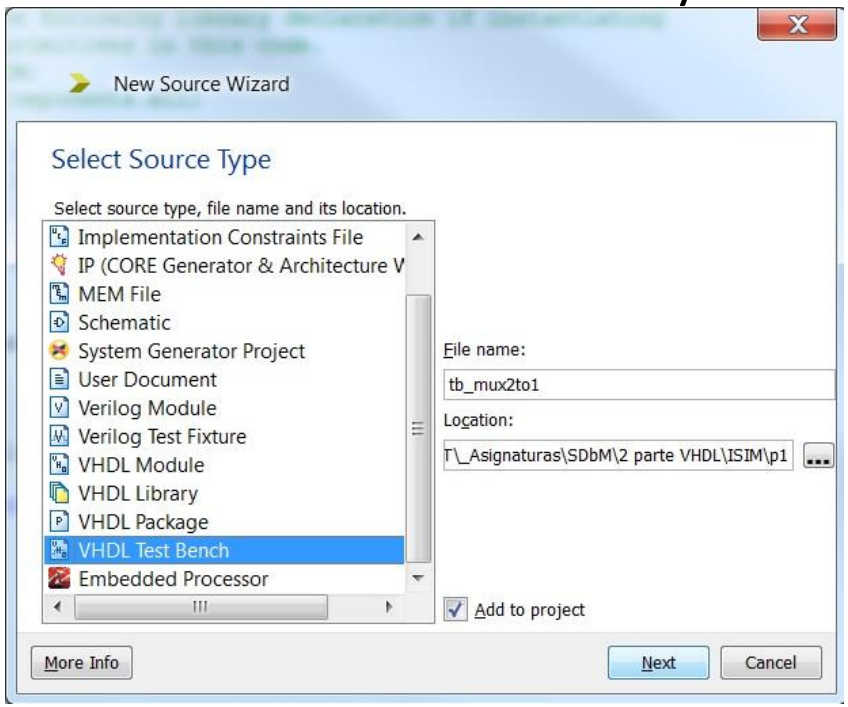
How to create a simulation test-bench

- Test-bench files need to be created to simulate the behavior of an entity: its architecture
- A VHDL test-bench is code that is only used for simulation purposes
- How to add a test-bench to your ISE project



How to create a simulation test-bench

- Test-bench files need to be created to simulate the behavior of an entity: its architecture
- A VHDL test-bench is code that is only used for simulation purposes
- How to add a test-bench to your ISE project




```
BEGIN

   -- Instantiate the Unit Under Test (UUT)
   uut: mux2to1 PORT MAP (
          A => A,
          B => B,
          sel => sel,
          O => O
        );

   -- Clock process definitions
   <clock>_process :process
   begin
      <clock> <= '0';
      wait for <clock>_period/2;
      <clock> <= '1';
      wait for <clock>_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      wait for <clock>_period*10;

      -- insert stimulus here

      wait;
   end process;

END;
```

Test-bench of a combinational entity

- Example: simulation of a comparator
- The VHDL code of the comparator is

```
-- 2 words comparator
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY comparator IS
  PORT ( A: IN std_logic_vector(7 DOWNTO 0);
         B: IN std_logic_vector(7 DOWNTO 0);
         enable:    IN  std_logic;
         AlessB:    OUT std_logic;
         AequalB:   OUT std_logic;
         AgreaterB: OUT std_logic);
END comparator ;

ARCHITECTURE Behavioral OF comparator IS

BEGIN

PROCESS(A, B, enable)
BEGIN
  IF (enable = '0') THEN
    AlessB<='0'; AequalB<= '0'; AgreaterB <= '0';
  ELSE
    IF ( A < B ) THEN --A less than B
      AlessB <= '1';
    ELSE
      AlessB <= '0';
    END IF;
    IF ( A > B ) THEN --A greater than B
      AgreaterB <= '1';
    ELSE
      AgreaterB <= '0';
    END IF;
    IF ( A = B ) THEN --A equals B
      AequalB <= '1';
    ELSE
      AequalB <= '0';
    END IF;
  END IF;
END PROCESS;
END Behavioral;
```

