EEE102 Lab 6 Report

Nizam Ercan 22302317 EEE102-2

1- Purpose:

The aim of this lab session is to get used to generating waveforms for our needs by generating a clock with arbitrary frequency from Clocking Wizard IP in Vivado.

2- Methodology:

I need to generate a clock with suitable frequency. Therefore, I will be using the Clocking Wizard IP in Vivado to generate a custom clock with a frequency of 10MHz. Then I will write counters triggered by the rising edges of this clock. Finally, I will write the code for generating an arbitrary waveform by using these counters. The waveform will be low for 2.5 ms and high for 7.5 ms. I will also write the constraints file and the testbench to see whether the design works properly or not before implementing it on my FPGA.

3- Desing Specifications:

I need a custom clock with a frequency of 10MHz. In order to achieve this, I found the “Clocking Wizard IP” in the IP catalog of Vivado. As you can see in Figure 1, in the Clocking Wizard, under the Output Clocks menu I created one output clock “clk_out1” with 10MHz.

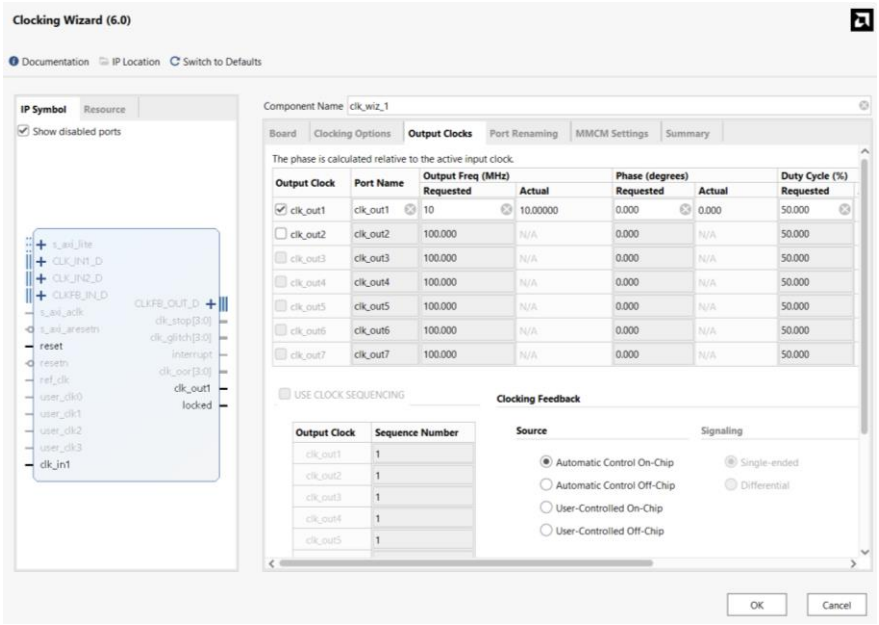

Figure 1: The clocking wizard

After creating a clock with 10 MHz I instantiated this module in my main code. Then, I wrote one counter that counts until 100000 which corresponds to 10 ms. After 100000, the counter comes back to 0 which makes the whole process repeat itself. Also, until the counter counts to 25000, my output

will be ‘0’ and between 25000 and 100000, the output will be ‘1’. These conditions make the output ‘0’ for 2.5 ms and ‘1’ for 7.5 ms which is what I wanted to achieve.

After that, I wrote the constraint file where I connected my output waveform to A14 which corresponds to the JB1 pin in my Basys3. I will connect this pin to an oscilloscope to see whether the waveform is generated correctly or not.

Finally, I wrote the testbench for my design and checked whether the waveform that is low for 2.5 ms and high for 7.5 ms is generated or not.

4- Results:

The final schematics for my main module can be seen in Figure 2.

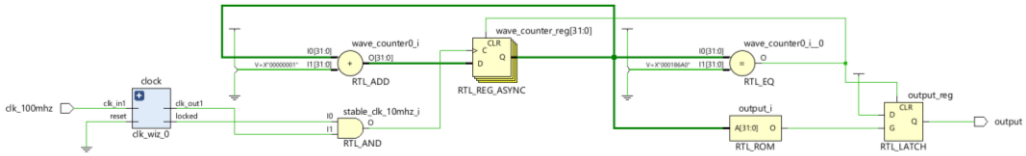

Figure 2: RTL schematics for my main module

Also, the schematics for the custom clock generated by Clocking Wizard can be seen in Figure 3.

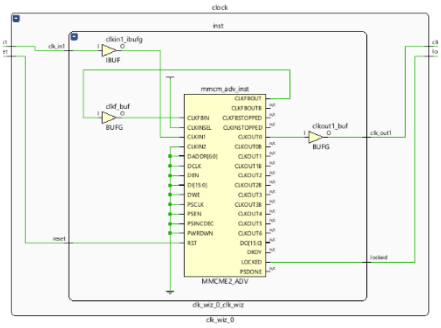

Figure 3: RTL schematics for my custom clock

Before implementing this design to my Basys3, I started the simulation. As you can see in Figure 4, the testbench works properly where the output signal is low for the first 2.5 ms and high for 7.5 ms.

