How to write a VHDL Test Bench

We need to write VHDL Test Benches in order to simulate our circuits. In other words we need to generate input waveforms and let the simulation tool of Xilinx ISE generate the output waveforms. Xilinx ISE 11.1 can make use of several simulators. One of them is Isim which comes with the webpack download. Another one is Modelsim which must be downloaded from the Xilinx web site or from the Modelsim web site. Student version of Modelsim is free. In EEE102 it suffices to use the ISim simulator.

In previous versions of Xilinx ISE (up to version 10.3) there was a tool called "Test Bench Waveform" which is a visual tool to define the input waveforms. In Xilinx 11.1 this tool is discontinued. In any case, writing your own VHDL Test Bench is something you must learn in order to create more flexible simulation input waveforms especially needed in large projects. Let us now illustrate how to write a VHDL Test Bench by a simple example. Suppose we have the following VHDL description for a simple circuit: F=A+B'C.

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

entity simple is
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           C : in STD_LOGIC;
           F : out STD_LOGIC);
end simple;

architecture Behavioral of simple is
begin
        F <= A or ((not B) and C);
end Behavioral;
```

From the "Project" tab we choose "New Source" and then "VHDL Test Bench". We give the name "simple_testbench" to our VHDL Test Bench. From the Sources for Behavioral Simulation we find that a VHDL Test bench template is already generated for us. This simulation VHDL code is given below:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY simple_testbench IS
END simple_testbench;

ARCHITECTURE behavior OF simple_testbench IS

-- Component Declaration for the Unit Under Test (UUT)
  COMPONENT simple
  PORT(
        A : IN std_logic;
```

```
        B : IN std_logic;
        C : IN std_logic;
        F : OUT std_logic
        );
  END COMPONENT;

--Inputs
signal A : std_logic := '0';
signal B : std_logic := '0';
signal C : std_logic := '0';

--Outputs
signal F : std_logic;

BEGIN
-- Instantiate the Unit Under Test (UUT)
uut: simple PORT MAP (
        A => A,
        B => B,
        C => C,
        F => F
);

-- No clocks detected in port list. Replace <clock> below with
-- appropriate port name

constant <clock>_period := 1ns;
<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 100ms.
        wait for 100ms;
        wait for <clock>_period*10;
        -- insert stimulus here
wait;

end process;

END;
```

Above testbench template is fairly general. Specifically since we do not have a clock in our circuit we may eliminate the clock related parts. In later projects in which you have clocked circuits you will need these parts as well. We may edit the VHDL Test Bench to suit our simple circuit and the edited version is given below:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY simple_testbench IS
END simple_testbench;

ARCHITECTURE behavior OF simple_testbench IS
-- Component Declaration for the Unit Under Test (UUT)
COMPONENT simple
PORT(
        A : IN std_logic;
        B : IN std_logic;
        C : IN std_logic;
        F : OUT std_logic
);
END COMPONENT;

--Inputs
signal A : std_logic := '0';
signal B : std_logic := '0';
signal C : std_logic := '0';

--Outputs
signal F : std_logic;

BEGIN
-- Instantiate the Unit Under Test (UUT)
uut: simple PORT MAP (A => A,B => B,C => C,F => F);
-- Stimulus process
stim_proc: process
  begin
        A<='0';B<='0';C<='0';
        wait for 100ns;
        A<='0';B<='0';C<='1';
        wait for 100ns;
        A<='0';B<='1';C<='0';
        wait for 100ns;
        A<='0';B<='1';C<='1';
        wait for 100ns;
        A<='1';B<='0';C<='0';
        wait for 100ns;
        A<='1';B<='0';C<='1';
        wait for 100ns;
        A<='1';B<='1';C<='0';
        wait for 100ns;
        A<='1';B<='1';C<='1';
        wait for 100ns;
    end process;
END;
```

When we run Isim we get the following result which contains the graphs of the input waveforms that we have generated and also the waveform of the output F calculated by ISim. Note that in the VHDL code for “simple_testbench” there is an entity declaration but it has no ports (inputs and outputs). In the architecture, the actual entity “simple” is used as a component and the input waveform are generated by using “wait for” commands. The above example was a simple one and we generated the input waveforms in 8 steps to cover all cases of the 3-input circuit F=A+B’C. However in larger projects you can use vector variables and generate input waveforms using all constructs of VHDL such as “for loop”s, “if”s, and others.