Test-bench of a combinational entity

- The ISE Wizard of the test-bench module generates code to only "fill in the gaps", including clock, **input** and **output ports** of the **entity under test**
- Note that for the simulation of combinational components, the lines dedicated to the clock must be removed
- Set initial values for the input **ports**

```
--Inputs
-- Here initial values for signals can be given. By default 0

  signal A : std_logic_vector(7 downto 0) := (others => '0');
  signal B : std_logic_vector(7 downto 0) := (others => '0');
  --signal enable : std_logic := '0';
  signal enable : std_logic := '1'; -- I change the initial value of enable
```

Test-bench of a combinational entity

- Test-bench wizard generates two processes
  - One for the clock, not applicable for combinational systems
  - One for the rest of inputs

```
-- Stimulus process
  stim_proc: process
  begin
     -- hold reset state for 100 ns.
     wait for 100 ns;

     -- insert stimulus here









     wait; -- wait forever
  end process;
```

Test-bench of a combinational entity

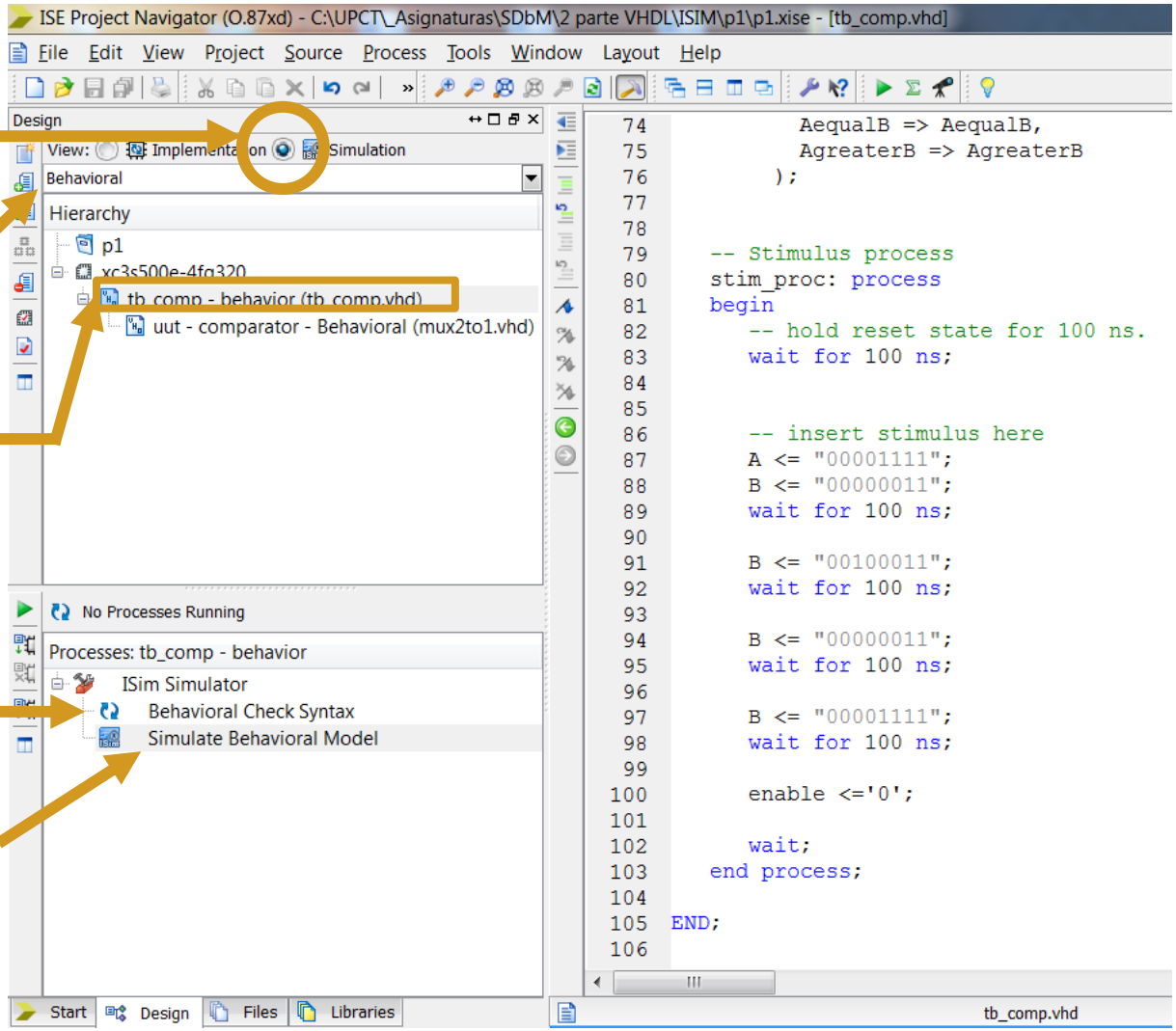
- Test-bench wizard generates two processes
  - One for the clock, not applicable for combinational systems
  - One for the rest of inputs

```
-- insert stimulus here

A <= "00001111"; -- I assign values to A and B
B <= "00000011";
wait for 100 ns; -- time interval between changes in the values

B <= "00100011"; -- A is not changed so no need to repeat
wait for 100 ns;

B <= "00000011";
wait for 100 ns;

B <= "00001111";
wait for 100 ns;

enable <='0';

wait; -- wait forever
```