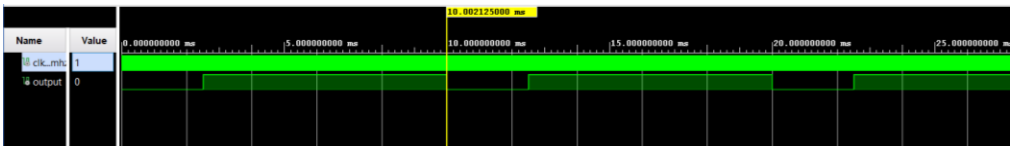

Figure 4: my testbench

After seeing that my design works properly in the testbench, I generated the bitstream and implemented my code to Basys3. I connected the output port to an oscilloscope and observed the waveform generated by my design. In Figure 5, you can see that my waveform is generated as intended having a period of 10 ms, high for 7.5 ms and low for 2.5 ms.

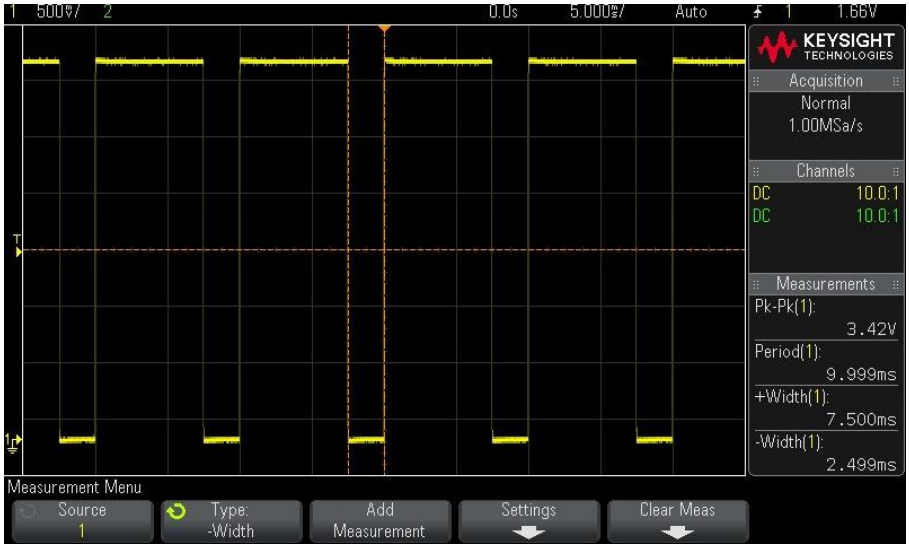

Figure 5: My waveform measured by the oscilloscope

5- Conclusion:

In the end, I managed to achieve the lab's goal by creating an arbitrary waveform and using the Clocking Wizard IP to generate a clock with custom frequency. There was a very small delay in the simulation since it takes some time for the computer to process the code. However, the delay was so small that it was negligible. Also, in the oscilloscope, I measured the period to be 9.999 ms instead of 10 ms and – width 2.499 ms instead of 2.5 ms. However, these results are very close to the actual results and happened because of the internal errors of the oscilloscope. In the end, I was able to generate my arbitrary waveform that is low for 2.5 ms and high for 7.5 ms successfully.

6- References:

https://www.youtube.com/watch?v=ngkpvMaNapA

https://support.xilinx.com/s/question/0D52E00006iHlm5SAC/vivado-clock-ip wizard?language=en_US

https://pdf1.alldatasheet.com/datasheet-pdf/download/1132104/ETC2/MG90S.html

7- Appendices:

waveform.vhd:

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;



entity waveform is

    Port ( clk_100mhz : in STD_LOGIC;

           output : out STD_LOGIC := '0');

end waveform;
```

```
architecture Behavioral of waveform is

    component clk_wiz_0
    port
     (-- Clock in ports
      -- Clock out ports
      clk_out1          : out    std_logic;
      -- Status and control signals
      reset             : in     std_logic := '0';
      locked            : out    std_logic := '1';
      clk_in1           : in     std_logic
     );
    end component;

    signal locked : std_logic;
    signal reset : std_logic;
    signal clk_10mhz: std_logic;
    signal stable_clk_10mhz: std_logic;
    signal wave_counter: integer := 0;

begin

    clock : clk_wiz_0
       port map (
      -- Clock out ports
       clk_out1 => clk_10mhz,
      -- Status and control signals
       reset => reset,
       locked => locked,
       -- Clock in ports
       clk_in1 => clk_100mhz
     );

    stable_clk_10mhz <= locked and clk_10mhz;
    process(stable_clk_10mhz)
    begin

        if rising_edge(stable_clk_10mhz) then
```

```
                    wave_counter <= wave_counter + 1;
                    end if;
            if wave_counter = 25000 then
                output <= '1';
            end if;
            if wave_counter = 100000 then
                output <= '0';
                wave_counter <= 0;
            end if;

    end process;

end Behavioral;
```

sim.vhd:

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



entity sim is
end sim;

architecture Behavioral of sim is

    component waveform is
        Port ( clk_100mhz : in STD_LOGIC;
               output : out STD_LOGIC);
    end component;

    signal clk_100mhz: std_logic;
    signal output: std_logic;

begin

    dut: waveform port map(clk_100mhz => clk_100mhz, output => output);

    process
    begin
```

```
        clk_100mhz <= '0';
            wait for 5 ns;
        clk_100mhz <= '1';
            wait for 5 ns;



    end process;
end Behavioral;
```

constraints.xdc:

```
set_property IOSTANDARD LVCMOS33 [get_ports output]
set_property IOSTANDARD LVCMOS33 [get_ports clk_100mhz]
set_property PACKAGE_PIN W5 [get_ports clk_100mhz]
set_property PACKAGE_PIN A14 [get_ports output]
```