Behavioral Simulation with the Vivado Simulator (XSIM)

Introduction

In the previous tutorial (4 - Simple RTL (VHDL) project) we have created a simple RTL project. In this tutorial we will use the Vivado Simulator (XSIM) to validate the behavior of our design.

Run the simulation

To launch the Vivado Simulator for behavioral simulation, click on "*Run Simulation*" under "*Simulation*" in the "*Flow Navigator*" and then click "*Run Behavioral Simulation*". This will open a waveform window.

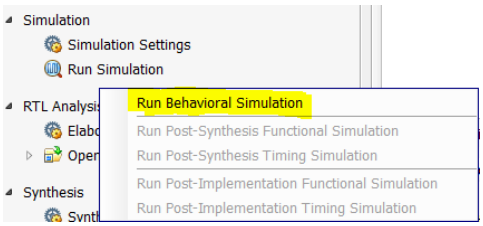

Figure 1 - Run Behavioral Simulation

In the waveform window, on the left, we can see the 4 inputs and the output of our design and the intermediate signals we have created for the output of the OR gate and of the first AND gate. We can see that the value of all the signal is 'U' which means Uninitialized. This is because we haven't given a value to our input signals.

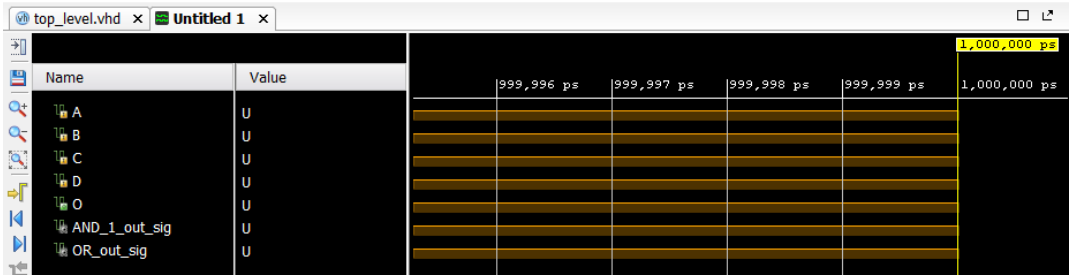

Figure 2 - Waveform window in Vivado

We will give a value to our input signals by forcing their value in the simulation. First, click on "*Run > Restart the Simulation*" to restart the simulation.

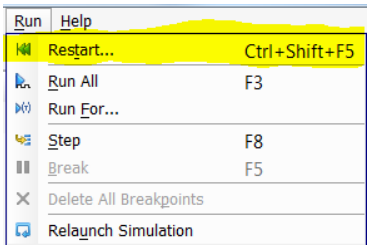

Figure 3 - Restart the simulation - Vivado Simulator

Then we will give a value to our inputs. To test all the possibilities of our system, we will force these signals to have a clock signal. Right click on the signal A and click on "*Force Clock".*

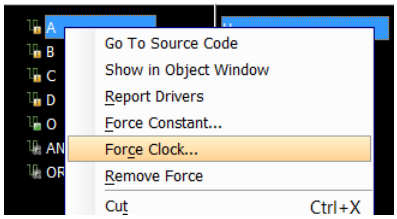

Figure 4 - Force Clock

Set the "*Value radix"* to "*Binary*", the "*Leading edge value"* to '0', the "*Trailing edge value*" to '1' and the period to "*10ns*".

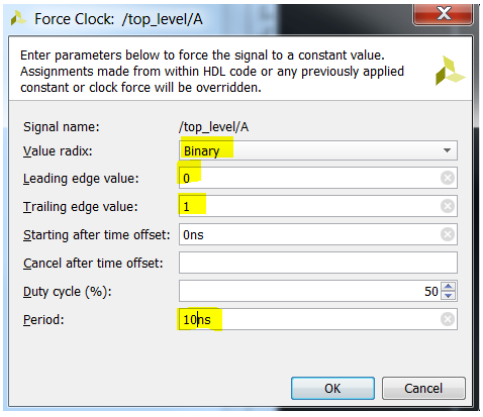

Figure 5 - Force clock on input A

Do the same for the inputs B, C and D but with period value respectively "*20ns*", "*40ns*" and "*80ns*".

Then run the simulation by clicking on "*Run > Run For… "*. Set the simulation time to 160ns.