Test-bench of a combinational entity

View of Xilinx ISE

- Option simulation must be selected
- Simulation type
- Test-bench file must be selected
- Syntax check
- Run behavioral simulation



Simulation with VHDL

- Check syntax and then run the simulation
- *ISim* window will appear

ISim (O.87xd) - [Default.wcfg]

File Edit View Simulation Window Layout Help

Instances and Processes

Instance and Process Name
tb_comp
std_logic_1164

Objects

Simulation Objects for tb_comp

Object Name	Value
a[7:0]	00001111
b[7:0]	00001111
enable	0
alessb	0
aequalb	0
agreaterb	0

Name	Value
a[7:0]	00001111
b[7:0]	00001111
enable	0
alessb	0
aequalb	0
agreaterb	0

999,997 ps 999,998 ps 999,999 ps 1,000,000 ps

X1: 1,000,000 ps

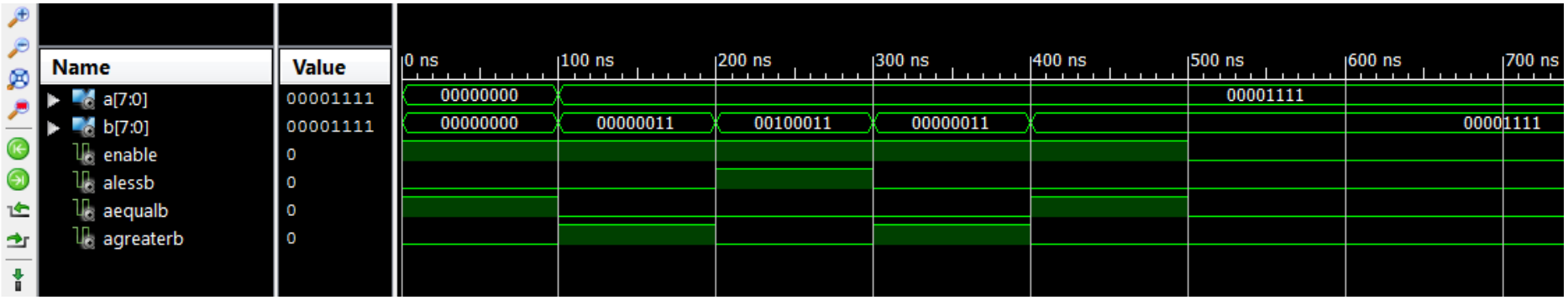
Console

ISim O.87xd (signature 0xc3576ebc)
This is a Full version of ISim.

