Lesson 14

Testbench (Sequential Logic)

Mohd Saufy Rohmad
Technical Trainer and Consultant

Sequential Testbench

- We have already seen how to build a testbench for combinational logic in lesson 13.
- What needs to be added for a testbench of sequential logic? A clock.
- We will test the 8-bit register with asynchronous reset in this section.
- The 8-bit registers' code is given below.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity regis is
port(
clk: in std_logic;
rst: in std_logic;
d_in: in std_logic_vector(7 downto 0); -- input1
d_out: out std_logic_vector(7 downto 0)); -- output, 1 bit wider
end regis;

architecture arch1 of regis is
signal d_R: std_logic_vector(7 downto 0); -- output register
begin
process(clk, rst)
begin
if rst = '0' then
d_R <= (others => '0');
elsif clk'event and clk = '1' then
d_R <= d_in;
end if;
end process;
d_out <= d_R;
end arch1;
```

Register code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity regis_tb is
end regis_tb;

architecture regis_arch_tb of regis_tb is
signal clk: std_logic := '0';
signal rst: std_logic := '0';
signal d_in: std_logic_vector(7 downto 0) := (others=>'0');
signal d_out: std_logic_vector(7 downto 0) := (others=>'0');
constant clk_period : time := 10 ns;

component regis is
port(
clk: in std_logic;
rst: in std_logic;
d_in: in std_logic_vector(7 downto 0); -- input1
d_out: out std_logic_vector(7 downto 0)); -- output, 1 bit wider
end component;
```

Testbench Code

```
begin
dut: regis port map(
clk => clk,
rst => rst,
d_in => d_in,
d_out => d_out);

clk_process :process
begin
clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;
end process;

stim_process:process
begin
wait for 1 ns;
rst <= '1';
wait for 27 ns;
d_in <= "00001111";
wait;
end process;
end regis_arch_tb;
```

Testbench Code

Code Analysis

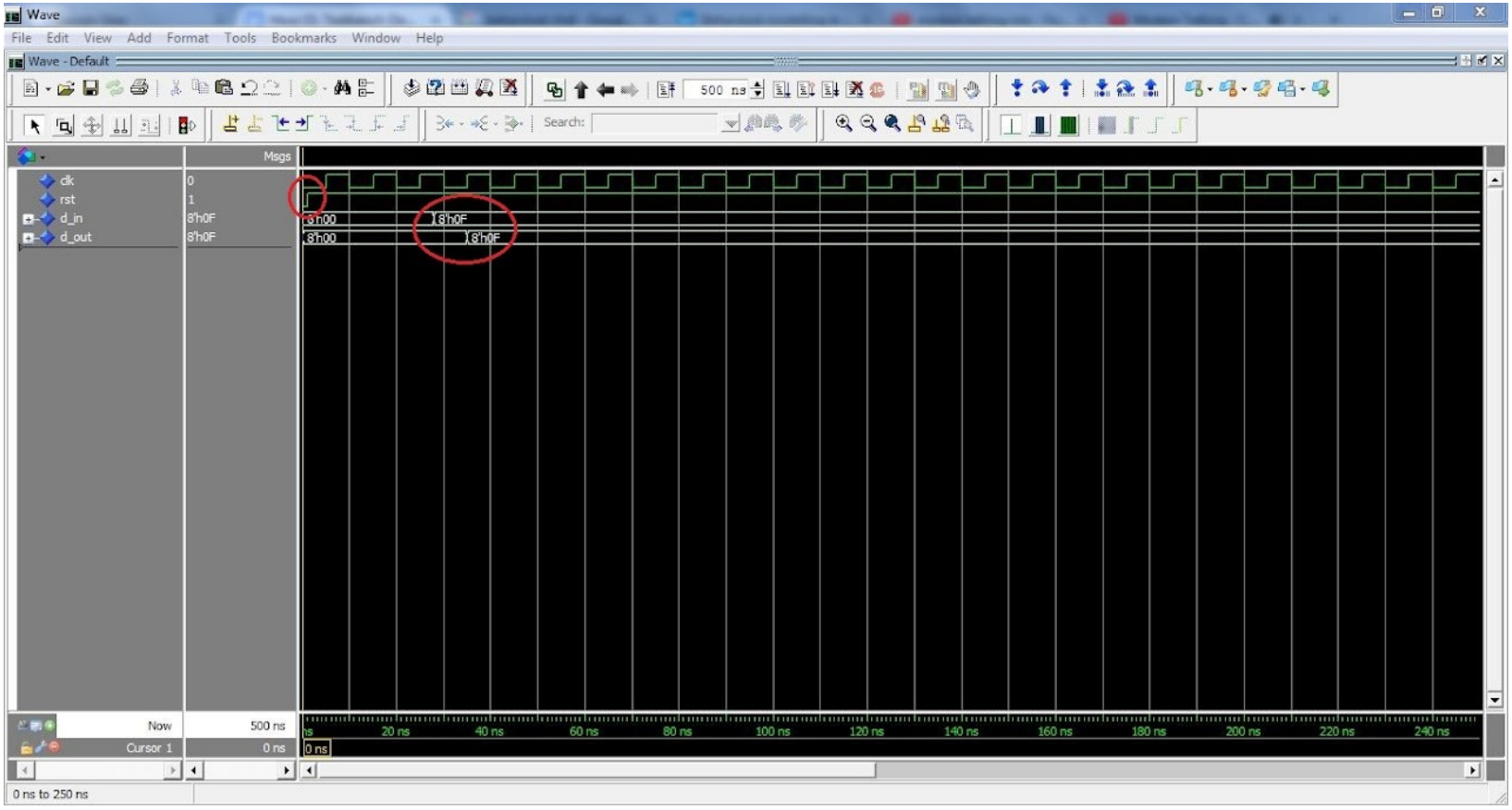
- Here, we created a clock, that is positive for half of the duration of the clock period and negative for the other half.

```
clk_process :process
   begin
        clk <= '0';
        wait for clk_period/2;
        clk <= '1';
    wait for clk_period/2;
end process;
```

Code Analysis

- We created another process, called the stimulus process.
- The process includes the reset and d_in value.
- After 1 ns, the reset becomes 1 and after 27ns d_in gets its value.
- The clocked process and stimulus process runs in parallel.
- If you run the testbench architecture (regis_arch_tb) similarly which is shown in details on lesson 13 you will get the waves as below.

Simulation


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
clk
rst
d_in
d_out
0
1
8'h0F
8'h00
8'h0F
8'h00
8'h0F
Now
500 ns
Cursor 1
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
180 ns
200 ns
220 ns
240 ns
0 ns to 250 ns

Simulation

- In the beginning of the simulation, all the signals are zero as we declared them right after the architecture syntax.
- When reset is zero, the output is also zero.
- The reset becomes 1 after 1ns.
- See the change of reset wave after 1 ns (small circle) and the input/output wave changes on big circle.
- You can see that the d_out is changed only on the positive edge of the clock.

Thank You