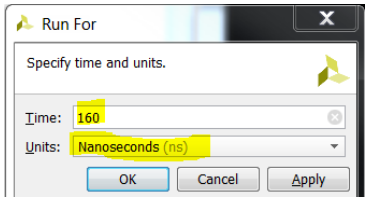

Figure 6 - Run the simulation for 160ns

On the waveform, we can see that O is high in 3 cases:

- When A, B and C are high and D is low

- When A, B and D are high and C is low
- When A, B, C and D are high

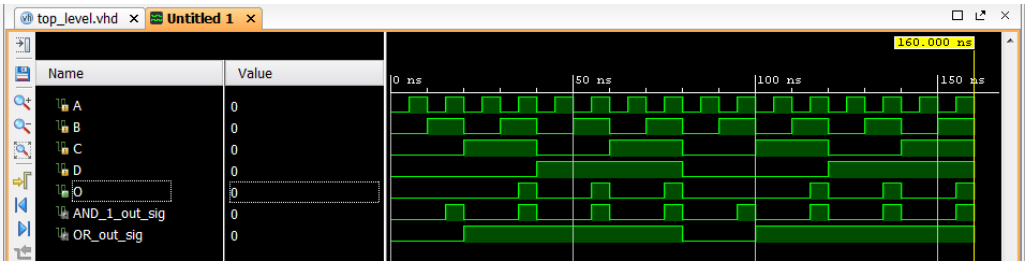

Figure 7 - Vivado Simulator Waveform

This is the expected behavior of our design. Close the simulation window.

Create a test bench (VHDL)

Another way to give a value to our input signals is to create a test bench to simulate our design.

What is a test bench? A test bench is a file written as an HDL file (VHDL, Verilog…) which generally provides a stimuli (inputs, clocks) to a Unit Under Test (UUT).

To create the test bench file in Vivado, click on "*Add Sources*" in the "*Flow Navigator*" and select "*Add or create simulation sources*".

Figure 8 - Add or create simulation sources

Then, click on "*Create File*", select VHDL, enter "*tb_top_level*" as file name, make sure the file type is VHDL and click on finish. Generally, test benches don't have inputs or outputs, so if the "*Define Module*" window appears, remove the first line by selecting it and clicking the "-" button and click on "*OK*".

We can see our created file under the "*Simulation Sources*" in the "*Sources*" window.

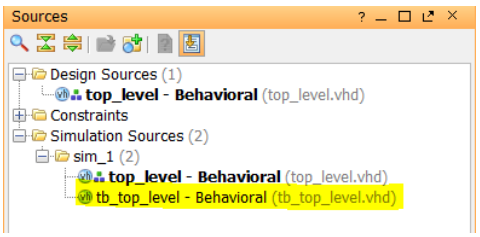

Figure 9 - Simulation Sources in the Sources window

Modify the test bench

Double click on the file in the "*Sources*" window to open it in the text editor. Vivado has already created a template for our file.

1. Declare the top_level bloc

First declare the top_level bloc in the declarative part (between the architecture and the begin keywords):

```
38  architecture Behavioral of tb_top_level is
39
40  component top_level is
41      Port (
42          A : in STD_LOGIC;
43          B : in STD_LOGIC;
44          C : in STD_LOGIC;
45          D : in STD_LOGIC;
46          O : out STD_LOGIC
47      );
48  end component top_level;
49
50  begin
```

Figure 10 - top_level bloc declaration

2. Declare the signals

Then declare all the signals we will use. We will connect all the top_level inputs and outputs so declare these signals in the declarative part and set the initial inputs values to '0':

```
49
50      signal A : STD_LOGIC := '0';
51      signal B : STD_LOGIC := '0';
52      signal C : STD_LOGIC := '0';
53      signal D : STD_LOGIC := '0';
54      signal O : STD_LOGIC;
55
56  begin
```

Figure 11 - Signals declaration

3. Instantiate the Unit Under Test (UUT)

Then instantiate the top_level bloc, that we will call UUT (for Unit Under Test), in the statement part (after the begin keyword) by connecting the signals:

```
58      UUT : top_level
59      Port map(
60          A => A,
61          B => B,
62          C => C,
63          D => D,
64          O => O
65      );
```

Figure 12 - Instantiate the UUT

4. Generate the stimuli

Then assign value to the input signals. A will be toggled every 10 ns, B every 20 ns, C every 40 ns and D every 80 ns. All the possible combinations will be tested after 160ns (simulation time).

```
66
67      A <= not A after 10 ns;
68      B <= not B after 20 ns;
69      C <= not C after 40 ns;
70      D <= not D after 80 ns;
71
72  end Behavioral;
```

Figure 13 - Stimuli generation

Run the behavioral simulation with test bench

Run the behavioral simulation by clicking "*Run Simulation > Run Behavioral Simulation*" under "*Simulation"* in the "*Flow Navigator*".

We can see the same waveform as in the first part of this tutorial.


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

entity tb_top_level is
--  Port ( );
end tb_top_level;

architecture Behavioral of tb_top_level is

component top_level is
    Port (
        A : in STD_LOGIC;
        B : in STD_LOGIC;
        C : in STD_LOGIC;
        D : in STD_LOGIC;
        O : out STD_LOGIC
    );
end component top_level;

    signal A : STD_LOGIC := 0;
    signal B : STD_LOGIC := 0;
    signal C : STD_LOGIC := 0;
    signal D : STD_LOGIC := 0;
    signal O : STD_LOGIC;

begin
    UUT : top_level
    Port map(
        A => A,
        B => B,
        C => C,
        D => D,
        O => O
    );

    A <= not A after 10 ns;
    B <= not B after 20 ns;
    C <= not C after 40 ns;
    D <= not D after 80 ns;

end Behavioral;
```