Test-bench of a combinational entity

- Click full zoom to see all the simulated time

- It can be seen how depending on the value of A and B, outputs change following the time sequence described in the test-bench



- Logical behavior of the entity is verified

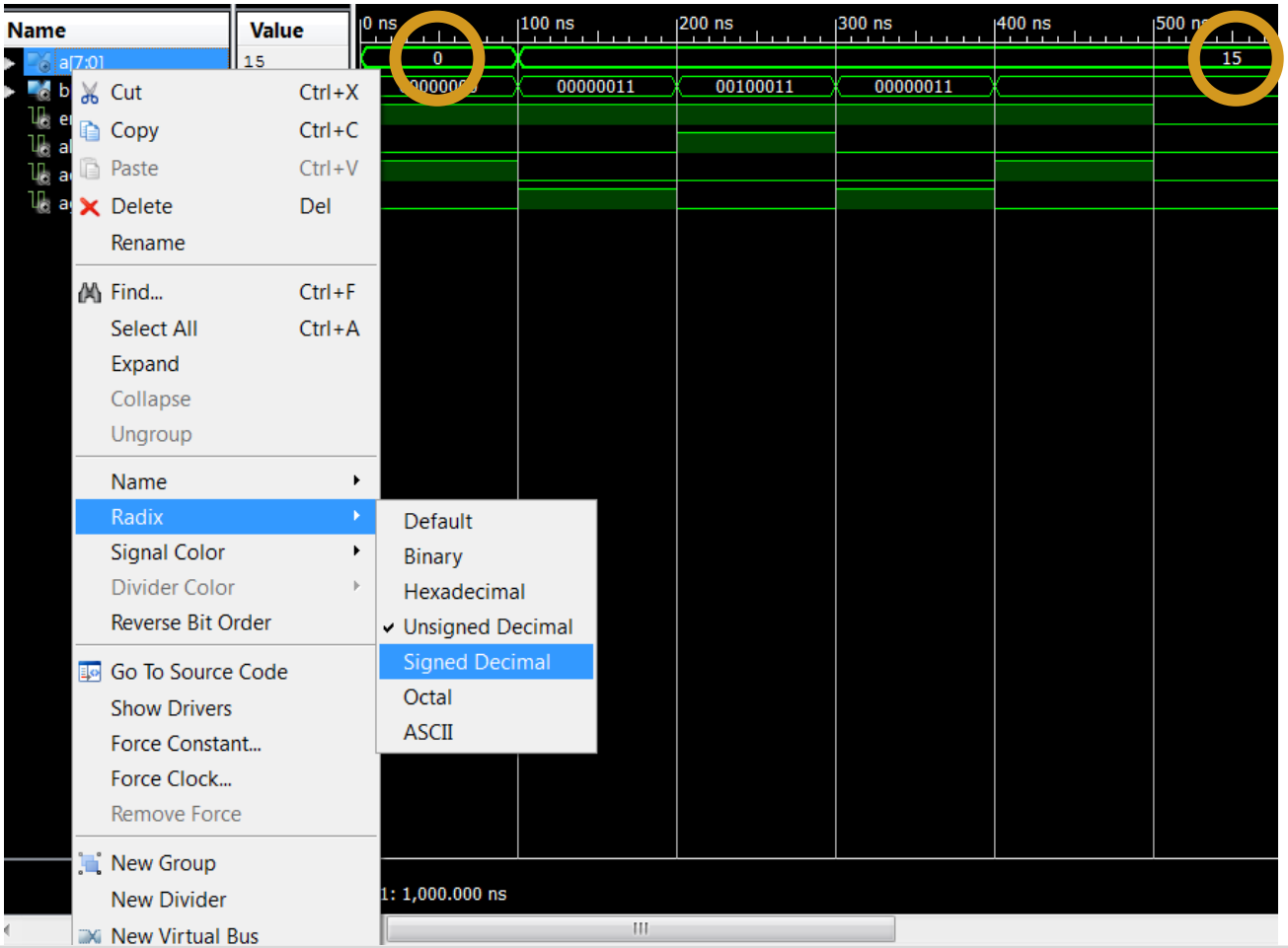
Test-bench of a sequential entity

- A convenient definition is to have the clock and reset defined in different processes. Other inputs remain as for a combinational entity.

```
 -- Clock period definition
constant clk_period : time := 10 ns;
-- ...
 -- Clock process definitions
   clk_process :process
   begin
                clk <= '0';
                wait for clk_period/2;
                clk <= '1';
                wait for clk_period/2;
   end process;

 -- Reset process
rst_process: a_reset <= '1', '0' after 100 ns;
```

Basics of the ISE Simulator

- Change radix of signal



Basics of the ISE Simulator

- If more time is needed in the simulated, run extra time

- Go to previous transition (to see at what time it happened)

- Go to next transition

- Add marker at this instant