EEE102 LAB 6-ARBITRARY WAVEFORM GENERATOR

Name: Burak Gökkaya

ID:22202209

Section: 001

Introduction

The purpose of this lab was to design a VHDL code which generates digital waveforms with using Xilinx IP's (clocking wizard). So that we can use the IP catalog in our project. For making this lab, I learned that what is Xilinx IP's and how I can use that. Then I designed a VHDL code properly.

Implementation

- We are wanted to design a VHDL code which generates waveforms with using Xilinx IP's. Therefore, I made research about clocking wizard IP. I used two inputs (reset and clock_in1) of it and 1 output (clock_out1). After instantiating IP, I finished my code part and implemented it successfully.

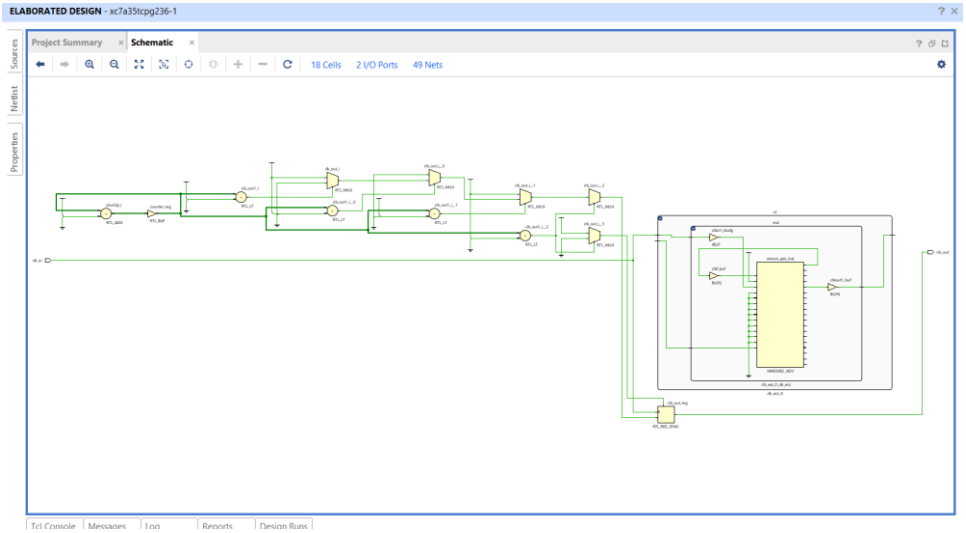

Figure 1. Schematic of arbitrary waveform generator

- Then I wrote test bench code. After that, I run simulation and compare the results if they are like what I wanted.

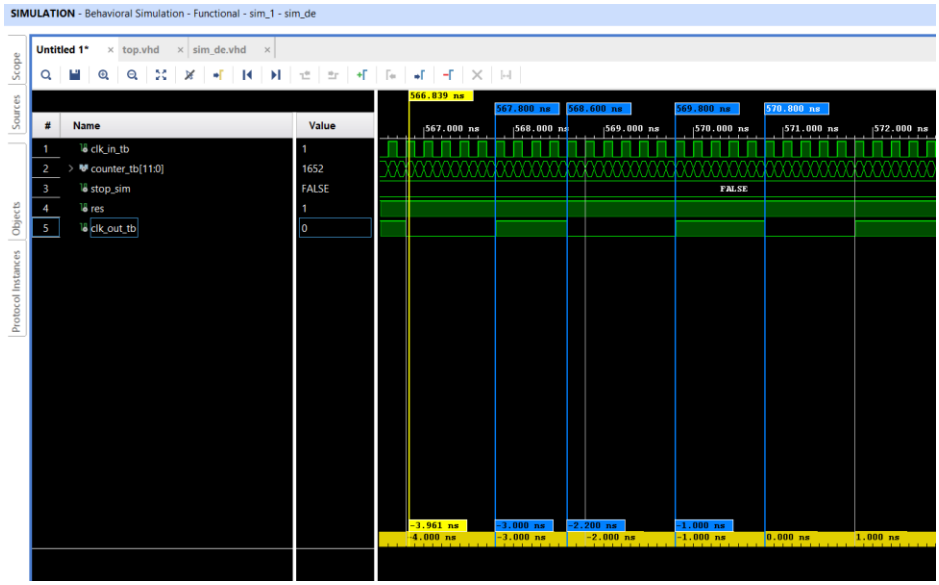

Figure 2. Simulation screen (waves frequencies are different as we can see)

Conclusion

In conclusion, this lab was very helpful for us. We learned to use IP and we can use these catalogs in our project. Also, we used oscilloscope and BASYS3 again and it was helpful for us; we used them in more efficient way. So it was successful lab for me.

Appendix

```
#Main Code
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;


entity top is
    Port(
        clk_in : in std_logic;
        clk_out : out std_logic
    );
end top;


architecture Behavioral of top is
    signal res : std_logic;
    signal final : std_logic;
```

```
    signal counter : unsigned(11 downto 0) := (others => '0');
    component clk_wiz_0
      port(
        clk_in1 : in std_logic;
        reset : in std_logic;
        clk_out1 : out std_logic
      );
    end component;

begin
    u1: clk_wiz_0
      port map (
        clk_in1 => clk_in,
        reset => res,
        clk_out1 => final
      );

    process(clk_in)
    begin
      if res = '0' then
        counter <= (others => '0');
      end if;
        if counter < 7000 then
          counter <= counter + 100;
          if rising_edge(clk_in) then

              if counter < 500 then
                clk_out <= '0';
              elsif counter < 1500 then
                clk_out <= '1';
              elsif counter < 2500 then
```

```
                clk_out <= '0';
            elsif counter < 3400 then
                clk_out <= '1';
            elsif counter < 4700 then
                clk_out <= '0';
            elsif counter < 6500 then
                clk_out <= '1';
            else
                clk_out <= '0';
            end if;
        end if;
    end if;
  end process;
end Behavioral;

##Simulation Code
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;

entity sim_de is
end sim_de;

architecture testbench of sim_de is
  signal clk_in_tb : std_logic := '0'; -- Testbench clock signal
  signal counter_tb : unsigned(11 downto 0) := (others => '0'); -- Başlangıç değeri olarak sıfır atanır.
  signal stop_sim : boolean := false;  -- Simulation stop signal
  signal res : std_logic := '0';       -- Reset signal for DUT
  signal clk_out_tb : std_logic;       -- Output signal from the DUT

  component top
```

```
    Port(
        clk_in : in std_logic;
        clk_out : out std_logic
    );
  end component;

begin
  -- Instantiate the DUT (Design Under Test)
  dut : top
    port map (
        clk_in => clk_in_tb,
        clk_out => clk_out_tb
    );

  -- Clock process for the testbench
  clk_process: process
  begin
    while not stop_sim loop
        clk_in_tb <= not clk_in_tb; -- Toggle the clock signal
        if counter_tb < 7000 then
            counter_tb <= counter_tb +100;
        end if;
        wait for 0.1 ns;            -- 10 ns clock period
    end loop;
    wait;
  end process clk_process;

  -- Stimulus process
  stim_process: process
  begin
    res <= '0'; -- Initialize reset signal
```

```
        wait for 100 ns; -- Wait for some time before starting

        -- Provide stimulus to the design
        res <= '1'; -- Release reset
        wait for 10000ns; -- Wait for simulation time

        -- Stop simulation
        stop_sim <= true;
        wait;
    end process stim_process;

end testbench;
```